MAN05010.250625

Linguaggio di programmazione R3

© 2025 Robox SpA

I contenuti di questo documento sono redatti con la massima cura/diligenza, e sottoposti ad un accurato controllo. Robox Spa tuttavia, declina ogni responsabilità, diretta e indiretta, nei confronti degli utenti e in generale di qualsiasi terzo, per eventuali imprecisioni, errori, omissioni, danni (diretti, indiretti, conseguenti, punibili e sanzionabili) derivanti dai suddetti contenuti.

Tabella dei contenuti

Parte I Indice generale 17

Parte II Costanti 17

1 Costanti 17

Parte III Funzioni/istruzioni 19

1 Indice funzioni/istruzioni 20
2 Codici di errore funzioni 59
3 Codici di reso dalle funzioni MV_ e MVA_ 63
4 Codici di errore SDO (CoE/SoE) 65
5 Codici di errore Modbus 68
6 Tipi di ciclo di zero 69
- Asse lineare: ciclo di zero a micro 69
- Asse lineare: ciclo di zero a micro + index 74
- Asse lineare: ciclo di zero a paracarro 79
- Asse lineare: ciclo di zero a paracarro + index 82
- Asse lineare: ciclo di zero sulla posizione 86
- Asse rotativo monodirezionale: ciclo0 a micro + index con ricerca preliminare della variazio..... 88 l micro
- Asse rotativo monodirezionale: ciclo0 a micro + index SENZA ricerca preliminare della varia..... 94 del micro
- Asse rotativo monodirezionale: ciclo0 solo a micro con ricerca preliminare della variazione..... 99icro
- Asse rotativo monodirezionale: ciclo0 solo a micro senza ricerca preliminare della variazi..... 104l micro

7 Categorie 109
- Funzioni/istruzioni matematiche 109
- Funzioni/istruzioni safety 111
- Funzioni/istruzioni supporto rule/task 112
- Gestione allarmi 114
- Gestione Axioline 116
- Gestione canali seriali 116
- Gestione CANopen/EtherCAT 118
- Gestione CNET 120
- Gestione comunicazione BCC 121
- Gestione controllo di flusso 123
- Gestione conversione endianess 125
- Gestione counter 130
- Gestione dati 131
- Gestione flash 136
- Gestione forzatura I/O 138
- Gestione modbus/TCP client 138
- Gestione MOT 142
- Gestione movimento 142
- Gestione power set 145
- Gestione stringhe 146
- Gestione timer 148
- Gestione UDP/IP 148
- Gestione variabili globali 149
- Gestioni miscellanee 152

8 Funzioni 156

© 2025 Robox SpA

Funzione abs() 156
Funzione abs_mod() 157
Funzione acos() 157
Funzione ah_cmd() 158
Funzione ah_get_d() 159
Funzione ah_get_s() 160
Funzione ah_info() 162
Funzione ah_log() 163
Funzione ah_log_us() 164
Funzione al2am() 166
Funzione alarm_reset() 169
Funzione alarm_reset_par() 170
Funzione alarm_reset_par_us() 172
Funzione alarm_set() 175
Funzione alarm_set_us() 176
Funzione as_get_d() 178
Funzione asin() 178
Funzione atan() 179
Funzione atan2() 180
Funzione axio_iol_r 181
Funzione axio_iol_w 182
Funzione axio_pm_r() 183
Funzione axio_robj() 186
Funzione axio_wobj() 199
Funzione b0() 213
Funzione b1() 215
Funzione b2() 215
Funzione b3() 216
Funzione b4() 216
Funzione b5() 217
Funzione b6() 218
Funzione b7() 218
Funzione bcc_get_double() 219
Funzione bcc_get_float() 220
Funzione bcc_get_i16() 220
Funzione bcc_get_i32() 221
Funzione bcc_get_i8() 221
Funzione bcc_get_string() 222
Funzione bcc_get_tcp_clients_info() 222
Funzione bcc_get_u16() 223
Funzione bcc_get_u32() 224
Funzione bcc_get_u8() 224
Funzione bcc_h_setup() 225
Funzione bcc_receive() 226
Funzione bcc_reply_ack() 227
Funzione bcc_reply_msg() 228
Funzione bcc_reply_nack() 229
Funzione bcc_send() 229
Funzione check_mv_follow2() 230
Funzione check_mva_follow2() 233
Funzione check_mva_follow2_m() 236
Funzione co_asw_sdo() 239
Funzione co_controlword_om() 241
Funzione co_controlword_state() 243

© 2025 Robox SpA

Funzione co_get_ws_diagno() 245
Funzione co_par_download() 248
Funzione co_par_upload() 251
Funzione co_rerr() 255
Funzione co_rnmt() 256
Funzione co_robj() 258
Funzione co_rpdo() 265
Funzione co_send_sdo() 266
Funzione co_statusword() 270
Funzione co_sync() 272
Funzione co_wnmt() 272
Funzione co_wobj() 273
Funzione co_wpdo() 278
Funzione coe_robj() 279
Funzione coe_wobj() 284
Funzione command() 288
Funzione cond_val() 289
Funzione cos() 290
Funzione crc16_be() 290
Funzione crc16_le() 292
Funzione ct_cv() 294
Funzione ct_pv() 294
Funzione ct_qd() 295
Funzione ct_qu() 295
Funzione dec() 295
Funzione delta_enc16() 296
Funzione delta_enc32() 297
Funzione diff() 299
Funzione dw0() 300
Funzione dw1() 301
Funzione dwbe2le() 302
Funzione dwbe2lef() 302
Funzione dwbe2mbe() 303
Funzione dwbe2mle() 303
Funzione dwle2be() 304
Funzione dwle2bef() 304
Funzione dwle2mbe() 305
Funzione dwle2mbef() 305
Funzione dwle2mle() 306
Funzione dwle2mlef() 306
Funzione dwmbe2be() 307
Funzione dwmbe2le() 307
Funzione dwmbe2lef() 308
Funzione dwmle2be() 308
Funzione ecat_dev_on_off() 309
Funzione ecat_get_state() 312
Funzione ecat_probe() 313
Funzione ecat_set_state() 314
Funzione excl_lam() 316
Funzione excl_pam() 317
Funzione f_avail() 318
Funzione f_close() 318
Funzione f_copy() 319
Funzione f_del() 320

© 2025 Robox SpA

Funzione f_dir() 321
Funzione f_format() 323
Funzione f_gets() 324
Funzione f_mkdir() 326
Funzione f_open() 327
Funzione f_puts() 329
Funzione f_read() 330
Funzione f_read_table() 331
Funzione f_read_table_s() 333
Funzione f_rename() 335
Funzione f_seek() 336
Funzione f_write() 337
Funzione fall() 338
Funzione fall_filt() 339
Funzione filter_lp() 341
Funzione filter_lp_st() 342
Funzione filter_notch() 343
Funzione filter_notch_st() 345
Funzione first_time() 346
Funzione force() 347
Funzione fract() 350
Funzione get_float() 350
Funzione get_hw_info() 351
Funzione get_i16() 353
Funzione get_i32() 354
Funzione get_i8() 355
Funzione get_real() 355
Funzione get_u16() 356
Funzione get_u32() 357
Funzione get_u8() 358
Funzione get_virt_ow() 359
Funzione gv_create 359
Funzione gv_datatype 362
Funzione gv_datatype_byname 362
Funzione gv_destroy 363
Funzione gv_exist 364
Funzione gv_flags 364
Funzione gv_flags_byname 366
Funzione gv_link 368
Funzione gv_lock 369
Funzione gv_name 370
Funzione gv_read 370
Funzione gv_read_partial 371
Funzione gv_size 372
Funzione gv_size_byname 372
Funzione gv_unlink 373
Funzione gv_unlock 373
Funzione gv_update_file 374
Funzione gv_write 374
Funzione gv_write_partial 375
Funzione in_range() 376
Funzione inc() 377
Funzione index_of() 377
Funzione ipaddr_to_str() 378

© 2025 Robox SpA

Funzione is_alarm_par_present() 379
Funzione is_alarm_par_present_us() 381
Funzione is_alarm_present() 383
Funzione is_forced() 384
Funzione kin_conv() 386
Funzione log() 388
Funzione log10() 388
Funzione loop_time() 389
Funzione master_filter() 389
Funzione maximum() 390
Funzione mbc_connect() 391
Funzione mbc_init() 391
Funzione mbc_mask_write_reg() 394
Funzione mbc_mask_write_reg_s() 396
Funzione mbc_read_coils() 398
Funzione mbc_read_coils_s() 399
Funzione mbc_read_discr_inp() 400
Funzione mbc_read_discr_inp_s() 402
Funzione mbc_read_hold_reg() 403
Funzione mbc_read_hold_reg_s() 405
Funzione mbc_read_inp_reg() 406
Funzione mbc_read_inp_reg_s() 408
Funzione mbc_shutdown() 410
Funzione mbc_write_coils() 410
Funzione mbc_write_coils_s() 412
Funzione mbc_write_multiple_coils() 413
Funzione mbc_write_multiple_coils_s() 414
Funzione mbc_write_multiple_regs() 416
Funzione mbc_write_multiple_regs_s() 418
Funzione mbc_write_reg() 419
Funzione mbc_write_reg_s() 421
Funzione mbcu_mask_write_reg() 422
Funzione mbcu_read_coils() 424
Funzione mbcu_read_discr_inp() 426
Funzione mbcu_read_hold_reg() 427
Funzione mbcu_read_inp_reg() 429
Funzione mbcu_write_coils() 431
Funzione mbcu_write_multiple_coils() 433
Funzione mbcu_write_multiple_regs() 434
Funzione mbcu_write_reg() 436
Funzione minimum() 438
Funzione mod() 439
Funzione mot_append() 439
Funzione mot_clear() 443
Funzione mot_create() 443
Funzione mot_delete() 447
Funzione mot_get() 447
Funzione mot_remove() 449
Funzione mt_ats() 449
Funzione mt_ats2() 450
Funzione mv_cam() 451
Funzione mv_crimper() 453
Funzione mv_follow() 454
Funzione mv_follow2() 458

© 2025 Robox SpA

Funzione mva_follow2() 467
Funzione mva_follow2_m() 469
Funzione mv_mot_exec() 470
Funzione mv_phase_adj() 472
Funzione mv_phase_adj2() 474
Funzione mv_ramp() 477
Funzione mv_reach_target() 477
Funzione mv_sine_cam() 478
Funzione mv_synchro() 479
Funzione mv_table() 481
Funzione mv_to() 482
Funzione mv_to(), parametrizzata in linea 484
Funzione mv_to_cj() 486
Funzione mv_to_cjv() 487
Funzione mv_to_cjv_info() 489
Funzione mv_to_vel() 489
Funzione mv_tracking() 491
Funzione mva_jog() 491
Funzione mva_jog2() 493
Funzione mva_open_loop() 496
Funzione mva_to_n() 496
Funzione mva_to_n_cj() 500
Funzione mva_to_n_v() 504
Funzione mva_zc() 511
Funzione normalize() 513
Funzione onoff_filt() 514
Funzione par_even() 516
Funzione par_odd() 517
Funzione planar_comp_cp() 518
Funzione planar_comp_ip() 519
Funzione pos_capture_cmd() 520
Funzione pos_capture_sts() 524
Funzione pow() 525
Funzione ps_channel_enable() 526
Funzione ps_channel_enabled() 529
Funzione ps_channel_fault() 531
Funzione ps_channel_reset_fault() 532
Funzione ps_drive_reset() 535
Funzione ps_exclude_lax() 538
Funzione ps_exclude_pax() 539
Funzione ps_power_enable() 540
Funzione ps_reinclude_lax() 541
Funzione ps_reinclude_pax() 542
Funzione ps_status() 543
Funzione qwbe2lef() 547
Funzione qwbedwx2lef() 547
Funzione qwmbe2lef() 548
Funzione qwmbedwx2lef() 548
Funzione r_rol() 549
Funzione r_ror() 550
Funzione r_shl() 551
Funzione r_shr() 552
Funzione r2m16() 553
Funzione r2m32() 554

© 2025 Robox SpA

Funzione r2m8() 554
Funzione ramp() 555
Funzione rand32() 556
Funzione random() 557
Funzione random_value() 557
Funzione range() 557
Funzione range_adj() 558
Funzione rbit() 559
Funzione release() 560
Funzione rise() 562
Funzione rise_filt() 563
Funzione round() 565
Funzione rtc_date() 565
Funzione rtc_time() 566
Funzione safe_appl_ack() 567
Funzione safe_appl_diag() 568
Funzione safe_diag() 569
Funzione safe_diag_reset() 572
Funzione safe_enab_out() 573
Funzione safe_oper_ack() 573
Funzione safe_prj_info() 574
Funzione safe_sts() 575
Funzione sb0() 577
Funzione sb1() 578
Funzione sb2() 578
Funzione sb3() 579
Funzione sb4() 580
Funzione sb5() 580
Funzione sb6() 581
Funzione sb7() 582
Funzione sbit() 582
Funzione sdw0() 583
Funzione sdw1() 584
Funzione ser_485_tx_sts() 585
Funzione ser_bcc_h_init() 587
Funzione ser_bin_in() 588
Funzione ser_bin_out() 588
Funzione ser_bin_ready() 589
Funzione ser_buff_free() 590
Funzione ser_buff_queue() 590
Funzione ser_flush() 590
Funzione ser_in() 591
Funzione ser_mod() 592
Funzione ser_out() 593
Funzione ser_r_sts() 593
Funzione ser_ready() 594
Funzione set_kinematics() 595
Funzione set_virt_iw() 595
Funzione shift_left() 596
Funzione shift_right() 597
Funzione shl() 599
Funzione shr() 599
Funzione sign() 600
Funzione similar() 601

© 2025 Robox SpA

Funzione sin() ... 602
Funzione sizeof() ... 602
Funzione soe_ridn() ... 603
Funzione soe_widn() ... 615
Funzione sqrt() ... 620
Funzione statistics() ... 621
Funzione store_nvr_nopf() ... 622
Funzione store_nvrr_nopf() ... 623
Funzione str_asc() ... 624
Funzione str_chr() ... 625
Funzione str_compact() ... 625
Funzione str_equal() ... 626
Funzione str_find() ... 627
Funzione str_format() ... 627
Funzione str_i() ... 629
Funzione str_insert() ... 630
Funzione str_len() ... 631
Funzione str_mid() ... 631
Funzione str_r() ... 632
Funzione str_replace() ... 633
Funzione str_to_ipaddr() ... 634
Funzione str_val() ... 635
Funzione sum8_mem() ... 637
Funzione sum16_mem() ... 638
Funzione sum32_mem() ... 638
Funzione sumr_mem() ... 638
Funzione sw0() ... 639
Funzione sw1() ... 639
Funzione sw2() ... 640
Funzione sw3() ... 641
Funzione tan() ... 641
Funzione time_micro() ... 642
Funzione tm_et() ... 642
Funzione tm_pt() ... 643
Funzione tm_pt2() ... 644
Funzione tm_q() ... 645
Funzione tobcd() ... 645
Funzione tobin() ... 646
Funzione todeg() ... 646
Funzione tolower() ... 647
Funzione torad() ... 647
Funzione toupper() ... 648
Funzione tracking() ... 648
Funzione trunc() ... 650
Funzione udp_close() ... 651
Funzione udp_open_client() ... 651
Funzione udp_open_server() ... 654
Funzione udp_recv() ... 657
Funzione udp_recv_from() ... 658
Funzione udp_recv_notify() ... 660
Funzione udp_send() ... 662
Funzione udp_send_notify() ... 665
Funzione udp_send_to() ... 667
Funzione virt_io() ... 669

© 2025 Robox SpA

Funzione w0() ... 671
Funzione w1() ... 672
Funzione w2() ... 672
Funzione w3() ... 673
Funzione wbe2le() ... 674
Funzione wbit() ... 674
Funzione wle2be() ... 675
Funzione write_dis() ... 676
9 Istruzioni ... 677
Istruzione __main_loop__ ... 677
Istruzione _else ... 678
Istruzione _if ... 678
Istruzione aux ... 679
Istruzione axes ... 680
Istruzione axes_m ... 680
Istruzione b0 ... 681
Istruzione b1 ... 681
Istruzione b2 ... 681
Istruzione b3 ... 682
Istruzione b4 ... 682
Istruzione b5 ... 682
Istruzione b6 ... 683
Istruzione b7 ... 683
Istruzione bcc_set_double ... 683
Istruzione bcc_set_float ... 684
Istruzione bcc_set_i16 ... 684
Istruzione bcc_set_i32 ... 685
Istruzione bcc_set_i8 ... 685
Istruzione bcc_set_string ... 686
Istruzione bcc_set_u16 ... 686
Istruzione bcc_set_u32 ... 687
Istruzione bcc_set_u8 ... 688
Istruzione bit_mask_add ... 688
Istruzione bit_mask_remove ... 690
Istruzione bit_mask_set ... 692
Istruzione break ... 694
Istruzione call ... 694
Istruzione case ... 696
Istruzione continue ... 697
Istruzione ct_cd ... 697
Istruzione ct_cdr ... 698
Istruzione ct_cu ... 699
Istruzione ct_cur ... 700
Istruzione ct_cv ... 701
Istruzione ct_load ... 701
Istruzione ct_pv ... 701
Istruzione ct_reset ... 702
Istruzione default ... 702
Istruzione do ... 703
Istruzione dw0 ... 703
Istruzione dw1 ... 704
Istruzione dwell ... 704
Istruzione else ... 704
Istruzione end_aux ... 705

© 2025 Robox SpA

Istruzione end_do_while 706
Istruzione end_for 706
Istruzione end_fun 706
Istruzione end_main_loop 707
Istruzione end_motion 707
Istruzione end_ref 708
Istruzione end_rule 708
Istruzione end_select 709
Istruzione end_while 710
Istruzione endif 710
Istruzione excl_lam 711
Istruzione excl_pam 712
Istruzione fill_mem 712
Istruzione for 713
Istruzione function 715
Istruzione get_i64 719
Istruzione get_u64 720
Istruzione goto 721
Istruzione group 721
Istruzione if 723
Istruzione inf_report 724
Istruzione int_after 725
Istruzione int_inp 726
Istruzione int_timer 728
Istruzione lit 729
Istruzione motion 730
Istruzione move_byte 731
Istruzione move_mem 732
Istruzione mt_dis 733
Istruzione mt_en 734
Istruzione mt_rr_flag 734
Istruzione mt_wait 735
Istruzione order 735
Istruzione out_m 737
Istruzione qwbe2le 738
Istruzione qwbedwx2le 739
Istruzione qwle2be 739
Istruzione qwle2bedwx 740
Istruzione qwle2bedwxf 740
Istruzione qwle2bef 741
Istruzione qwle2mbe 741
Istruzione qwle2mbedwx 742
Istruzione qwle2mbedwxf 742
Istruzione qwle2mbef 743
Istruzione qwmbe2le 743
Istruzione qwmbedwx2le 744
Istruzione ref 744
Istruzione ref_prologue 745
Istruzione restart_point 746
Istruzione return 746
Istruzione rule 748
Istruzione rule_epilogue 756
Istruzione rule_freq 756
Istruzione rule_init 757

© 2025 Robox SpA

Istruzione rule_periodic ... 758
Istruzione rule_prologue ... 759
Istruzione rule_start_norc ... 760
Istruzione sb0 ... 760
Istruzione sb1 ... 761
Istruzione sb2 ... 761
Istruzione sb3 ... 761
Istruzione sb4 ... 762
Istruzione sb5 ... 762
Istruzione sb6 ... 762
Istruzione sb7 ... 763
Istruzione sdw0 ... 763
Istruzione sdw1 ... 763
Istruzione select ... 764
Istruzione ser_con ... 765
Istruzione ser_w_sts ... 766
Istruzione set_float ... 767
Istruzione set_i16 ... 768
Istruzione set_i32 ... 769
Istruzione set_i64 ... 770
Istruzione set_i8 ... 771
Istruzione set_real ... 772
Istruzione set_u16 ... 772
Istruzione set_u32 ... 773
Istruzione set_u64 ... 774
Istruzione set_u8 ... 775
Istruzione set_version ... 776
Istruzione sw0 ... 777
Istruzione sw1 ... 777
Istruzione sw2 ... 778
Istruzione sw3 ... 778
Istruzione terminator ... 778
Istruzione tm_et ... 779
Istruzione tm_int ... 779
Istruzione tm_of ... 780
Istruzione tm_on ... 782
Istruzione tm_pt ... 784
Istruzione tm_pt2 ... 784
Istruzione tm_reset ... 784
Istruzione tm_sqw ... 784
Istruzione tm_tp ... 786
Istruzione w0 ... 787
Istruzione w1 ... 787
Istruzione w2 ... 788
Istruzione w3 ... 788
Istruzione while ... 789

Parte IV Grammatica 790

1 Generalità grammatica ... 790

Parte V Introduzione 791

1 Generalità linguaggio R3 ... 791

© 2025 Robox SpA

Parte VI Metacomandi 798

1 Metacomandi ... 798
2 Metacomando $define ... 800
3 Metacomando $ifdef/$ifndef ... 801
4 Metacomando $include ... 802
5 Metacomando $undef ... 802

Parte VII Operatori 803

1 Operatori ... 803
2 Operatori aritmetici ... 804
3 Operatori a bit (binari) ... 806
4 Operatori di assegnazione ... 808
5 Operatori di concatenamento stringhe ... 808
6 Operatori logici ... 809
7 Operatori relazionali numerici ... 811

Parte VIII Tipi di dato 814

1 Struttura BCC_MSGUSER ... 820
2 Struttura STRU_AAX ... 822
3 Struttura STRU_AH_D ... 823
4 Struttura STRU_AH_D_EXT ... 824
5 Struttura STRU_AH_INFO ... 825
6 Struttura STRU_AS_D ... 826
7 Struttura STRU_AS_D_EXT ... 827
8 Struttura STRU_AXIO_PDI ... 828
9 Struttura STRU_BCC_TCP_CLIENT ... 830
10 Struttura STRU_CAM ... 831
11 Struttura STRU_CO_OBJ ... 832
12 Struttura STRU_COE_OBJ ... 834
13 Struttura STRU_COERRMSG ... 836
14 Struttura STRU_CRIMPER ... 839
15 Struttura STRU_ECAT_PROBE ... 841
16 Struttura STRU_FILEHND ... 842
17 Struttura STRU_FILEINFO ... 843
18 Struttura STRU_FOLLOW ... 844
19 Struttura STRU_FOLLOW2 ... 846
20 Struttura STRU_FOLLOW2_INFO ... 854
21 Struttura STRU_FOLLOW2_S ... 855
22 Struttura STRU_FOLLOW2_SM ... 862

© 2025 Robox SpA

23 Struttura STRU_GROR ... 869
24 Struttura STRU_HW_INFO ... 870
25 Struttura STRU_INT ... 870
26 Struttura STRU_LARGE_INT ... 872
27 Struttura STRU_LARGE_UINT ... 873
28 Struttura STRU_MASTERF ... 874
29 Struttura STRU_MATCAM ... 875
30 Struttura STRU_MODBUS ... 876
31 Struttura STRU_MVGENITP ... 877
32 Struttura STRU_MVITP ... 879
33 Struttura STRU_MVITPCJ ... 880
34 Struttura STRU_MVTO ... 881
35 Struttura STRU_MVTOCJ ... 884
36 Struttura STRU_MVTOCJV ... 886
37 Struttura STRU_MVTOCJV_INFO ... 888
38 Struttura STRU_MVTOCJVE ... 890
39 Struttura STRU_MVZC ... 892
40 Struttura STRU_PHADJ ... 898
41 Struttura STRU_PHADJ2 ... 901
42 Struttura STRU_PM_PD ... 904
43 Struttura STRU_REACHTARGET ... 906
44 Struttura STRU_SAFETY_PRJINFO ... 908
45 Struttura STRU_SOE_IDN ... 909
46 Struttura STRU_STAT ... 912
47 Struttura STRU_SYNCHRO ... 913
48 Struttura STRU_TRACK ... 916
49 Struttura STRUCT ... 917
50 Struttura STRUCT_P ... 918
51 Struttura TABSTR ... 919
52 Tipo BOOL ... 919
53 Tipo CAST ... 920
54 Tipo CHAR ... 921
55 Tipo COUNTER ... 921
56 Tipo FLOAT ... 922
57 Tipo I16 ... 923
58 Tipo I32 ... 924
59 Tipo I8 ... 925
60 Tipo POWER_SET ... 926
61 Tipo REAL ... 928

© 2025 Robox SpA

62 Tipo STRING 929
63 Tipo TIMER 930
64 Tipo U16 930
65 Tipo U32 931
66 Tipo U8 932

Parte IX Varie 933

1 Modifiche LIVE 933

Indice 937

© 2025 Robox SpA

Indice generale

Principali argomenti:

- Introduzione 791
- Costanti 17
- Grammatica 790
- Metacomandi 798
- Operatori 803
- Tipi di dato 814
- Funzioni/Istruzioni 19

Altri argomenti utili:

- Modifiche "LIVE" 933

Collegamenti esterni:

- Firmware RTE (Real Time Extended)
- Variabili globali (RTE)
- Variabili predefinite (RTE)
- Linguaggio di programmazione LADDER IEC1131
- Libreria Object Block (Interfaccia verso OS)
- Libreria RPE (Robox Path Executor)

Costanti

Costanti

La costante è un valore numerico che non cambia durante l'esecuzione del programma. Nella scrittura delle istruzioni R3 è ammesso l'uso di costanti.

Esistono alcune costanti proprie del linguaggio e non modificabili da parte dell'utente.

Costanti matematiche predefinite

Keyword	Valore Numerico	Significato
K_E	2.718281828459045	Costante di Eulero
K_PI	3.141592653589793	Pigreco
K_PIMEZZI	1.570796326794896	Pigreco mezzi
K_PI_2	1.570796326794896	Pigreco mezzi
K_1_PI	0.318309886183790	Uno diviso pigreco
K_DUEPI	6.283185307179586	Due pigreco
K_2PI	6.283185307179586	Due pigreco
K_EPSILON	1e-6	Soglia di decisione per confronti fra reali
K_MAXFLOAT	3.37e+38	Costante piu grande rappresentabile in un float
K_MINFLOAT	8.43e-37	Costante piu piccola rappresentabile in un float
K_MAXREAL	1.797693e+308	Costante piu grande rappresentabile in un double

© 2025 Robox SpA

Keyword	Valore Numerico	Significato
K_MINREAL	2.225074e-308	Costante piu piccola rappresentabile in un double

Costanti definibili dall'utente

I tipi di costanti che il programmatore può definire sono:

- Costanti intere 18
- Costanti reali 18
- Costanti stringa 19

Costanti intere

Possono essere qualsiasi numero intero compreso fra -2147483648 e +2147483647.

Le costanti intere possono essere rappresentate in base decimale o esadecimale.

Per identificare un numero esadecimale sarà necessario aggiungere la lettera "**H**" alla fine del numero, oppure far precedere il numero dal prefisso "**0x**".

I numeri esadecimali devono sempre iniziare per cifra numerica (anteporre lo 0 se necessario). I numeri esadecimali sono rappresentati in complemento a due su 8 digit e il segno meno non è pertanto ammesso.

Esempi

Esempio costanti decimali:

```
; Costanti decimali
10
-20
```

Esempio costanti esadecimali:

```
0XA          ; Valore decimale 10
0AH          ; Valore decimale 10
10H          ; Valore decimale 16
0FFH         ; Valore decimale 255
FFFFFFFFH    ; Valore decimale -1
0XFFFFFFFF   ; Valore decimale -1
```

Costanti reali

Possono essere un qualsiasi numero reale compreso fra -1.79 E 308 e 1.79 E 308.

L'entità più piccola che si può rappresentare è, in valore assoluto 2.22 E -308.

Le cifre significative sono almeno 15.

La sintassi della rappresentazione di un numero reale è la seguente: *si.d*

Parametro	Significato
s	segno del numero (opzionale)
i	parte intera del numero
.	separatore delle cifre decimali (opzionale)

© 2025 Robox SpA

Parametro	Significato
d	parte decimale del numero (opzionale)

La sintassi della rappresentazione esponenziale è la seguente: *si.dEse*

Parametro	Significato
s	segno del numero (opzionale)
i	parte intera della mantissa
.	separatore delle cifre decimali (opzionale)
d	parte decimale della mantissa (opzionale)
E	identificatore del formato esponenziale (in base 10)
s	segno dell' esponente (opzionale)
e	esponente

Esempi

Esempio costanti reali:

```
10
10.51
1452.7
1.4527E3
-12.4567E-4
```

Costanti stringa

Le costanti stringa possono essere una qualsiasi sequenza di caratteri compresa tra doppi apici " ".

NOTA: Per usare il carattere doppi apici (") all'interno di una costante stringa la sintassi da usare è \" . La lunghezza massima è di 127 caratteri netti.

Esempio di utilizzo:

```
; Esempio di costante stringa
"questa è una stringa"
"il carattere doppiapici \" e' contenuto in questa stringa "
```

Funzioni/istruzioni

Elenco delle principali funzioni e istruzioni suddivise per categoria:

- Funzioni/istruzioni matematiche 109
- Funzioni/istruzioni safety 111
- Funzioni/istruzioni supporto rule/task 112

Lista completa di tutte le funzioni e istruzioni:

- Indice funzioni/istruzioni 20

Altro:

- Codici di errore funzioni 59
- Codici di reso dalle funzioni mv_ e mva_ 63

© 2025 Robox SpA

- Gestione allarmi 114
- Gestione Axioline 116
- Gestione canali seriali 116
- Gestione CANopen/EtherCAT 118
- Gestione CNET 120
- Gestione comunicazione BCC 121
- Gestione controllo di flusso 123
- Gestione conversione endianess 125
- Gestione counter 130
- Gestione dati 131
- Gestione flash 136
- Gestione forzatura I/O 138
- Gestione modbus/TCP client 138
- Gestione MOT 142
- Gestione movimento 142
- Gestione power set 145
- Gestione stringhe 146
- Gestione timer 148
- Gestione UDP/IP 148
- Gestione variabili globali 149
- Gestioni miscellanee 152
- Codici di errore SDO (CoE/SoE) 65
- Codici di errore Modbus 68
- Tipi di ciclo di zero 69

Indice funzioni/istruzioni

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)

Keyword	R	T	F/I	Descrizione
_else 678	R	T	I	Istruzione controllo di flusso ALTRIMENTI
_if 678	R	T	I	Istruzione controllo di flusso SE
abs 156()	R	T	F	Valore assoluto di una grandezza
abs_mod 157()	R	T	F	Resto della divisione

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
acos 157()	R	T	F	Arcocoseno
ah_cmd 158()	R	T	F	Comando per storico allarmi
ah_get_d 159()	R	T	F	Comando per storico allarmi
ah_get_s 160()	R	T	F	Comando per storico allarmi
ah_info 162()	R	T	F	Comando per storico allarmi
ah_log 163()	R	T	F	Comando per storico allarmi
ah_log_us 164()	R	T	F	Comando per storico allarmi
al2am 166()	R	T	F	Restituisce informazioni sull'AM in funzione di un allarme
alarm_reset 169()	R	T	F	Reset di un allarme
alarm_reset_par 170()	R	T	F	Reset di un allarme specificando anche il parametro
alarm_reset_par_us 172()	R	T	F	Reset di un allarme specificando anche il parametro USV
alarm_set 175()	R	T	F	Set di un allarme
alarm_set_us 176()	R	T	F	Set di un allarme con variabile stringa utente USV
and 809	R	T	I	Operatore logico AND
as_get_d 178()	R	T	F	Comando per storico allarmi
asin 178()	R	T	F	Arcoseno
atan 179()	R	T	F	Arcotangente di un valore
atan2 180()	R	T	F	Arcotangente di un rapporto

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
aux 679	R	-	I	Inizio Blocco AUX nella RULE
axes 680	R	-	I	Definizione degli assi coinvolti nella RULE (costante)
axes_m 680	R	-	I	Definizione degli assi coinvolti nella RULE (maschera)
axio_iol_r 181()	-	T	F	Funzione per leggere un'area dati da IOlink master noto
axio_iol_w 182()	-	T	F	Funzione per scrivere un'area dati su IOlink master noto
axio_pm_r 183()	-	T	F	Funzione per leggere un'area dati da modulo AXL_F_PM_EF_1F
axio_robj 186()	-	T	F	Funzione per leggere gli oggetti su fieldbus axioline
axio_wobj 199()	-	T	F	Funzione per scrivere gli oggetti su fieldbus axioline
b0 213() / b0 681()	R	T	F/I	Legge/scrive il byte 0 dell'argomento
b1 215() / b1 681()	R	T	F/I	Legge/scrive il byte 1 dell'argomento
b2 215() / b2 681()	R	T	F/I	Legge/scrive il byte 2 dell'argomento
b3 216() / b3 682()	R	T	F/I	Legge/scrive il byte 3 dell'argomento
b4 216() / b4 682()	R	T	F/I	Legge/scrive il byte 4 dell'argomento
b5 217() / b5 682()	R	T	F/I	Legge/scrive il byte 5

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				dell'argomento
b6 218() / b6 683()	R	T	F/I	Legge/scrive il byte 6 dell'argomento
b7 218() / b7 683()	R	T	F/I	Legge/scrive il byte 7 dell'argomento
bcc_get_double 219()	R	T	F	Legge un double da una struttura BCC_MSGUSER 820
bcc_get_float 220()	R	T	F	Legge un float da una struttura BCC_MSGUSER 820
bcc_get_i16 220()	R	T	F	Legge un i16 da una struttura BCC_MSGUSER 820
bcc_get_i32 221()	R	T	F	Legge un i32 da una struttura BCC_MSGUSER 820
bcc_get_i8 221()	R	T	F	Legge un i8 da una struttura BCC_MSGUSER 820
bcc_get_string 222()	R	T	F	Legge una stringa da una struttura BCC_MSGUSER 820
bcc_get_tcp_clients_info 222()	R	T	F	Legge un i32 da una struttura STRU_BCC_TCP_CLIENT 830
bcc_get_u16 223()	R	T	F	Legge un u16 da una struttura BCC_MSGUSER 820
bcc_get_u32 224()	R	T	F	Legge un u32 da una struttura BCC_MSGUSER 820
bcc_get_u8 224()	R	T	F	Legge un u8 da una struttura BCC_MSGUSER 820
bcc_h_setup 225()	R	T	F	Prepara la struttura per la risposta

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
bcc_receive 226()	R	T	F	Riceve un pacchetto BCC3
bcc_reply_ack 227()	R	T	F	Invia risposta positiva a messaggio ricevuto
bcc_reply_msg 228()	R	T	F	Invia risposta a messaggio ricevuto
bcc_reply_nack 229()	R	T	F	Invia risposta negativa a messaggio ricevuto
bcc_send 229()	R	T	F	Invia un pacchetto BCC3
bcc_set_double 683	R	T	I	Scrive un double in una struttura BCC_MSGUSER 820
bcc_set_float 684	R	T	I	Scrive un float in una struttura BCC_MSGUSER 820
bcc_set_i16 684	R	T	I	Scrive un i16 in una struttura BCC_MSGUSER 820
bcc_set_i32 685	R	T	I	Scrive un i32 in una struttura BCC_MSGUSER 820
bcc_set_i8 685	R	T	I	Scrive un i8 in una struttura BCC_MSGUSER 820
bcc_set_string 686	R	T	I	Scrive una string in una struttura BCC_MSGUSER 820
bcc_set_u16 686	R	T	I	Scrive un u16 in una struttura BCC_MSGUSER 820
bcc_set_u32 687	R	T	I	Scrive un u32 in una struttura BCC_MSGUSER 820
bcc_set_u8 688	R	T	I	Scrive un u8 in una struttura BCC_MSGUSER 820

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
bit_mask_add 688	-	-	I	Somma un valore esadecimale a una variabile di tipo INT
bit_mask_remove 690	-	-	I	Sottrae un valore esadecimale a una variabile di tipo INT
bit_mask_set 692	-	-	I	Inizializza una variabile di tipo INT a un valore esadecimale
break 694	R	T	I	Uscita da un blocco di selezione
call 694	R	T	I	Chiamata di un sottoprogramma
case 696	R	T	I	Costrutto SELECT
check_mv_follow2 230()	R	T	F	Controlla se la funzione mv_follow2 458() può essere eseguita
check_mva_follow2 233()	R	T	F	Controlla se la funzione mva_follow2 458() può essere eseguita
check_mva_follow2_m 236()	R	T	F	Controlla se la funzione mva_follow2_m 458() può essere eseguita
cnet_diag()	R	T	F	Funzioni comunicazione sincrona CNET. Riservata
cnet_get_time_sync()	R	T	F	Funzioni comunicazione sincrona CNET. Riservata
cnet_receive()	R	T	F	Funzioni comunicazione sincrona CNET. Riservata
cnet_receive_status()	R	T	F	Funzioni comunicazione

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				sincrona CNET. Riservata
cnet_rq_sync()	R	T	F	Funzioni comunicazione sincrona CNET. Riservata
cnet_send()	R	T	F	Funzioni comunicazione sincrona CNET. Riservata
cnet_status()	R	T	F	Funzioni comunicazione sincrona CNET. Riservata
co_statusword 270 ()	R	T	F	Restituisce il contenuto della status word della workstation indicata
co_asw_sdo 239()	R	T	F	Restituisce la risposta alla precedente funzione co_send_sdo 266()
co_controlword_state 243()	R	T	F	Funzione per effettuare transizioni nella macchina a stati DS 402
co_controlword_om 241()	R	T	F	Funzione per scrivere alcuni bit della controlword
co_get_ws_diagno 245()	R	T	F	Funzione per leggere informazioni di stato della workstation indicata
co_par_download 248()	-	T	F	Funzione per caricare sul drive i parametri leggendoli da CF del controllo
co_par_upload 251 ()	-	T	F	Funzione per salvare su un file

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				in CF i parametri leggendoli da drive
co_rerr 255()	R	T	F	Restituisce informazioni di errore della workstation indicata
co_rnmt 256()	R	T	F	Funzione che rende il modo operativo NMT della WS indicata
co_robj 258()	R	T	F	Funzione per leggere gli oggetti su fieldbus
co_rpdo 265()	R	T	F	Legge un PDO
co_send_sdo 266()	R	T	F	Invia un messaggio SDO (Service Data Object)
co_sync 272()	R	T	F	Abilita/disabilita messaggi di Sync su Canbus
co_wnmt 272()	R	T	F	Impone lo stato NMT della workstation indicata
co_wobj 273()	R	T	F	Funzione per scrivere gli oggetti su fieldbus
co_wpdo 278()	R	T	F	Invia un PDO
coe_robj 279()	R	T	F	Funzione per leggere gli oggetti su fieldbus COE
coe_wobj 284()	R	T	F	Funzione per scrivere gli oggetti su fieldbus COE
command 288()	R	T	F	Esegue un comando (direttiva)
cond_val 289	R	T	I	Costrutto IF-THEN-ELSE semplificato

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
continue 697	R	T	I	Costrutto FOR-WHILE
cos 290()	R	T	F	Coseno
crc16()	R	T	F	Deprecata. Utilizzare crc16_be 290()
crc16_ero()	R	T	F	Deprecata. Utilizzare crc16_le 292()
crc16_init()	R	T	F	Deprecata. Utilizzare crc16_be 290() / crc16_le 292()
crc16_be 290()	R	T	F	Funzioni per il calcolo del CRC
crc16_le 292()	R	T	F	Funzioni per il calcolo del CRC
ct_cd 697()	R	T	I	Counter down (decrementale)
ct_cdr 698()	R	T	I	Counter down ring
ct_cu 699()	R	T	I	Counter up (incrementale)
ct_cur 700()	R	T	I	Counter up ring
ct_cv 294() / ct_cv 701()	R	T	F/I	Valore corrente del counter
ct_load 701()	R	T	I	Carica il CP (valore di preset) nel CV (valore corrente) del counter
ct_pv 294() / ct_pv 701()	R	T	F/I	Valore di preset del counter
ct_qd 295()	R	T	F	Esito del counter down
ct_qu 295()	R	T	F	Esito del counter up
ct_reset 702()	R	T	I	Reset del CV del counter
dec 295()	R	T	F	Decrementa la variabile

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
default 702	R	T	I	Costrutto SELECT
delta_enc16 296()	R	T	F	Calcola la differenza di due grandezze a 16 bit gestendo lo scaravolto
delta_enc32 297()	R	T	F	Calcola la differenza di due grandezze a 32 bit gestendo lo scaravolto
diff 299()	R	T	F	Rende vero se il valore della variabile è cambiato
do 703	R	T	I	Ciclo DO
dw0 300() / dw0 703 ()	R	T	F/I	Legge/scrive la Dword 0 dell'argomento
dw1 301() / dw1 704 ()	R	T	F/I	Legge/scrive la Dword 1 dell'argomento
dwbe2le 302()	R	T	F	Converte big endian dword (x32) in little endian dword (x32)
dwbe2lef 302()	R	T	F	Converte big endian dword (x32) in little endian float (x32)
dwbe2mbe 303()	R	T	F	Converte big endian dword (x32) in middle big endian dword (x32)
dwbe2mle 303()	R	T	F	Converte big endian dword (x32) in middle little endian dword (x32)
dwell 704	-	T	I	Arresta il TASK per n secondi

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
dwle2be 304()	R	T	F	Converte little endian dword (x32) in big endian wdord (x32)
dwle2bef 304()	R	T	F	Converte little endian float (4 bytes) in big endian dword (U32)
dwle2mbe 305()	R	T	F	Converte little endian dword (x32) in middle big endian dword (x32)
dwle2mbef 305()	R	T	F	Converte little endian float (4 bytes) in middle big endian dword (U32)
dwle2mle 306()	R	T	F	Converte little endian dword (x32) in middle little endian dword (x32)
dwle2mlef 306()	R	T	F	Converte little endian float (4 bytes) in middle little endian dword (U32)
dwmbe2be 307()	R	T	F	Converte middle big endian dword (x32) in big endian dword (x32)
dwmbe2le 307()	R	T	F	Converte middle big endian dword (x32) in little endian dword (x32)
dwmbe2lef 308()	R	T	F	Converte middle big endian (U32) in little endian float (4 bytes)
dwmle2be 308()	R	T	F	Converte middle little endian dword (x32) in big endian dword (x32)

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
ecat_dev_on_off 309()	-	T	F	Abilita/disabilita un dispositivo su linea EtherCAT
ecat_get_state 312()	R	T	F	Legge lo stato della linea EtherCAT
ecat_probe 313()	-	T	F	Legge le informazioni di un dispositivo su linea EtherCAT
ecat_set_state 314()	R	T	F	Impone lo stato della linea EtherCAT
else 704	R	T	I	Scelta alternativa blocco IF
end_aux 705	R	-	I	Fine blocco AUX
end_for 706	R	T	I	Fine ciclo FOR
end_fun 706	R	T	I	Fine funzione
end_motion 707	R	-	I	Fine blocco MOTION
end_ref 708	R	-	I	Fine blocco REF
end_rule 708	R	-	I	Fine rule
end_select 709	R	T	I	Fine costrutto SELECT
end_while 710	R	T	I	Fine costrutto WHILE
end_do_while 706	R	T	I	Fine costrutto DO
endif 710	R	T	I	Fine costrutto IF
exl_lam 316()	R	T	F	Mostra gli assi logici inclusi/esclusi di un powerset
excl_lam 711	R	T	I	Include/esclude gli assi logici di un powerset
excl_pam 317()	R	T	F	Mostra gli assi fisici inclusi/esclusi di un powerset

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
excl_pam 712	R	T	I	Include/esclude gli assi fisici di un powerset
f_avail 318()	-	T	F	Memoria disponibile su flash
f_close 318()	-	T	F	Chiusura di un file aperto in Flash
f_copy 319()	-	T	F	Copia un file presente sulla CF
f_del 320()	-	T	F	Cancellazione di un file sulla Flash
f_dir 321()	-	T	F	Informazioni sui file contenuti nella Flash
f_gets 324()	-	T	F	Legge da un file nella flash aperto in modo testo
f_mkdir 326()	-	T	F	Crea una nuova directory in CF
f_open 327()	-	T	F	Apertura di un file in flash
f_puts 329()	-	T	F	Scrive in un file sulla flash aperto in modo testo
f_read 330()	-	T	F	Legge da un file nella flash aperto in modo binario
f_read_table 331()	-	T	F	Legge da flash un file ascii e carica i valori in esso contenuti in variabili
f_read_table_s 333()	-	t	F	Legge da flash un file ascii e carica i valori in esso contenuti in variabili (controlla eventuali sbordamenti dal buffer)
f_rename 335()	-	T	F	Rinomina un file sulla CF

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
f_seek 336()	-	T	F	Sposta il puntatore all'interno di un file sulla flash
f_write 337()	-	T	F	Scrive in un file sulla flash aperto in modo binario
f_format 323()	-	T	F	Formattazione partizioni Flash
fall 338()	R	T	F	Coglie il fronte di discesa di una condizione
fall_filt 339()	R	T	F	Coglie il fronte di discesa di una condizione filtrato
fill_mem 712	R	T	I	Permette di inizializzare un area di memoria
filter_lp 341()	R	-	F	Filtro passa basso sulla variabile
filter_lp_st 342()	R	-	F	Filtro passa basso sulla variabile utilizzabile in un ciclo for
filter_notch 343()	R	-	F	Filtro notch sulla variabile
filter_notch_st 345()	R	-	F	Filtro passa basso sulla variabile utilizzabile in un ciclo for
first_time 346()	R	-	F	Gira solo la prima volta che la regola viene attivata
for 713	R	T	I	Ciclo for
force 347()	R	T	F	Forza gli IO
fract 350()	R	T	F	Restituisce parte frazionaria dell'argomento
function 715()	R	T	I	Funzione (sottoprogramma)
get_u8 358()	R	T	F	Leggi un byte (u8) da un buffer in un

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				offset specificato
get_i8 355()	R	T	F	Leggi un byte (i8) da un buffer in un offset specificato
get_u16 356()	R	T	F	Leggi una word (u16) da un buffer in un offset specificato
get_i16 353()	R	T	F	Leggi una word (i16) da un buffer in un offset specificato
get_u32 357()	R	T	F	Leggi una dword (u32) da un buffer in un offset specificato
get_i32 354()	R	T	F	Leggi una dword (i32) da un buffer in un offset specificato
get_u64 720()	R	T	I	Leggi una qword (u64) da un buffer in un offset specificato
get_i64 719()	R	T	I	Leggi una qword (i64) da un buffer in un offset specificato
get_float 350()	R	T	F	Leggi un float (x32) da un buffer in un offset specificato
get_hw_info 351	-	T	F	Permette di avere informazioni relative alle schede direttamente connesse alla CPU
get_real 355()	R	T	F	Leggi un real (x64) da un buffer in un offset specificato
get_virt_ow 359()	R	T	F	Legge lo stato reale degli output virtuali

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
goto 721	R	T	I	Istruzione di salto alla label
group 721	R	T	I	Attiva gli esecutori delle regole
gv_create 359()	-	T	F	Crea una nuova variabile globale. Crea le informazioni in memoria e, se necessario, nella memoria ritentiva
gv_datatype 362()	R	T	F	Ottiene il tipo di una variabile globale
gv_datatype_byname 362()	R	T	F	Ottiene il tipo di una variabile globale
gv_destroy 363()	-	T	F	Rimuove una variabile globale. elimina il file di configurazione e lo spazio in memoria
gv_exist 364()	R	T	F	Controlla se la stringa specificta coincide con una variabile globale definita
gv_flags 364()	R	T	F	Ottiene il flag associato a una variabile globale
gv_flags_byname 366()	R	T	F	Ottiene il flag associato a una variabile globale
gv_link 368()	R	T	F	Link a una variabile globale esistente. Crea un'interfaccia a una GV
gv_lock 369()	R	T	F	Blocca l'accesso della variaile globale - l'accesso non è permesso da nessun'altra interfaccia

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
gv_name 370()	R	T	F	Ottiene il nome di una variabile globale
gv_read 370()	R	T	F	Aggiorna il dato della variabile locale associata tramite l'interfaccia, leggendo dalla GV
gv_read_partial 371()	R	T	F	Aggiorna parzialmente il dato della variabile locale associata tramite l'interfaccia, leggendo dalla GV
gv_size 372()	R	T	F	Ottiene la dimensione di una variabile globale
gv_size_byname 372()	R	T	F	Ottiene la dimensione di una variabile globale
gv_unlink 373()	R	T	F	Rimuove la connessione dell'interfaccia a una variabile globale
gv_unlock 373()	R	T	F	Sblocca l'accesso della variaile globale - l'accesso è quindi permesso da tutte le interfacce
gv_update_file 374()	-	T	F	Aggiorna le informazioni nel file di configurazione
gv_write 374()	R	T	F	Aggiorna il valore della variabile globale con il valore della variabile locale associata tramite l'interfaccia
gv_write_partial 375()	R	T	F	Aggiorna parzialmente il valore della variabile globale

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				con il valore della variabile locale associata tramite l'interfaccia
if 723	R	T	I	Costrutto IF
in_range 376()	R	T	F	Restituisce il valore nel range indicato
inc 377()	R	T	F	Incrementa la variabile
index_of 377()	R	T	F	Restituisce l'indice della variabile simbolica
inf_report 724	R	T	I	Permette di inserire una informazione nel report
int_after 725	R	T	I	Abilta regola asincrona ritardata su evento
int_inp 726	R	T	I	Abilita rule asincrona su variazione input interruptante
int_timer 728	R	T	I	Abilita rule asincrona a frequenza fissa
ipaddr_to_str 378()	R	T	F	Converte un valore numerico in una stringa contenente l'IP
is_alarm_present 383()	R	T	F	Verifica la presenza dell'allarme e del parametro nello stack
is_alarm_par_present 379()	R	T	F	Verifica la presenza dell'allarme nello stack
is_alarm_par_present_us 381()	R	T	F	Verifica la presenza dell'allarme con parametro USV nello stack

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
is_forced 384()	R	T	F	Verifica gli IO forzati
lit 729	R	T	I	Associazione di un simbolico ad una stringa di testo
log 388()	R	T	F	Restituisce il logaritmo naturale dell'argomento
log10 388()	R	T	F	Restituisce il logaritmo base10 dell' argomento
loop_time 389()	R	T	F	Calcola la durata temporale tra due chiamata successive
master_filter 389()	R	T	F	Filtra la velocità di un master
maximum 390()	R	T	F	Restituisce il più grande tra due valori
mbc_connect 391()	-	T	F	Connette il client al server Modbus/TCP
mbc_init 391()	-	T	F	Inizializza client Modbus/TCP
mbc_mask_write_reg 394()	-	T	F	Scrittura registers con maschera
mbc_mask_write_reg_s 396()	-	T	F	Scrittura registers con maschera tramite struttura di appoggio
mbc_read_coils 398()	-	T	F	Lettura coils
mbc_read_coils_s 399()	-	T	F	Lettura coils con struttura di appoggio
mbc_read_discr_inp 400()	-	T	F	Lettura discrete inputs
mbc_read_discr_inp_s 402()	-	T	F	Lettura discrete inputs con struttura di appoggio

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
mbc_read_hold_reg 403()	-	T	F	Lettura holding registers
mbc_read_hold_reg_s 405()	-	T	F	Lettura holding registers con struttura di appoggio
mbc_read_inp_reg 406()	-	T	F	Lettura input registers
mbc_read_inp_reg_s 408()	-	T	F	Lettura input registers con struttura di appoggio
mbc_shutdown 410()	-	T	F	Chiude la connessione con il server Modbus/TCP
mbc_write_coils 410()	-	T	F	Scrittura coils
mbc_write_coils_s 412()	-	T	F	Scrittura coils tramite struttura di appoggio
mbc_write_multiple_coils 413()	-	T	F	Scrittura coils usando codice funzione Modbus delle scritture multiple corrispettive
mbc_write_multiple_coils_s 414()	-	T	F	Scrittura coils tramite struttura di appoggio usando codice funzione Modbus delle scritture multiple corrispettive
mbc_write_multiple_regs 416()	-	T	F	Scrittura registers usando codice funzione Modbus delle scritture multiple corrispettive
mbc_write_multiple_regs_s 418()	-	T	F	Scrittura registers tramite struttura di appoggio usando codice funzione Modbus delle

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				scritture multiple corrispettive
mbc_write_reg 419 ()	-	T	F	Scrittura registers
mbc_write_reg_s 421()	-	T	F	Scrittura registers tramite struttura di appoggio
mbcu_mask_write_reg 422()	-	T	F	Scrittura registers con maschera specificando lo unitID
mbcu_read_coils 424()	-	T	F	Lettura coils specificando lo unitID
mbcu_read_discr_inp 426()	-	T	F	Lettura discrete inputs specificando lo unitID
mbcu_read_hold_reg 427()	-	T	F	Lettura holding registers specificando lo unitID
mbcu_read_inp_reg 429()	-	T	F	Lettura input registers specificando lo unitID
mbcu_write_coils 431()	-	T	F	Scrittura coils specificando lo unitID
mbcu_write_multiple_coils 433()	-	T	F	Scrittura coils specificando lo unitID usando codice funzione Modbus delle scritture multiple corrispettive
mbcu_write_multiple_regs 434()	-	T	F	Scrittura registers specificando lo unitID usando codice funzione Modbus delle scritture multiple corrispettive
mbcu_write_reg 436()	-	T	F	Scrittura registers specificando lo unitID

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
minimum 438()	R	T	F	Restituisce il più piccolo tra due valori
mod 439()	R	T	F	Funzione che rende il resto della divisione fra variabili numeriche
mot_append 439()	R	T	F	Funzione per gestione MOT (multiple output table)
mot_clear 443()	R	T	F	Funzione per gestione MOT (multiple output table)
mot_create 443()	R	T	F	Funzione per gestione MOT (multiple output table)
mot_delete 447()	R	T	F	Funzione per gestione MOT (multiple output table)
mot_get 447()	R	T	F	Funzione per gestione MOT (multiple output table)
mot_remove 449()	R	T	F	Funzione per gestione MOT (multiple output table)
motion 730	R	-	I	Inizio Blocco Motion all'interno di una rule
move_byte 731	R	T	I	Copia una zona di memoria
move_mem 732	R	T	I	Copia una zona di memoria
mt_ats 449()	R	T	F	Test atomico di una variabile
mt_ats2 450()	R	T	F	Test atomico di una variabile

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
mt_dis 733	-	T	I	Abilita l'esecuzione di un task (time-sharing)
mt_en 734	-	T	I	Disabilita l'esecuzione di un task (time-sharing)
mt_rr_flag 734	-	T	I	Manda in esecuzione un OB infinito (loop)
mt_wait 735	-	T	I	Mette in attesa un TASK
mv_cam 451()	R	-	F	Esegue una camma cubica per tratti a,b,c)
mv_crimper 453()	R	-	F	Funzione crimper per flow-pack
mv_follow 454()	R	-	F	Taglio al volo (consigliata la follow2)
mv_follow2 458()	R	-	F	Taglio al volo (master e slave non associati a un asse)
mv_mot_exec 470()	R	T	F	Funzione di interpolazione mediante tabella MOT (multiple output table)
mv_phase_adj 474()	R	-	F	Correzione di fase tra due assi
mv_phase_adj2 474()	R	-	F	Correzione di fase tra due assi
mv_ramp 477()	R	-	F	Deprecata. Usare ramp 555()
mv_reach_target 477()	R	-	F	Aggancio tra due assi
mv_sine_cam 478()	R	-	F	Esecuzione di una camma (y in funzione di x)
mv_synchro 479()	R	-	F	Aggancio tra due assi

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
mv_table 481()	R	-	F	Esegue una Camma per punti
mv_to 482()	R	-	F	Movimento verso un target
mv_to_cj 486()	R	-	F	Movimento verso un target con jerk controllato
mv_to_cjv 487()	R	-	F	Movimento verso un target con jerk controllato
mv_to_cjv_info 489()	R	T	F	Funzione di supporto a MV_TO_CJV
mv_to_vel 489()	R	-	F	Movimento verso un target con una velocità finale specificata
mv_tracking 491()	R	-	F	Deprecata. Usare tracking 648()
mva_follow2 458()	R	-	F	Taglio al volo (solo slave associato a un asse)
mva_follow2_m 458()	R	-	F	Taglio al volo (master e slave associati a un asse)
mva_jog 491()	R	-	F	Movimentazione JOG (manuale)
mva_jog2 493()	R	-	F	Movimentazione JOG (manuale)
mva_open_loop 496()	R	-	F	Apertura dell'anello
mva_to_n 496()	R	-	F	Movimento verso un target con piu assi
mva_to_n_cj 500()	R	-	F	Movimento verso un target con piu assi
mva_to_n_v 504()	R	-	F	Movimento verso un target con piu assi anche già in movimento

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
mva_zc 511()	R	-	F	Movimento di Homing
normalize 513()	R	T	F	Restituisce valore normalizzato tra 0 e 1
not 809	R	T	I	Operatore logico not
onoff_filt 514()	R	T	F	Coglie i fronti di salita e discesa filtrati di una condizione
or 809	R	T	I	Operatore logico or
order 735	R	T	I	Stabilisce l'ordine di esecuzione degli esecutori delle regole
out_m 737	R	T	I	Modifica lo stato dei canali di output
par_even 516()	R	T	F	Funzione che rende vero se il numero di bit in 1 del byte passatogli è pari
par_odd 517()	R	T	F	Funzione che rende vero se il numero di bit in 1 del byte passatogli è dispari
planar_comp_ip 519()	R	-	F	Funzione che applica la position compensation table diretta
planar_comp_cp 518()	R	-	F	Funzione che applica la position compensation table inversa
pos_capture_cmd 520()	R	T	F	Funzione per catturare hardware la quota di un asse
pos_capture_sts 524()	R	T	F	Funzione di supporto per catturare

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				hardware la quota di un asse
pow 525()	R	T	F	Elevazione a potenza
ps_exclude_lax 538()	R	T	F	Esclude gli assi logici di un powerset
ps_exclude_pax 539()	R	T	F	Esclude gli assi fisici di un powerset
ps_reinclude_lax 541()	R	T	F	Include gli assi logici di un powerset
ps_reinclude_pax 542()	R	T	F	Include gli assi fisici di un powerset
ps_power_enable 540()	R	T	F	Power set (gestione potenza) -Abilita la potenza del power set-
ps_channel_enable 526()	R	T	F	Power set (gestione potenza) -Abilita gli assi del power set-
ps_channel_reset_fault 532()	R	T	F	Power set (gestione potenza) -Invia un comando di reset allarme-
ps_drive_reset 535()	R	T	F	Power set (gestione potenza) -Invia un comando di reset drive-
ps_status 543()	R	T	F	Power set (gestione potenza) -Rende informazioni sullo stato del power set-
ps_channel_enabled 529()	R	T	F	Power set (gestione potenza) -Rende informazioni circa gli assi abilitati di un power set-

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
ps_channel_fault 531()	R	T	F	Power set (gestione potenza) -Rende informazioni circa gli assi in fault di un power set-
qwbe2le 738	R	T	I	Converti big endian qword (x64) in little endian qword (x64)
qwbe2lef 547()	R	T	F	Converti big endian qword (x64) in little endian real (x64)
qwbedwx2le 739	R	T	I	Converti dword exchange big endian qword (x64) in little endian qword (x64)
qwbedwx2lef 547()	R	T	F	Converti dword exchange big endian qword (x64) in little endian real (x64)
qwle2be 739	R	T	I	Converti little endian qword (x64) in big endian qword (x64)
qwle2bedwx 740	R	T	I	Converti little endian qword (x64) in dword exchange big endian qword (x64)
qwle2bedwxf 740	R	T	I	Converti little endian real (x64) in dword exchange big endian real (x64)
qwle2bef 741	R	T	I	Converti little endian real (x64) in big endian qword (x64)
qwle2mbe 741	R	T	I	Converti little endian qword

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				(x64) in middle big endian qword (x64)
qwle2mbedwx 740	R	T	I	Converti little endian qword (x64) in dword exchange middle big endian qword (x64)
qwle2mbedwxf 740	R	T	I	Converti little endian real (x64) in dword exchange middle big endian qword (x64)
qwle2mbef 743	R	T	I	Converti little endian real (x64) in middle big endian qword (x64)
qwmbe2le 743	R	T	I	Converti middle big endian qword (x64) in little endian qword (x64)
qwmbe2lef 548 ()	R	T	F	Converti middle big endian qword (x64) in little endian real (x64)
qwmbedwx2le 744	R	T	I	Converti dword exchange middle big endian qword (x64) in little endian qword (x64)
qwmbedwx2lef 548 ()	R	T	F	Converti dword exchange middle big endian qword (x64) in little endian real (x64)
r_and 806	R	T	I	And logico bit a bit
r_not 806	R	T	I	Not logico bit a bit
r_or 806	R	T	I	Or logico bit a bit
r_rol 549 ()	R	T	F	Rotazione a sinistra del numero di bit specificati

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
r_ror 550()	R	T	F	Rotazione a destra del numero di bit specificati
r_shl 551()	R	T	F	Shift a sinistra del numero di bit specificati
r_shr 552()	R	T	F	Shift a destra del numero di bit specificati
r_xor 806	R	T	I	Or esclusivo bit a bit
r2m16 553()	R	T	F	Conversione da real a U16
r2m32 554()	R	T	F	Conversione da real a U32
r2m8 554()	R	T	F	Conversione da real a U8
ramp 555()	R	T	F	Rampa la grandezza verso un valore target
rand32 556()	R	T	F	Rende un numero casuale (i16)
random 557()	R	T	F	Rende un numero casuale (i32)
random_value 557()	R	T	F	Rende un numero casuale (real)
range 557()	R	T	F	Rende TRUE se la variabile e nel range
range_adj 558()	R	T	F	Scaling di una variabile
rbit 559()	R	T	F	Reset di bit in una variabile intera
ref 744	R	-	I	Inizio Blocco Reference all'interno di una rule
ref_prologue 745()	R	-	I	Rule speciale
release 560()	R	T	F	Rilascia gli IO forzati

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
restart_point 746	-	T	I	Punto di interruzione programma in caso di live modification 933
return 746	R	T	I	Interrompe una funzione
rise 562()	R	T	F	Coglie il fronte di salita di una condizione
rise_filt 563()	R	T	F	Coglie il fronte di salita di una condizione filtrato
round 565()	R	T	F	Rende il numero intero piu vicino
rtc_date 565()	R	T	F	Setta la data del clock calendar
rtc_time 566()	R	T	F	Setta l'orario del clock calendar
rule 748	R	-	I	Definizione di una regola
rule_epilogue 756	R	-	I	Rule speciale
rule_freq 756	R	T	I	Definizione intervallo di campionamento
rule_init 757	R	-	I	Regola di inizializzazione (gira solo all'accensione)
rule_periodic 758	R	T	I	Manda in esecuzione una regola a frequenza impostata (bassa priorità)
rule_prologue 759	R	-	I	Rule speciale
safe_appl_ack 567 ()	R	T	F	Setta il valore dell'Application Acknowledge di un'isola safety
safe_appl_diag 568 ()	R	T	F	Ritorna lo stato degli Application

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				Diagnostic di un'isola safety
safe_diag 569()	R	T	F	Ritorna il registro di diagnostica del satellite desiderato
safe_diag_reset 572()	R	T	F	Invia un Diagnostic Reset al satellite selezionato
safe_enab_out 573()	R	T	F	Imposta il valore del registro di abilitazione degli output dell'isola selezionata
safe_oper_ack 573()	R	T	F	Impone un Operation Acknowledge sul satellite selezionato
safe_prj_info 574()	-	T	F	Permette di ottenere informazioni riguardo un progetto safety
safe_sts 575()	R	T	F	Ritorna lo stato degli input/output safety del satellite selezionato
sb0 577() / sb0 760()	R	T	F/I	Restituisce il byte 0 dell'argomento con estensione del segno
sb1 578() / sb1 761()	R	T	F/I	Restituisce il byte 1 dell'argomento con estensione del segno
sb2 578() / sb2 761()	R	T	F/I	Restituisce il byte 2 dell'argomento con estensione del segno
sb3 579() / sb3 761()	R	T	F/I	Restituisce il byte 3 dell'argomento con estensione del segno

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
sb4 580() / sb4 762()	R	T	F/I	Restituisce il byte 4 dell'argomento con estensione del segno
sb5 580() / sb5 762()	R	T	F/I	Restituisce il byte 5 dell'argomento con estensione del segno
sb6 581() / sb6 762()	R	T	F/I	Restituisce il byte 6 dell'argomento con estensione del segno
sb7 582() / sb7 763()	R	T	F/I	Restituisce il byte 7 dell'argomento con estensione del segno
sbit 582()	R	T	F	Set di bit in una variabile intera
sdw0 583() / sdw0 763()	R	T	F/I	Restituisce/scrive la dword 0 dell'argomento con l'estensione del segno
sdw1 584() / sdw1 763()	R	T	F/I	Restituisce/scrive la dword 1 dell'argomento con l'estensione del segno
search()	R	T	F	Riservata
select 764	R	T	I	Costrutto select end_select
ser_485_tx_sts 585()	R	T	F	Funzione che rende lo stato dell'ultima trasmissione fatta sul dispositivo specificato
ser_bcc_h_init 587()	R	T	F	Inizializza la seriale per la comunicazione BCC3 tra controlli
ser_bin_in 588()	R	T	F	Funzione che permette di leggere un numero specifico di

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				caratteri dal dispositivo specificato
ser_bin_out 588()	R	T	F	Invia i caratteri sul canale seriale
ser_bin_ready 589()	R	T	F	Segnala se ci sono caratteri sul canale seriale
ser_buff_free 590()	R	T	F	Numero di Bytes liberi sul buffer TX
ser_buff_queue 590()	R	T	F	Numero di Bytes occupati sul buffer TX
ser_con 765	R	T	I	Imposta i parametri del canale seriale
ser_flush 590()	R	T	F	Pulisce tutti i dati nei buffer di tx e rx della seriale specificata
ser_in 591()	R	T	F	Riceve da seriale la stringa terminata
ser_mod 592()	R	T	F	Imposta la modalità di lavoro del canale seriale specificato
ser_out 593()	R	T	F	Invia sul seriale la stringa terminata
ser_r_sts 593()	R	T	F	Legge la parola di stato del seriale
ser_ready 594()	R	T	F	Segnala la presenza di una stringa terminata sul seriale
ser_w_sts 766()	R	T	I	Imposta la parola di stato del seriale
set_i8 771()	R	T	I	Scrivi un byte (i8) da un buffer in un offset specificato
set_u8 775()	R	T	I	Scrivi un byte (u8) da un buffer in un offset specificato

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
set_i16 768()	R	T	I	Scrivi una word (i16) da un buffer in un offset specificato
set_u16 772()	R	T	I	Scrivi una word (u16) da un buffer in un offset specificato
set_i32 769()	R	T	I	Scrivi una dword (i32) da un buffer in un offset specificato
set_u32 773()	R	T	I	Scrivi una dword (u32) da un buffer in un offset specificato
set_i64 770()	R	T	I	Scrivi una qword (i64) da un buffer in un offset specificato
set_u64 774()	R	T	I	Scrivi una qword (u64) da un buffer in un offset specificato
set_float 767()	R	T	I	Scrivi un float (x32) da un buffer in un offset specificato
set_kinematics 595()	R	-	F	Impone le grandezze cinematiche contemporaneamente
set_real 772()	R	T	I	Scrivi un real (x64) da un buffer in un offset specificato
set_version 776()	R	T	I	Permette di scrivere la versione software dell'applicativo
set_virtual_iw 595()	R	T	F	Imposta lo stato reale degli input virtuali
shift_left 596()	R	T	F	Shift di un bit a sinistra dell'area di memoria indicata

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
shift_right 597()	R	T	F	Shift di un bit a destra dell'area di memoria indicata
shl 599()	R	T	F	Shift logico a sinistra
shr 599()	R	T	F	Shift logico a destra
sign 600()	R	T	F	Restituisce il segno della variabile
similar 601()	R	T	F	Confronta due variabili e stabilisce se sono simili
sin 602()	R	T	F	Seno
sizeof 602()	R	T	F	Rende la dimensione della variabile
soe_ridn 603()	R	T	F	Funzione per leggere i parametri su fieldbus SOE
soe_widn 615()	R	T	F	Funzione per scrivere i parametri su fieldbus SOE
sqrt 620()	R	T	F	Esegue la radice quadrata
statistics 621()	R	T	F	Esegue statistiche sulla variabile indicata
str_asc 624()	R	T	F	Converte un carattere in numero
str_chr 625()	R	T	F	Converte un numero in carattere
str_compact 625()	R	T	F	Compatta gli spazi
str_equal 626()	R	T	F	Confronta due stringhe
str_find 627()	R	T	F	Cerca all'interno di una stringa

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
str_format627()	R	T	F	Formatta la stringa (sprintf)
str_i629()	R	T	F	Converte un valore numerico intero in AscII
str_insert630()	R	T	F	Inserisce all'interno di una stringa
str_len631()	R	T	F	Calcola la lunghezza della stringa
str_mid631()	R	T	F	Prende nella stringa
str_r632()	R	T	F	Converte un valore numerico reale in AscII
str_replace633()	R	T	F	Sostituisce all'interno di una stringa
str_val635()	R	T	F	Rende, in formato numerico, un valore contenuto in una stringa
str_to_ipaddr634()	R	T	F	Converte una stringa contenente l'IP in un valore numerico
store_nvr_nopf622()	R	T	F	Salva i NVR specificati nella memoria statica
store_nvrr_nopf623()	R	T	F	Salva i NVRR specificati nella memoria statica
sum8_mem637()	R	T	F	Restituisce la somma di n elementi di un array di U8
sum16_mem638()	R	T	F	Restituisce la somma di n elementi di un array di U16
sum32_mem638()	R	T	F	Restituisce la somma di n

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				elementi di un array di U32
sumr_mem 638()	R	T	F	Restituisce la somma di n elementi di un array di real
sw0 639() / sw0 777()	R	T	F/I	Restituisce/scrive la word 0 dell'argomento con l'estensione del segno
sw1 639() / sw1 777()	R	T	F/I	Restituisce/scrive la word 1 dell'argomento con l'estensione del segno
sw2 640() / sw2 778()	R	T	F/I	Restituisce/scrive la word 2 dell'argomento con l'estensione del segno
sw3 641() / sw3 778()	R	T	F/I	Restituisce/scrive la word 3 dell'argomento con l'estensione del segno
tan 641()	R	T	F	Tangente
terminator 778	R	T	I	Inserisce un terminatore
time_micro 642()	R	T	F	Restituisce il tempo in microsecondi dall'accensione del controllo assi
tm_et 642() / tm_et 779()	R	T	F/I	Legge/setta il tempo del timer
tm_int 779()	R	T	I	Timer integrale
tm_pt 643() / tm_pt 784()	R	T	F/I	Legge/setta il preset del timer
tm_pt2 644() / tm_pt2 784()	R	T	F/I	Legge/setta il secondo preset del timer (timer square)

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
tm_q 645()	R	T	F	Uscita del timer (bobina)
tm_reset 784()	R	T	I	Resetta il tempo del timer (timer integrale)
tm_sqw 784()	R	T	I	Timer square wawe
tm_of 780()	R	T	I	Timer OFF
tm_on 782()	R	T	I	Timer ON
tm_tp 786()	R	T	I	Timer pulse
tobcd 645()	R	T	F	Converte il valore da binario a BCD
tobin 646()	R	T	F	Converte il valore da BCD a binario
todeg 646()	R	T	F	Converte il valore da radianti a gradi
tolower 647()	R	T	F	Converte un carattere in minuscolo
torad 647()	R	T	F	Converte il valore da gradi a radianti
toupper 648()	R	T	F	Converte un carattere in maiuscolo
tracking 648()	R	T	F	Porta la grandezza controllata verso il target specificato
trunc 650()	R	T	F	Restituisce la parte intera dell'argomento
udp_open_client 651()	R	T	F	Apre una connessione client udp
udp_open_server 654()	R	T	F	Apre una connessione server udp
udp_close 651()	R	T	F	Chiude una connessione udp

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
udp_send 662()	R	T	F	Invia un pacchetto dati udp
udp_send_to 667()	R	T	F	Invia un pacchetto dati udp
udp_recv 657()	R	T	F	Riceve un pacchetto dati udp
udp_recv_from 658()	R	T	F	Riceve un pacchetto dati udp
udp_send_notify 665()	R	T	F	Invia un pacchetto dati udp con notifica
udp_recv_notify 660()	R	T	F	Riceve un pacchetto dati udp con notifica
virt_io 669()	-	T	F	Definisce un area di I/O virtuali
w0 671() / w0 787()	R	T	F/I	Legge/scrive la word 0 dell'argomento
w1 672() / w1 787()	R	T	F/I	Legge/scrive la word 1 dell'argomento
w2 672() / w2 788()	R	T	F/I	Legge/scrive la word 2 dell'argomento
w3 673() / w3 788()	R	T	F/I	Legge/scrive la word 3 dell'argomento
wbe2le 674()	R	T	F	Converti big endian word (x16 o x32) in little endian word (x16)
wle2be 675()	R	T	F	Converti little endian word (x16 o x32) in big endian word (x16)
wbit 674()	R	T	F	Set/Reset di bit in una variabile intera
while 789	R	T	I	Costrutto while end_while

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
write_dis 676()	R	T	F	Scrive una stringa sul display 4*20
xor 809	R	T	I	Operatore logico or esclusivo

Codici di errore funzioni

Elenco dei codici disponibili.

Codice	Etichetta	Descrizione
1	EPERM	Operazione non consentita
2	ENOENT	Nessun file o directory di questo tipo
3	ESRCH	Nessun processo di questo tipo
4	EINTR	Chiamata di sistema interrotta
5	EIO	Errore di I/O
6	ENXIO	Nessun dispositivo o indirizzo di questo tipo
7	E2BIG	Elenco di argomenti troppo lungo
8	ENOEXEC	Errore di formato dell'esecuzione
9	EBADF	Numero di file errato
10	ECHILD	Nessun processo figlio
11	EAGAIN	Riprovare
12	ENOMEM	Memoria esaurita
13	EACCES	Permessi negati
14	EFAULT	Indirizzo errato
15	ENOTBLK	Dispositivo di blocco richiesto
16	EBUSY	Dispositivo o risorsa occupata
17	EEXIST	Il file esiste
18	EXDEV	Collegamento tra dispositivi
19	ENODEV	Dispositivo non esistente

© 2025 Robox SpA

Codice	Etichetta	Descrizione
20	ENOTDIR	Non è una directory
21	EISDIR	È una directory
22	EINVAL	Argomento non valido
23	ENFILE	Overflow della tabella dei file
24	EMFILE	Troppi file aperti
25	ENOTTY	Non è una macchina da scrivere
26	ETXTBSY	File di testo occupato
27	EFBIG	File troppo grande
28	ENOSPC	Non c'è più spazio sul dispositivo
29	ESPIPE	Ricerca illegale
30	EROFS	File system di sola lettura
31	EMLINK	Troppi collegamenti
32	EPIPE	Tubo rotto
33	EDOM	Argomento matematico fuori dal dominio della funzione
34	ERANGE	Risultato matematico non rappresentabile
35	EDEADLK	Si verificherebbe un deadlock delle risorse
36	ENAMETOOLONG	Nome del file troppo lungo
37	ENOLCK	Nessun record lock disponibile
38	ENOSYS	Funzione non implementata
39	ENOTEMPTY	Directory non vuota
40	ELOOP	Troppi link simbolici incontrati
42	ENOMSG	Nessun messaggio del tipo desiderato
43	EIDRM	Identificatore rimosso
44	ECHRNG	Numero di canale fuori portata
48	ELNRNG	Numero di collegamento fuori campo

© 2025 Robox SpA

Codice	Etichetta	Descrizione
49	EUNATCH	Driver di protocollo non collegato
53	EBADR	Descrittore di richiesta non valido
56	EBADRQC	Codice richiesta non valido
57	EBADSLT	Slot non valido
60	ENOSTR	Il dispositivo non è un flusso
61	ENODATA	Nessun dato disponibile
62	ETIME	Timer scaduto
63	ENOSR	Risorse del flusso esaurite
64	ENONET	La macchina non è in rete
65	ENOPKG	Pacchetto non installato
66	EREMOTE	L'oggetto è remoto
67	ENOLINK	Il collegamento è stato interrotto
68	EADV	Errore di pubblicità
69	ESRMNT	Errore di montaggio
70	ECOMM	Errore di comunicazione all'invio
71	EPROTO	Errore di protocollo
72	EMULTIHOP	Tentativo di multihop
73	EDOTDOT	Errore specifico RFS
74	EBADMSG	Non è un messaggio di dati
75	EOVERFLOW	Valore troppo grande per il tipo di dati definito
76	ENOTUNIQ	Nome non univoco sulla rete
77	EBADFD	Descrittore di file in cattivo stato
78	EREMCHG	Indirizzo remoto cambiato
79	ELIBACC	Impossibile accedere a una libreria condivisa necessaria
80	ELIBBAD	Accesso a una libreria condivisa danneggiata
81	ELIBSCN	Sezione .lib in a.out danneggiata

© 2025 Robox SpA

Codice	Etichetta	Descrizione
82	ELIBMAX	Tentativo di collegare troppe librerie condivise
83	ELIBEXEC	Impossibile eseguire direttamente una libreria condivisa
84	EILSEQ	Sequenza di byte illegale
85	ERESTART	La chiamata di sistema interrotta deve essere riavviata
86	ESTRPIPE	Errore della stream pipe
87	EUSERS	Troppi utenti
88	ENOTSOCK	Operazione su socket su non-pocket
89	EDESTADDRREQ	Indirizzo di destinazione richiesto
90	EMGSIZE	Messaggio troppo lungo
91	EPROTTYPE	Protocollo di tipo sbagliato per il socket
92	ENOPROTOOPT	Protocollo non disponibile
93	EPROTONOSUPPORT	Protocollo non supportato
94	ESOCKTNOSUPPORT	Tipo di socket non supportato
95	EOPNOTSUPP	Operazione non supportata sull'endpoint di trasporto
96	EPFNOSUPPORT	Famiglia di protocollo non supportata
97	EAFNOSUPPORT	Famiglia di indirizzi non supportata dal protocollo
98	EADDRINUSE	Indirizzo già in uso
99	EADDRNOTAVAIL	Impossibile assegnare l'indirizzo richiesto
100	ENETDOWN	La rete è inattiva
101	ENETUNREACH	La rete è irraggiungibile
102	ENETRESET	La rete ha interrotto la connessione a causa di un reset
103	ECONNABORTED	Interruzione della connessione causata dal software

© 2025 Robox SpA

Codice	Etichetta	Descrizione
104	ECONNRESET	Connessione resettata dal peer
105	ENOBUFS	Non c'è spazio disponibile per il buffer
106	EISCONN	L'endpoint di trasporto è già connesso
107	ENOTCONN	L'endpoint di trasporto non è connesso
108	ESHUTDOWN	Impossibile inviare dopo l'arresto dell'endpoint di trasporto
109	ETOOMANYREFS	Troppi riferimenti: impossibile giuntare
110	ETIMEDOUT	Connessione scaduta
111	ECONNREFUSED	Connessione rifiutata
112	EHOSTDOWN	L'host è inattivo
113	EHOSTUNREACH	Nessun percorso verso l'host
114	EALREADY	Operazione già in corso
115	EINPROGRESS	Operazione in corso
116	ESTALE	Maniglia di file NFS obsoleta
117	EUCLEAN	La struttura deve essere pulita
120	EISNAM	È un file di tipo denominato
121	EREMOTEIO	Errore di I/O remoto
122	EMSGSIZE	Messaggio troppo lungo
123	ENOMEDIUM	Nessun supporto trovato
124	EMEDIUMTYPE	Tipo di supporto sbagliato

Codici di reso dalle funzioni MV_ e MVA_

Elenco dei codici di reso comuni alle funzioni e istruzioni di gestione movimento assi.

Simbolico	Valore	Significato
M_IN_ALARM	-6	Situazione di allarme presente.
M_ILLEGAL_ARGS	-5	Argomento/i illegale.
M_LOW_ACCELERATION	-4	Accelerazione insufficiente.

© 2025 Robox SpA

Simbolico	Valore	Significato
M_SLAVE_EARLY_STOPPED	-3	Slave in anticipo, fermato.
M_SLAVE_IS_EARLY	-2	Slave in anticipo all'appuntamento.
M_SLAVE_TOO_LATE	-1	Slave in ritardo all'appuntamento.
M_INITIALIZING	1	Inizializzazione funzione.
M_POSITIONING	2	Posizionamento iniziale.
M_WAITING	3	Attesa partenza.
M_WAITING_NEXT_CYCLE	4	Attesa prossimo ciclo.
M_FOLLOWING	5	Inseguimento in corso.
M_REACHED	6	Target raggiunto/missione terminata.
M_HOMING	7	Recupero posizione di inizio ciclo / azzeramento in corso.
M_SYNCHRONIZED	8	Velocità sincronizzata al master.
M_ACCEL_LIMITED	9	Accelerazione limitata alla max. desiderata.
M_SPEED_LIMITED	10	Velocità limitata alla max. desiderata.
M_ADJUSTING_UP	11	Correzione in corso (salita triangolo).
M_ADJUSTING_DOWN	12	Correzione in corso (discesa triangolo).
M_OVERSHOOTING	13	La posizione target è/sarà oltrepassata.
M_HOLDED	14	Movimento arrestato causa flag HOLD. Viene restituito anche nei casi in cui la velocità desiderata dell'asse viene impostata a zero (es. il campo dSpeed della struct stru_mvtocjv) oppure il feedrate viene impostato a zero.
M_ACCEL	15	Accelerazione in corso.
M_AT_SPEED	16	Movimentazione a regime.
M_DECEL	17	Decelerazione in corso.

© 2025 Robox SpA

Simbolico	Valore	Significato
M_LAST_CYCLE	18	Ultimo ciclo prima di oltrepassare pDes.
M_OUT_OF_LIMITS	19	Fuori dai limiti prefissati.

Codici di errore SDO (CoE/SoE)

Codici di errore SDO CoE/SoE

SDO error code	Descrizione
0x0500 0000	Trovato errore generale protocollo SDO.
0x0503 0000	Toggle bit non alternato.
0x0504 0000	Timeout del protocollo SDO.
0x0504 0001	Identificatore del comando client/server non valido o sconosciuto.
0x0504 0002	Dimensione del blocco non valida (solo gestione bloccante).
0x0504 0003	Numero di sequenza non valido (solo gestione bloccante).
0x0504 0004	Errore CRC (solo gestione bloccante).
0x0504 0005	Eccesso di memoria.
0x0601 0000	Accesso non supportato a un oggetto.
0x0601 0001	Tentativo di leggere un oggetto a sola scrittura.
0x0601 0002	Tentativo di scrivere un oggetto a sola lettura.
0x0602 0000	L'oggetto non esiste nel dizionario oggetti.
0x0604 0041	Impossibile mappare l'oggetto sul PDO.
0x0604 0042	Il numero e la lunghezza degli oggetti da mappare supererebbero la lunghezza del PDO.
0x0604 0043	Incompatibilità dei parametri.
0x0604 0047	Incompatibilità interna al dispositivo.
0x0606 0000	Accesso non riuscito a causa di un errore hardware.
0x0607 0010	Il tipo di dati non corrisponde, la lunghezza del parametro di servizio non corrisponde.

© 2025 Robox SpA

SDO error code	Descrizione
0x0607 0012	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo alto.
0x0607 0013	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo basso.
0x0609 0011	Il sottoindice non esiste.
0x0609 0030	Superamento del campo di valori del parametro (solo per accesso in scrittura).
0x0609 0031	Valore del parametro scritto troppo alto.
0x0609 0032	Valore del parametro scritto troppo basso.
0x0609 0036	Il valore massimo è inferiore al valore minimo.
0x060A 0023	Risorsa non disponibile: connessione SDO.
0x0800 0000	Errore generale.
0x0800 0020	I dati non possono essere trasferiti o archiviati nell'applicazione.
0x0800 0021	I dati non possono essere trasferiti o archiviati nell'applicazione a causa del controllo locale.
0x0800 0022	I dati non possono essere trasferiti o archiviati nell'applicazione a causa dello stato attuale del dispositivo.
0x0800 0023	La generazione dinamica del dizionario oggetti non riesce o non è presente alcun dizionario oggetti (ad esempio, il dizionario oggetti viene generato da un file e la generazione non riesce a causa di un errore di file).
0x0800 0024	Nessun dato disponibile.
Altri valori	Codice non valido - non è presente CiA DS301 v4.1.

Codici di errore SDO SoE

SDO error code	Descrizione
0x0009	Accesso non valido all'elemento 0.
0x1001	Non esiste.
0x1009	Accesso non valido all'elemento 1.
0x2001	Nome inesistente.
0x2002	Nome sottodimensionato in trasmissione.

© 2025 Robox SpA

SDO error code	Descrizione
0x2003	Nome sovradimensionato in trasmissione.
0x2004	Nome non si può cambiare.
0x2005	Nome attualmente protetto da scrittura.
0x3002	Attributo sottodimensionato in trasmissione.
0x3003	Attributo sovradimensionato in trasmissione.
0x3004	Attributo non si può cambiare.
0x3005	Attributo attualmente protetto da scrittura.
0x4001	Unità inesistente.
0x4002	Unità sottodimensionato in trasmissione.
0x4003	Unità sovradimensionato in trasmissione.
0x4004	Unità non si può cambiare.
0x4005	Unità attualmente protetto da scrittura.
0x5001	Valore minimo in input inesistente.
0x5002	Valore minimo in input sottodimensionato in trasmissione.
0x5003	Valore minimo in input sovradimensionato in trasmissione.
0x5004	Valore minimo in input non si può cambiare.
0x5005	Valore minimo in input attualmente protetto da scrittura.
0x6001	Valore massimo in input inesistente.
0x6002	Valore massimo in input sottodimensionato in trasmissione.
0x6003	Valore massimo in input sovradimensionato in trasmissione.
0x6004	Valore massimo in input non si può cambiare.
0x6005	Valore massimo in input attualmente protetto da scrittura.
0x7001	Elemento inesistente.
0x7002	Elemento sottodimensionato in trasmissione.
0x7003	Elemento sovradimensionato in trasmissione.
0x7004	Elemento non si può cambiare.
0x7005	Elemento attualmente protetto da scrittura.

© 2025 Robox SpA

SDO error code	Descrizione
0x7006	Elemento inferiore al limite di *minimum input value.*
0x7007	Elemento superiore al limite di *maximum input value.*
0x7008	Elemento non corretto.
0x7009	Elemento protetto da password.
0x700A	Elemento temporaneamente non modificabile (AT o MDT).
0x700B	Indirettamente non valido.
0x700C	Elemento temporaneamente non modificabile (parametro o modo operativo).
0x7010	Comando già attivo.
0x7011	Comando non interrompibile.
0x7012	Comando non disponibile (in questa fase).
0x7013	Comando non disponibile (parametro non valido).

Codici di errore Modbus

Codici di errore Modbus

Modbus error code	Valore	Descrizione
MBC_WAIT	1	Comando in attesa di risposta
MBC_OK	0	Comando eseguito
MBC_INVAL	-1	Argomenti non validi
MBC_BUSY	-2	Risorsa occupata
MBC_PERM	-3	Permesso negato
MBC_CONN	-4	Connessione non riuscita
MBC_NOBUFS	-5	Buffer esauriti
MBC_NOMEM	-6	Memoria insufficiente
MBC_TX_ERR	-7	Errore di trasmissione
MBC_RX_ERR	-8	Errore di ricezione
MBC_TIMEOUT	-9	Timeout comando bloccante

© 2025 Robox SpA

Modbus error code	Valore	Descrizione
MBC_EXC_CODE_1	-10	Risposta con eccezione 1 (function code non supportato dal server)
MBC_EXC_CODE_2	-11	Risposta con eccezione 2 (address non valido)
MBC_EXC_CODE_3	-12	Risposta con eccezione 3 (quantity o valore non valido)
MBC_EXC_CODE_4	-13	Risposta con eccezione 4 (operazione fallita)
MBC_WRONG_ANSWER	-14	Risposta non valida
MBC_INVAL_MBC	-15	Id MBC non valido

Tipi di ciclo di zero

Tipologie di ciclo di zero:

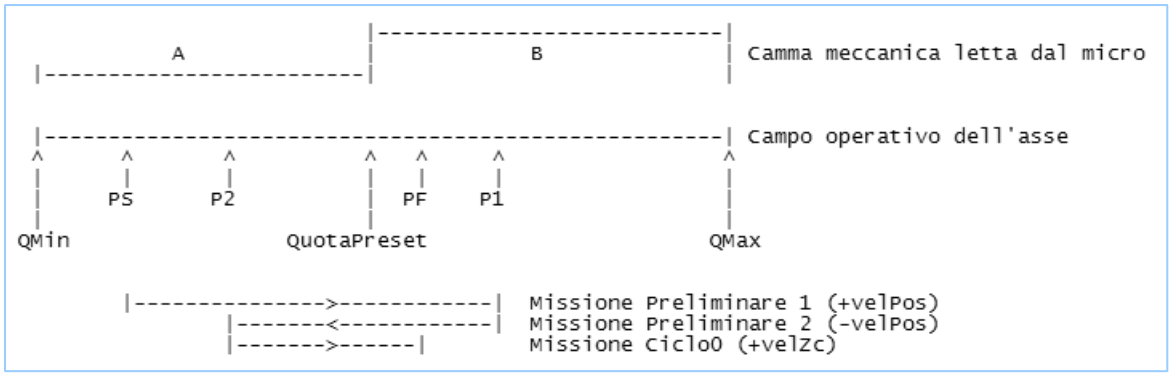
- Asse lineare: ciclo di zero a micro 69
- Asse lineare: ciclo di zero a micro + index 74
- Asse lineare: ciclo di zero a paracarro 79
- Asse lineare: ciclo di zero a paracarro + index 82
- Asse rotativo monodirezionale: ciclo0 a micro + index con ricerca preliminare della variazione del micro 88
- Asse rotativo monodirezionale: ciclo0 a micro + index SENZA ricerca preliminare della variazione del micro 94
- Asse rotativo monodirezionale: ciclo0 solo a micro con ricerca preliminare della variazione del micro 99
- Asse rotativo monodirezionale: ciclo0 solo a micro senza ricerca preliminare della variazione del micro 104

Asse lineare: ciclo di zero a micro

Validità: **Rule**

Caso 1: l'asse parte dal punto PS in zona A

© 2025 Robox SpA



Fasi:

a) Viene letto lo stato del micro per poter definire il segno della velocità di movimentazione.
Assumendo il parametro zcFlag.0 = 0, dato che se l'asse si trova in zona A la velocità contenuta nel parametro velPos sarà assunta positiva.

b) Missione Preliminare 1: l'asse muove alla velocità (+velPos) fino a riscontrare il fronte a salire del micro, quindi decelera e si arresta in posizione P1.

c) Missione Preliminare 2: l'asse muove alla velocità (-velPos) fino a riscontrare il fronte a scendere del micro, quindi decelera e si arresta in posizione P2.

d) Missione finale (CICLO0): si esegue il movimento finale con una corsa massima pari al valore del parametro spSch, con la nuova velocità imposta (+velZc).
Nel momento in cui viene trovato il fronte a salire del micro, la missione viene fermata e l'asse si arresta in PF.

e) Si assegna come nuova quota reale il valore ottenuto sommando al parametro Preset il delta di quote (PF-quota letta sul fronte di salita del micro)

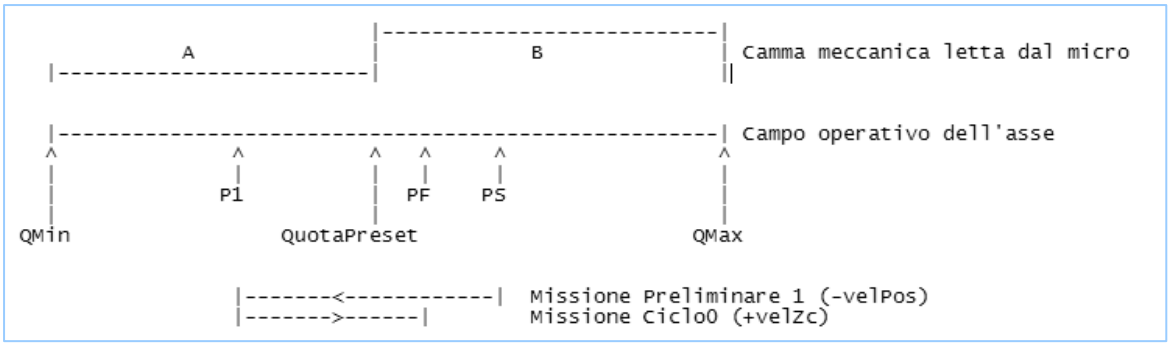
f) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse

Legenda:

- *A, zona in cui il micro legge 0*
- *B, zona in cui il micro legge 1*
- *PS, punto di start iniziale per la mva_zc511()*
- *PF, punto di arresto finale per la mva_zc511()*
- *P1, punto intermedio 1*
- *P2, punto intermedio 2*
- *Qmin, quota minima dell'asse*
- *Qmax, quota massima dell'asse*

Caso 2: l'asse parte dal punto PS in zona B

© 2025 Robox SpA



Fasi:

a) Viene letto lo stato del micro per poter definire il segno della velocità di movimentazione.
Assumendo il parametro zcFlag.0 = 0, dato che se l'asse si trova in zona B la velocità contenuta nel parametro velPos sarà assunta negativa.

b) Missione Preliminare 1: l'asse muove alla velocità (-velPos) fino a riscontrare il fronte a scendere del micro, quindi decelera e si arresta in posizione P1.

c) Missione finale (CICLO0): si esegue il movimento finale con una corsa massima pari al valore del parametro spSch, con la nuova velocità imposta (+velZc).
Nel momento in cui viene trovato il fronte a salire del micro, la missione viene fermata e l'asse si arresta in PF.

d) Si assegna come nuova quota reale il valore ottenuto sommando al parametro Preset il delta di quote (PF-quota letta sul fronte di salita del micro)

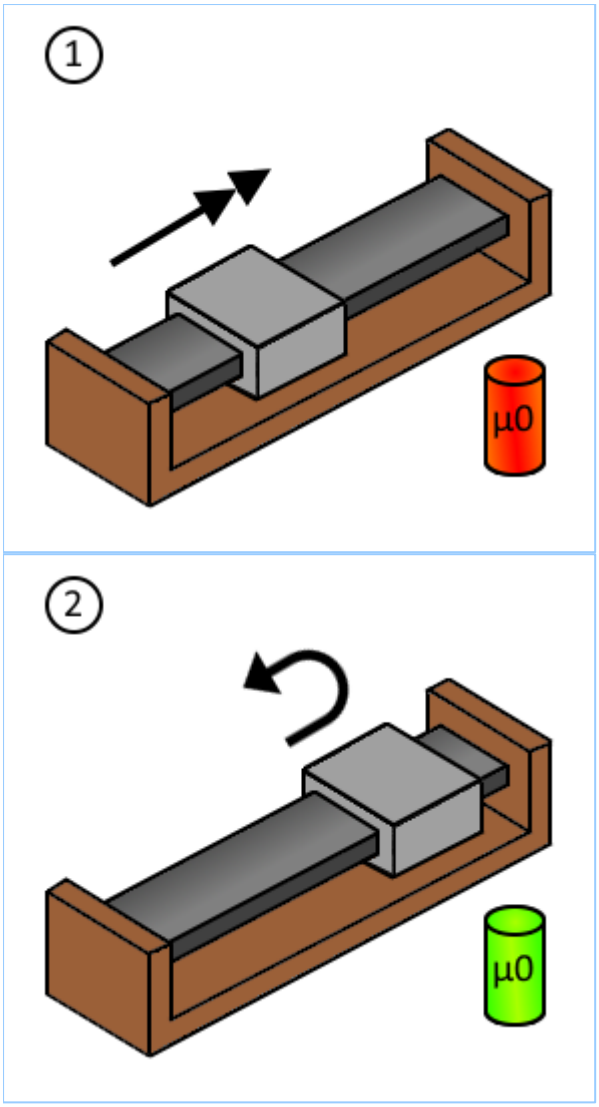
e) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse

Leggenda:

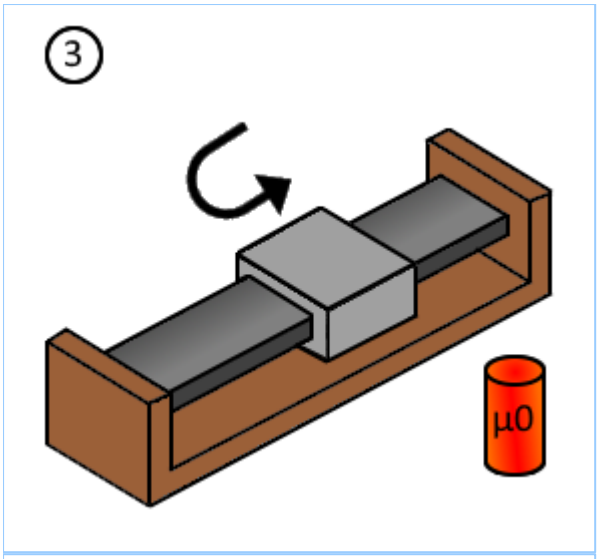
- *A, zona in cui il micro legge 0*
- *B, zona in cui il micro legge 1*
- *PS, punto di start iniziale per la mva_zc511()*
- *PF, punto di arresto finale per la mva_zc511()*
- *P1, punto intermedio 1*
- *Qmin, quota minima dell'asse*
- *Qmax, quota massima dell'asse*

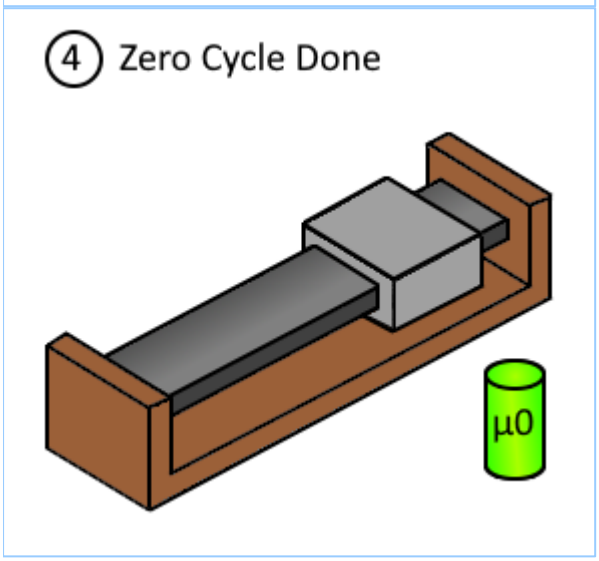
Esempio grafico

© 2025 Robox SpA



© 2025 Robox SpA





Esempi

Esempio di utilizzo:

© 2025 Robox SpA

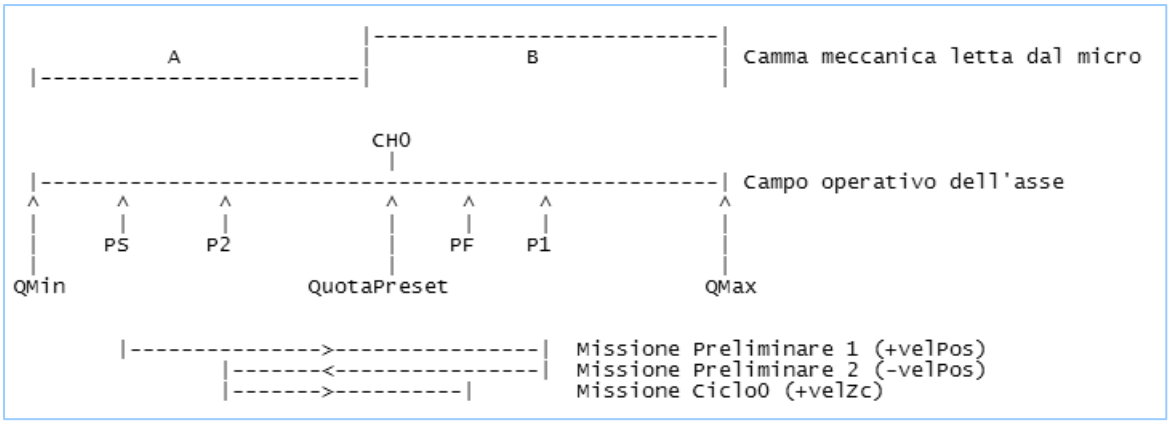
```
INT esito
STRU_MVZC 892 zc
INT Flag
...
rule 123
    axes 1
    motion
        ; First time initialization
        if (first_time 346())
            Flag = true
            zc.nAx    = 1 ; Axis number
            zc.preset = 0 ; Preselection quotas
            zc.velPos = 10.0 ; Epos(nAx) threshold in units to conside
                             ; (if you also have cv(nAx)<(zc.velZC/100
            zc.velZC  = rVelc0 ; Speed for final handling CYCLE0 mva_z
            zc.acc    = rMaxAcc * 0.5 ; Acceleration to be used in all
            zc.jerk   = rMaxAcc * 10 ; Derivative of acceleration
            zc.spSCh  = 4096 ; Maximum space of the final movement CYC
            zc.filter = 0 ; If drive in canopen reappears % torque red
            zc.zcFlag.0 = 1 ; Positive speed
            zc.zcFlag.1 = 0 ; You select the falling edge of the micro
            zc.zcFlag.2 = 0 ; Disable full precision
            zc.zcFlag.3 = 0 ; Enable use of the micro
            zc.zcFlag.4 = 0 ; The execution of the movement is bidirec
            zc.zcFlag.5 = 0 ; The space that can be traversed during p
            zc.zcFlag.6 = 0 ; Not used
            zc.zcFlag.7 = 0 ; Disable search of the end of stroke
            zc.zcFlag.8 = 0 ; Disable 0 cycle ON_POSITION
            zc.zcFlag.9 = 0 ; Disable positioning on preset elevation
            zc.zcFlag.10 = 0 ; Disable repositioning within software e
        endif
        ; Call the move function
        esito = mva_zc 511(Flag, zc)
    end_motion
end_rule
```

Asse lineare: ciclo di zero a micro + index

Validità: **Rule**

Caso 1: l'asse parte dal punto PS in zona A

© 2025 Robox SpA



Fasi:

a) Viene letto lo stato del micro per poter definire il segno della velocità di movimentazione.
Assumendo il parametro zcFlag.0 = 0, dato che l'asse si trova in zona A la velocità contenuta nel parametro velPos sarà assunta positiva.

b) Missione Preliminare 1: l'asse muove alla velocità (+velPos) fino a riscontrare il fronte a salire del micro, quindi decelera e si arresta in posizione P1.

c) Missione Preliminare 2: l'asse muove alla velocità (-velPos) fino a riscontrare il fronte a scendere del micro, quindi decelera e si arresta in posizione P2.

d) Missione finale (CICLO0): si esegue il movimento finale con una corsa massima pari al valore del parametro spSch, con la nuova velocità imposta (+velZc).
Nel momento in cui viene trovato il primo canale di 0 encoder col micro in 1, la missione viene fermata e l'asse si arresta in PF.

e) Si assegna come nuova quota reale il valore ottenuto sommando al parametro Preset il delta di quote (PF-quota letta sul primo CH0 encoder col micro in 1).

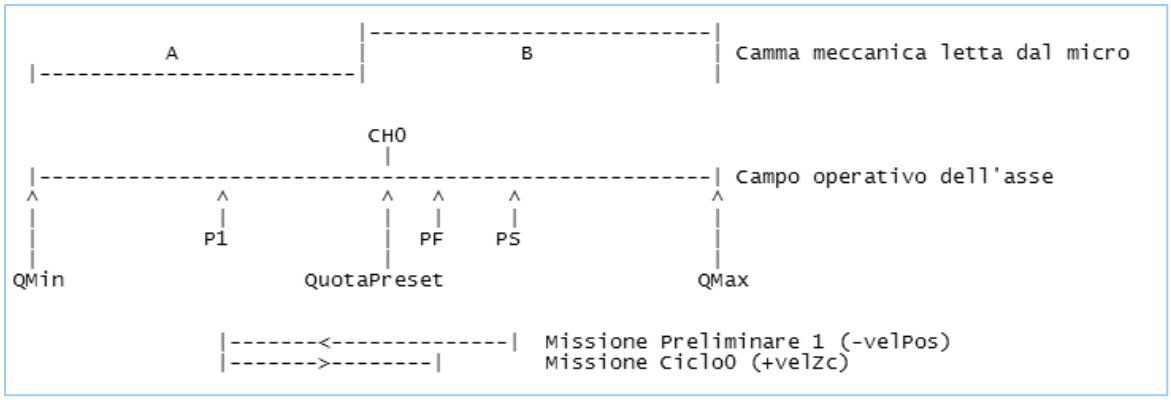
f) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse

Legenda:

- *A, zona in cui il micro legge 0*
- *B, zona in cui il micro legge 1*
- *PS, punto di start iniziale per la mva_zc511()*
- *PF, punto di arresto finale per la mva_zc511()*
- *P1, punto intermedio 1*
- *P2, punto intermedio 2*
- *Qmin, quota minima dell'asse (min_str)*
- *Qmax, quota massima dell'asse (max_str)*
- *CH0, canale di 0 dell'encoder incrementale*

Caso 2: l'asse parte dal punto PS in zona B

© 2025 Robox SpA



Fasi:

a) Viene letto lo stato del micro per poter definire il segno della velocità di movimentazione.
Assumendo il parametro zcFlag.0 = 0, dato che se l'asse si trova in zona B la velocità contenuta nel parametro velPos sarà assunta negativa.

b) Missione Preliminare 1: L'asse muove alla velocità (-velPos) fino a riscontrare il fronte a scendere del micro, quindi decelera e si arresta in posizione P1.

c) Fase assente, lasciata solo per mostrare la differenza con il caso precedente.

d) Missione finale (CICLO0): Si esegue il movimento finale con una corsa massima pari al valore del parametro spSch, con la nuova velocità imposta (+velZc).
Nel momento in cui viene trovato il primo canale di 0 encoder col micro in 1, la missione viene fermata e l'asse si arresta in PF.

e) Si assegna come nuova quota reale il valore ottenuto sommando al parametro Preset il delta di quote (PF-quota letta sul primo CH0 encoder col micro in 1).

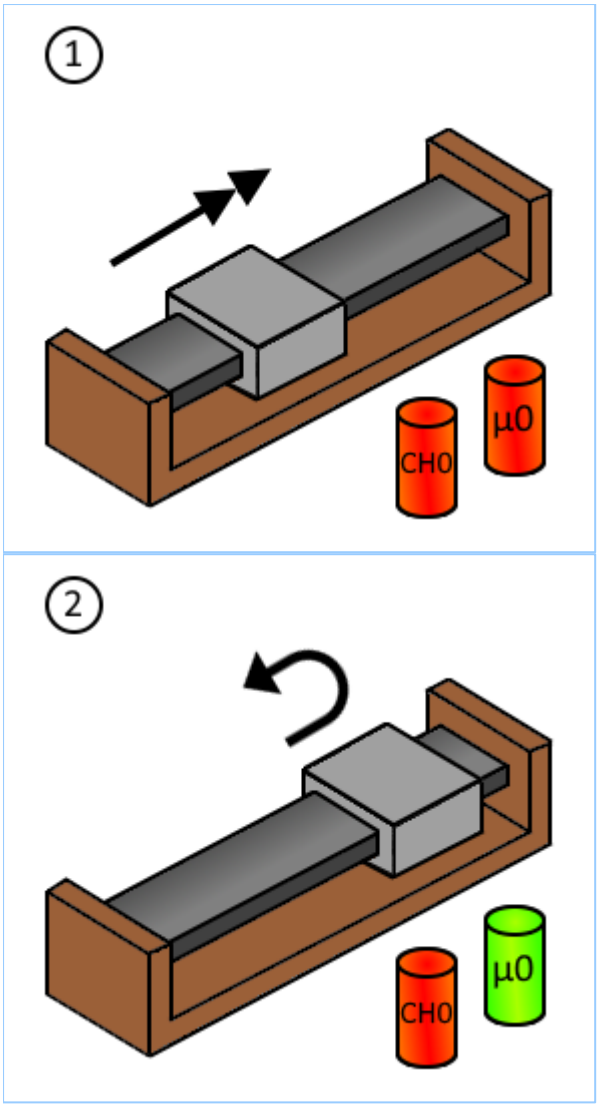
f) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse.

Leggenda:

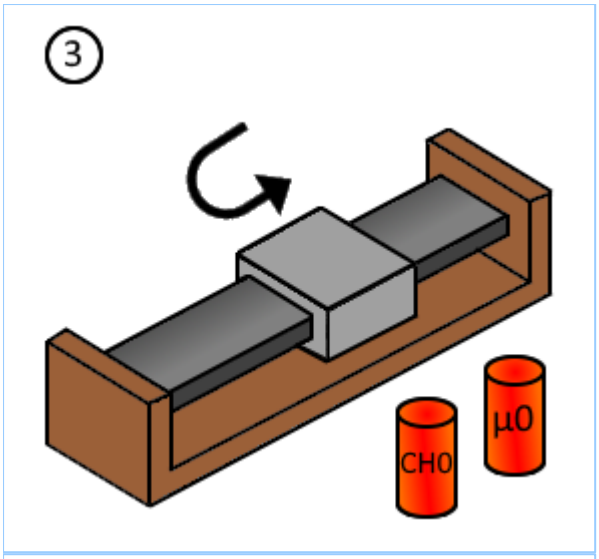
- *A, zona in cui il micro legge 0*
- *B, zona in cui il micro legge 1*
- *PS, punto di start iniziale per la mva_zc511()*
- *PF, punto di arresto finale per la mva_zc511()*
- *P1, punto intermedio 1*
- *Qmin, quota minima dell'asse*
- *Qmax, quota massima dell'asse*
- *CH0, canale di 0 dell'encoder incrementale*

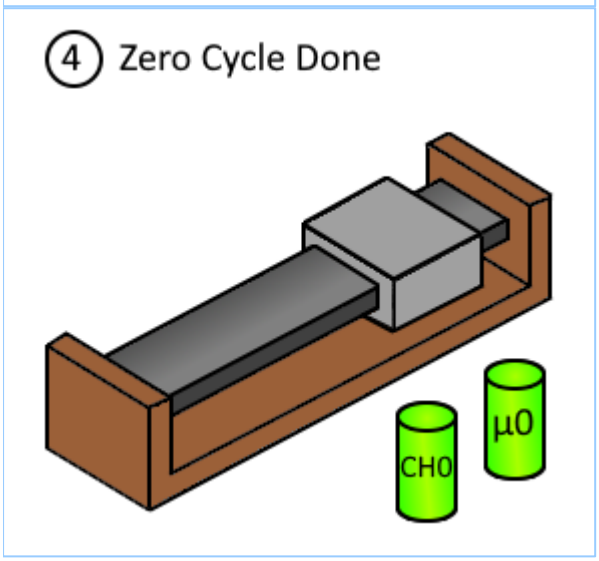
Esempio grafico

© 2025 Robox SpA



© 2025 Robox SpA





Esempi

Esempio di utilizzo:

```
INT esito
STRU_MVZC 892  zc
INT Flag
...
rule 123
    axes 1
    motion
        ; First time initialization
        if (first_time 346())
            Flag = true
            zc.nAx    = 1 ; Axis number
            zc.preset = 0 ; Preselection quotas
```

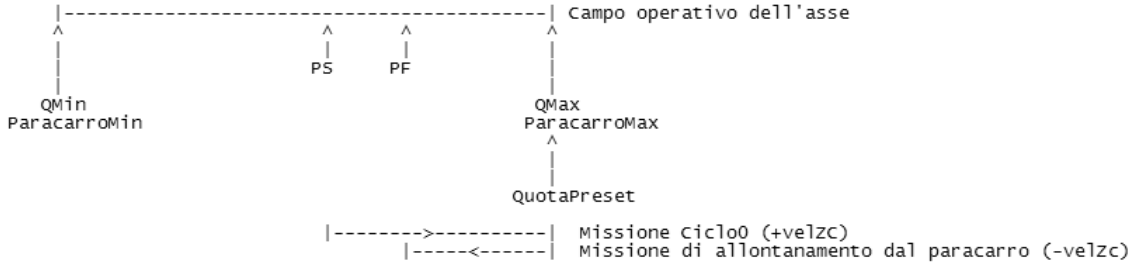
© 2025 Robox SpA

```
            zc.velPos = rVelc0 ; Speed for preliminary movements mva_z
            zc.velZC  = rVelc0 * 0.5 ; Speed for final movement CYCLE0
            zc.acc    = rMaxAcc * 0.5 ; Acceleration to be used in all
            zc.jerk   = rMaxAcc * 10 ; Derivative of acceleration
            zc.spSCh  = 20000 ; Maximum space of the final movement CY
            zc.filter = 0 ; Time (sec) of micro signal filtering
            zc.zcFlag.0 = 0  ; Negative speed when the micro is in 1
            zc.zcFlag.1 = 1  ; Wants micro in 1 when looking for CH0
            zc.zcFlag.2 = 1  ; Enable full-precision (in this case CH0
            zc.zcFlag.3 = 0  ; Enable use of micro
            zc.zcFlag.4 = 0  ; Bidirectional search (since the axis is
            zc.zcFlag.5 = 0  ; Preliminary unrestricted excursion miss
            zc.zcFlag.6 = 0  ; Not used (because two-way search)
            zc.zcFlag.7 = 0  ; Disable end of troke search (because mi
            zc.zcFlag.8 = 0  ; Disable zero cycle ON_POSITION
            zc.zcFlag.9 = 0  ; Disable positioning on preset elevation
            zc.zcFlag.10 = 0 ; Disable repositioning within software l
        endif

        ; Call the move function
        esito = mva_zc[511](Flag, zc)
    end_motion
end_rule
```

Asse lineare: ciclo di zero a paracarro

Validità: **Rule**

Caso unico, l'asse parte dal punto PS



Fasi:

a) Viene disabilitato l'allarme servo (sam) dell'asse.

b) Missione di CICLO0 (ricerca paracarro): l'asse muove alla velocità (+velZC) fino a riscontrare le condizioni "abs(cv) < zc.velZC/1000.0" e "abs(ip-cp) > zc.velPos". Nota: da RTE v34.24.10, se zcFlag.13 (0x2000) della STRU_MVZC[892] è a 1, la

Legenda:

- *PS, punto di start iniziale per la mva_zc()*
- *PF, punto di arresto finale per la mva_zc()*
- *Qmin, quota minima dell'asse (min_str)*
- *Qmax, quota massima dell'asse (max_str)*

© 2025 Robox SpA

condizione "abs(ip-cp) > zc.velPos" viene disabilitata.

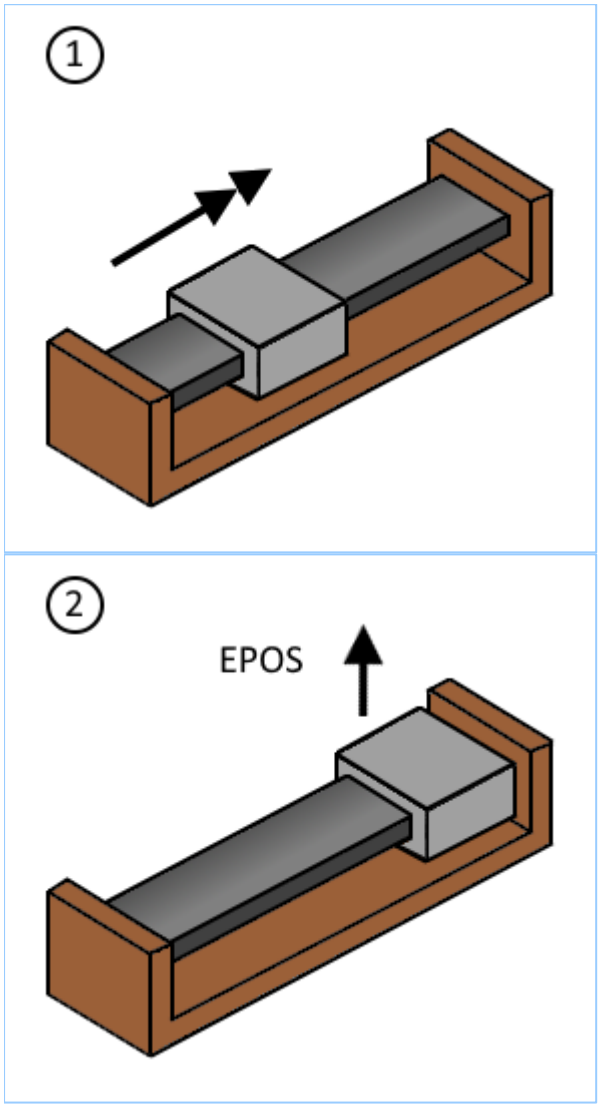
c) Viene reimpostato il valore precedente sulla maschera SAM relativo all'asse.

d) Missione di allontanamento dal paracarro: l'asse muove alla velocità (-velZC) fino a percorrere uno spazio almeno pari a "zc.spSCh" e si arresta in posizione PF.

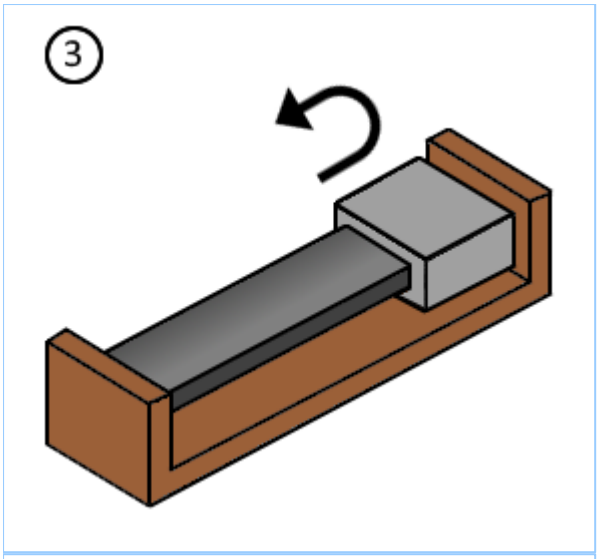
e) Si assegna come nuova quota reale il valore ottenuto sottraendo al parametro Preset (quota su paracarro) il valore di quota del punto PF.

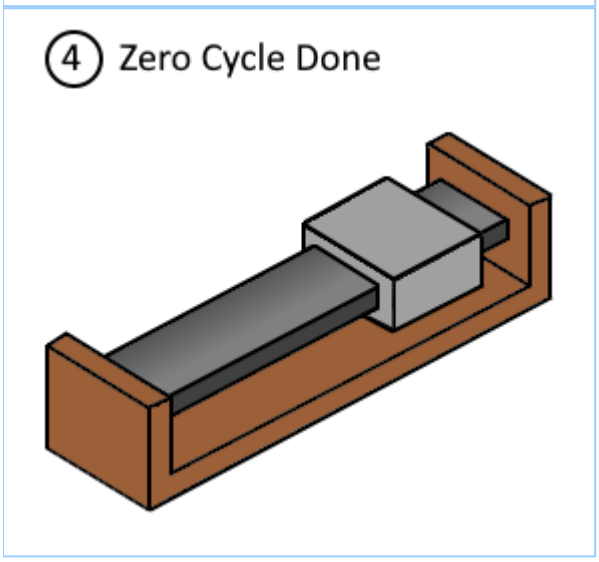
f) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse.

Esempio grafico



© 2025 Robox SpA





Esempi

Esempio di utilizzo:

```
INT esito
STRU_MVZC 892 zc
INT Flag
...
rule 123
    axes 1
    motion
        ; First time initialization
        if (first_time 346())
            Flag = true
            zc.nAx    = 1 ; Axis number
            zc.preset = 0 ; Preselection quotas
```

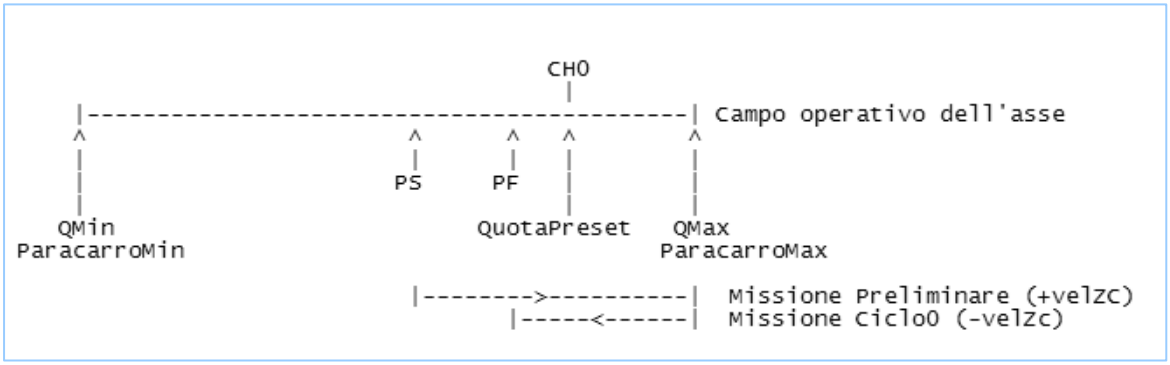
© 2025 Robox SpA

```
            zc.velPos = 10.0 ; Threshold epos(nAx) in units x consider
                             ; (if you also have cv(nAx)<(zc.velZC/100
            zc.velZC  = rVelc0 ; Speed for final movement CYCLE0 mva_z
            zc.acc    = rMaxAcc * 0.5 ; Acceleration to be used in all
            zc.jerk   = rMaxAcc * 10 ; Derivative of acceleration
            zc.spSCh  = 20.0 ; Minimal space x mission away from the p
            zc.filter = 0 ; If drive in canopen represents % torque re
            zc.zcFlag.0 = 1  ; Positive speed x paracart search
            zc.zcFlag.1 = 0  ; Torque reduction %, only if drive type
            zc.zcFlag.2 = 0  ; Disabilita full-precision (ch0 enc)
            zc.zcFlag.3 = 1  ; Disable use of micro (because paracar r
            zc.zcFlag.4 = 1  ; Not used
            zc.zcFlag.5 = 1  ; Preliminary missions of limited excursi
                             ; (equal to max_str(nAx)-min_str(nAx))
            zc.zcFlag.6 = 0  ; Not used
            zc.zcFlag.7 = 1  ; Enabled end of stroke search
            zc.zcFlag.8 = 0  ; Disable zero cycle ON_POSITION
            zc.zcFlag.9 = 0  ; Disable positioning on preset elevation
            zc.zcFlag.10 = 0 ; Disable repositioning within software l
        endif

        ; Call the move function
        esito = mva_zc(Flag, zc)
    end_motion
end_rule
```

Asse lineare: ciclo di zero a paracarro + index

Validità: **Rule**

Caso unico: l'asse parte dal punto PS



Fasi:

a) Viene disabilitato l'allarme servo (sam) dell'asse.

b) Missione preliminare (ricerca paracarro): l'asse muove alla velocità (+velZC) fino a

Legenda:

- *PS, punto di start iniziale per la mva_zc()*
- *PF, punto di arresto finale per la mva_zc()*
- *Qmin, quota minima dell'asse (min_str)*

© 2025 Robox SpA

riscontrare le condizioni "abs(cv) < zc.velZC/1000.0" e "abs(ip-cp) > zc.velPos".
Nota: Da RTE v34.24.10, se zcFlag.13 (0x2000) della STRU_MVZC 892 è a 1, la condizione "abs(ip-cp) > zc.velPos" viene disabilitata.

c) Viene reimpostato il valore precedente sulla maschera SAM relativo all'asse.

d) Missione finale (CICLO0): l'asse muove alla velocità (-velZC) fino a riscontrare il CH0 dell'encoder, quindi decelera e si arresta in posizione PF.

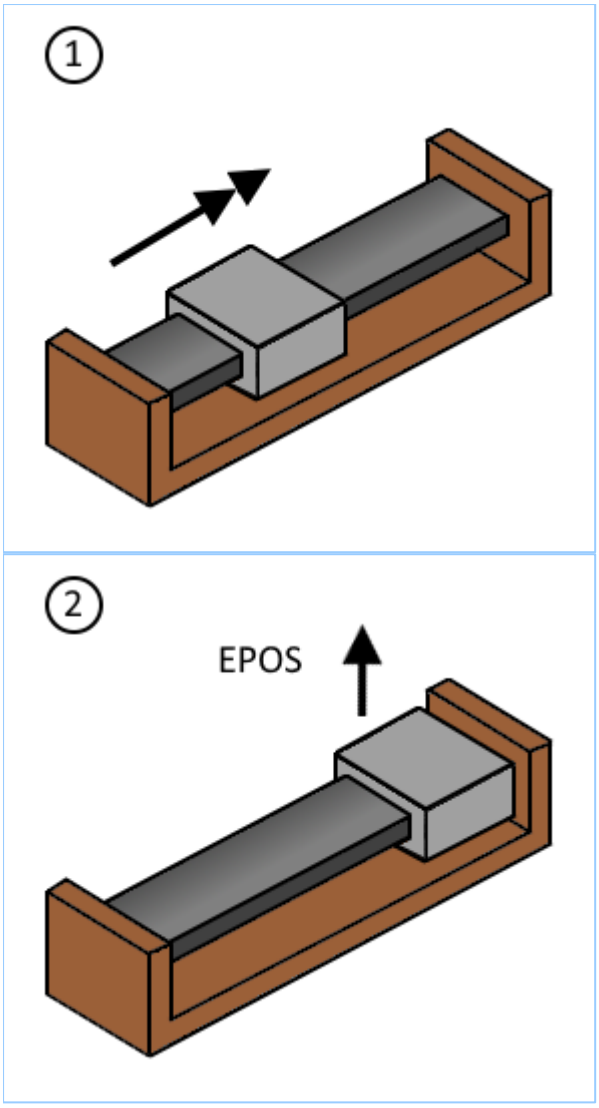
e) Si assegna come nuova quota reale il valore ottenuto sommando al parametro Preset il delta di quote (PF-quota letta sul primo CH0).

f) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse.

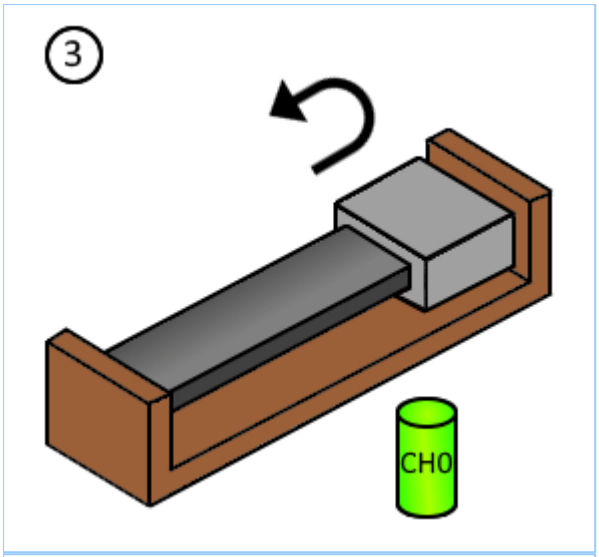
Esempio grafico

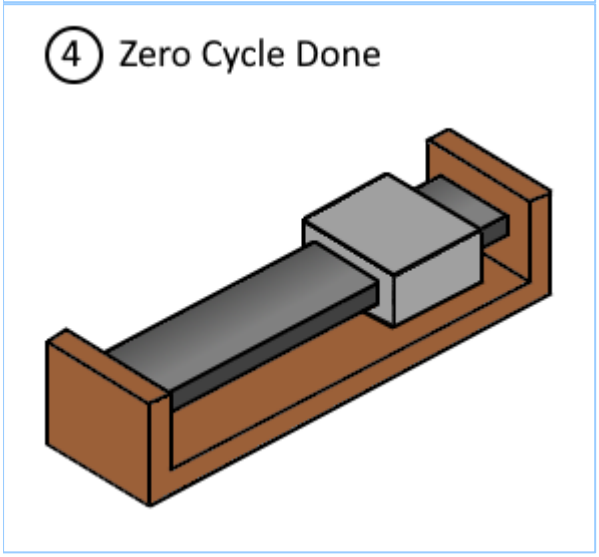
- *Qmax, quota massima dell'asse (max_str)*
- *CH0, canale di 0 dell'encoder incrementale*

© 2025 Robox SpA



© 2025 Robox SpA





Esempi

Esempio di utilizzo:

```
INT esito
STRU_MVZC 892 zc
INT Flag
...
rule 123
    axes 1
    motion
        ; First time initialization
        if (first_time 346())
            Flag = true
            zc.nAx    = 1 ; Axis number
            zc.preset = 0 ; Preselection quotas
```

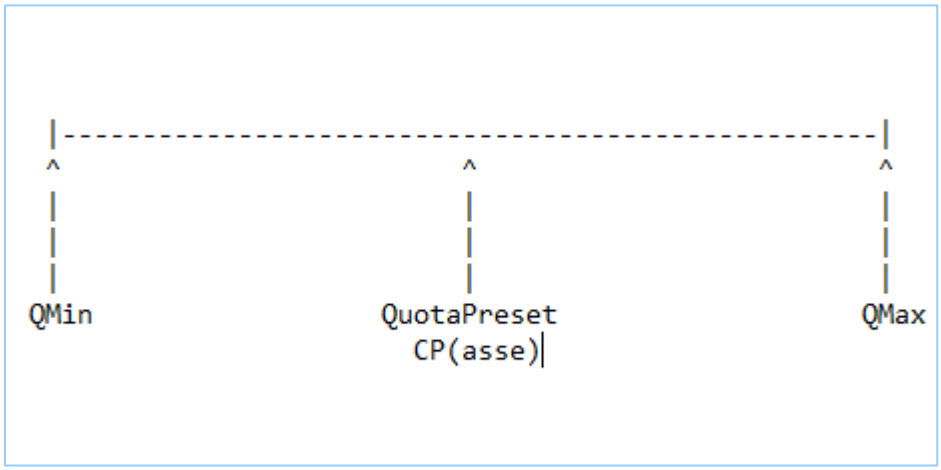
© 2025 Robox SpA

```
            zc.velPos = 10.0 ; Threshold epos(nAx) in units to conside
                             ; (if you also have cv(nAx)<(zc.velZC/100
            zc.velZC  = rVelc0 ; Speed for final movement CYCLE0 mva_z 511
            zc.acc    = rMaxAcc * 0.5 ; Acceleration to be used in all
            zc.jerk   = rMaxAcc * 10 ; Derivative of acceleration
            zc.spSCh  = 4096 ; Maximum space of the final movement CYC
            zc.filter = 0 ; If drive in fieldbus reappears % torque re
            zc.zcFlag.0 = 1  ; Positive speed x paracart search
            zc.zcFlag.1 = 0  ; Not used
            zc.zcFlag.2 = 1  ; Abilita full-precision (ch0 enc)
            zc.zcFlag.3 = 1  ; Disable use of micro (because end of st
            zc.zcFlag.4 = 1  ; Not used
            zc.zcFlag.5 = 1  ; Preliminary missions of limited excursi
                             ; (equal to max_str(nAx)-min_str(nAx))
            zc.zcFlag.6 = 0  ; Not used
            zc.zcFlag.7 = 1  ; Enable end of stroke search
            zc.zcFlag.8 = 0  ; Disable zero cycle ON_POSITION
            zc.zcFlag.9 = 0  ; Disable positioning on preset elevation
            zc.zcFlag.10 = 0 ; Disable repositioning within software l
        endif

        ; Call the move function
        esito = mva_zc 511(Flag, zc)
    end_motion
end_rule
```

Asse lineare: ciclo di zero sulla posizione

Validità: **Rule**

Caso unico: la quota di preselezione coincide con la quota reale



Fasi:

a) La posizione attuale dell'asse diventa il valore di Preset indicato

Legenda:

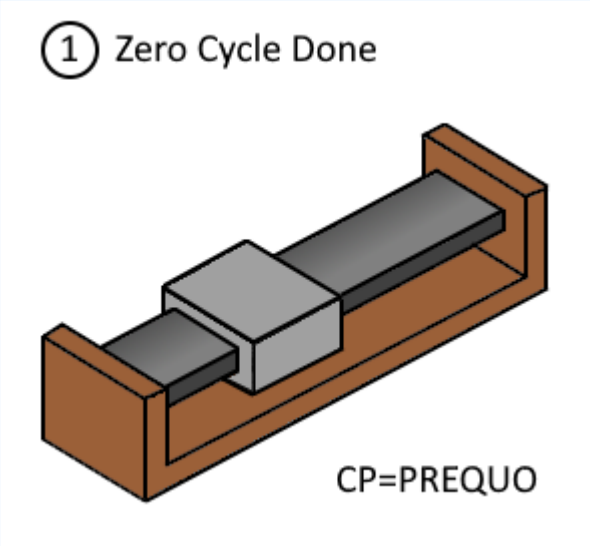
- *Qmin, quota minima dell'asse*

© 2025 Robox SpA

b) RTE provvede ad aggiornare la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse

- *Qmax, quota massima dell'asse*

Esempio grafico



Esempi

Esempio di utilizzo:

```
INT esito
STRU_MVZC 892 zc
INT Flag
...
rule 123
    axes 1
    motion
        ; First time initialization
        if (first_time 346())
            Flag = true
            zc.nAx    = 1 ; Axis number
            ...
            zc.zcFlag.0 = 1 ; Positive speed
            zc.zcFlag.1 = 0 ; Not used
            zc.zcFlag.2 = 1 ; Enable full precision
            zc.zcFlag.3 = 1 ; Disable use of the micro
            zc.zcFlag.4 = 1 ; The execution of the movement is monodir
            zc.zcFlag.5 = 0 ; Space that can be traversed during preli
            zc.zcFlag.6 = 0 ; Not used
            zc.zcFlag.7 = 0 ; Disable search of the end of stroke
            zc.zcFlag.8 = 1 ; Enable 0 cycle ON_POSITION
            zc.zcFlag.9 = 0 ; Disable positioning on preset elevation
            zc.zcFlag.10 = 0 ; Disable repositioning within software e
```

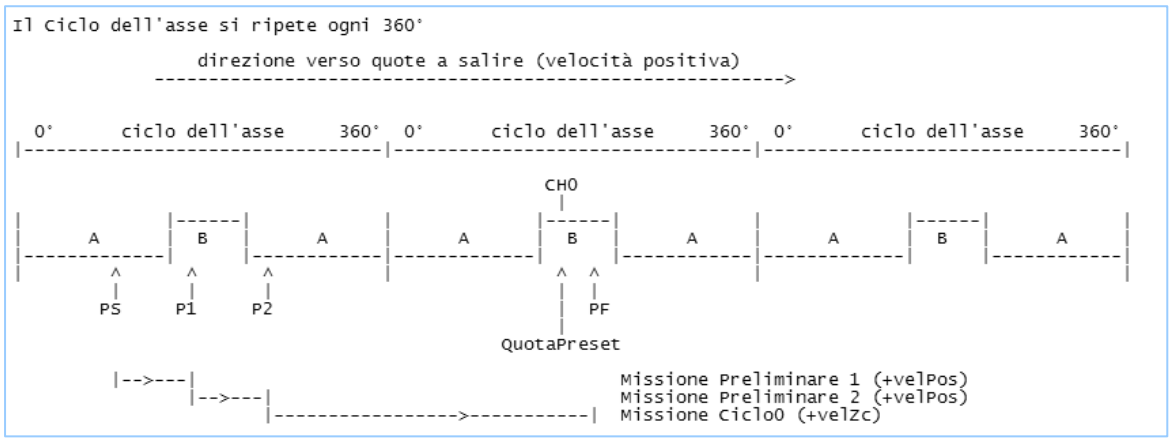
© 2025 Robox SpA

```
            zc.zcFlag.11 = 0 ; Not used
            zc.zcFlag.12 = 0 ; Not used
            zc.zcFlag.13 = 0 ; Not used
        endif

        ; Call the move function
        esito = mva_zc511(Flag, zc)
    end_motion
end_rule
```

Asse rotativo monodirezionale: ciclo0 a micro + index con ricerca preliminare della variazione del micro

Validità: **Rule**

Caso 1: l'asse parte dal punto PS in zona A.



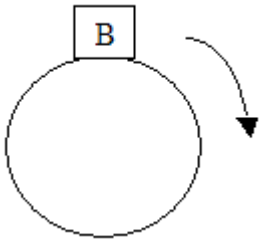
Fasi:

a) Dato che è abilitata ricerca preliminare della variazione del micro.
Missione Preliminare 1: l'asse muove alla velocità (+velPos) fino a riscontrare il primo fronte del micro (in questo caso a salire), quindi decelera e si arresta in posizione P1.

b) Dato che il micro è in 1 e il ciclo0 si completa con micro in 1 serve un'ulteriore missione preliminare.
Missione Preliminare 2: l'asse muove alla velocità (+velPos) fino a riscontrare il fronte a scendere del micro, quindi decelera e si arresta in posizione P2.

Legenda:

- *A, zona in cui il micro legge 0*
- *B, zona in cui il micro legge 1*
- *PS, punto di start iniziale per la mva_zc511()*
- *PF, punto di arresto finale per la mva_zc511()*



© 2025 Robox SpA

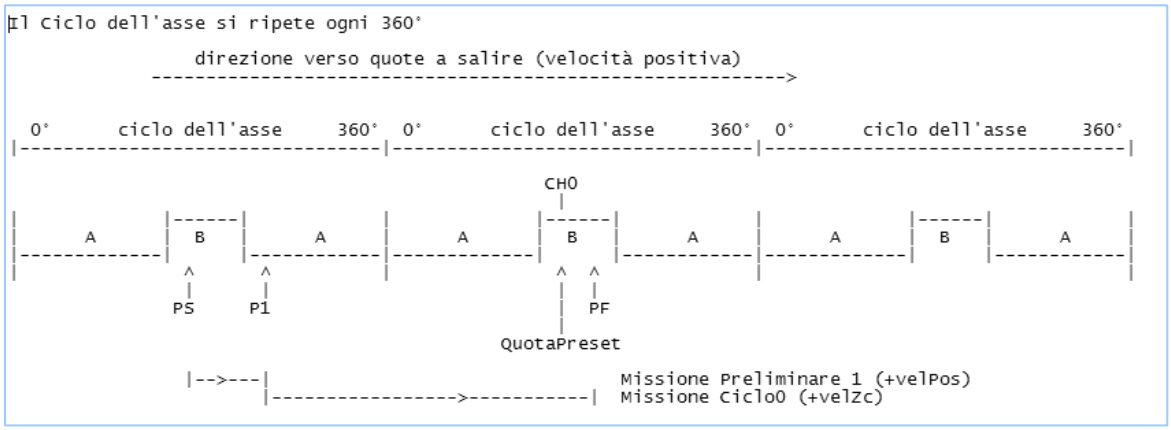
c) Missione finale (CICLO0): si esegue il movimento finale con una corsa massima pari al valore del parametro spSch, con la velocità imposta (+velZc). Nel momento in cui viene trovato il primo canale di 0 encoder col micro in 1, la missione viene fermata e l'asse si arresta in PF.

d) Si assegna come nuova quota reale il valore ottenuto sommando al parametro Preset il delta di quote (PF-quota letta sul primo CH0 encoder col micro in 1).

e) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse.

NOTA: la lunghezza della camma B letta dal micro deve essere superiore ad un giro encoder x garantire di vedere il CH0 col micro in 1.

Caso 2: l'asse parte dal punto PS in zona B.



Fasi:

a) Dato che è abilitata ricerca preliminare della variazione del micro.
Missione Preliminare 1: l'asse muove alla velocità (+velPos) fino a riscontrare il primo fronte del micro (in questo caso a scendere), quindi decelera e si arresta in posizione P1.

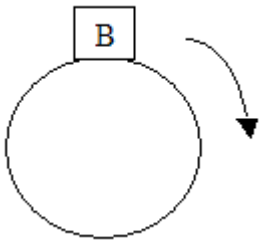
b) Dato che il micro è in 0 e il ciclo0 si completa con micro in 1 non servono ulteriori missioni preliminari.

Leggenda:

- *A, zona in cui il micro legge 0.*
- *B, zona in cui il micro legge 1.*
- *PS, punto di start iniziale per la mva_zc511()*
- *PF, punto di arresto finale per la mva_zc511()*
- *P1, punto intermedio 1*

© 2025 Robox SpA

c) Missione finale (CICLO0): si esegue il movimento finale con una corsa massima pari al valore del parametro spSch, con la velocità imposta (+velZc). Nel momento in cui viene trovato il primo canale di 0 encoder col micro in 1, la missione viene fermata e l'asse si arresta in PF.



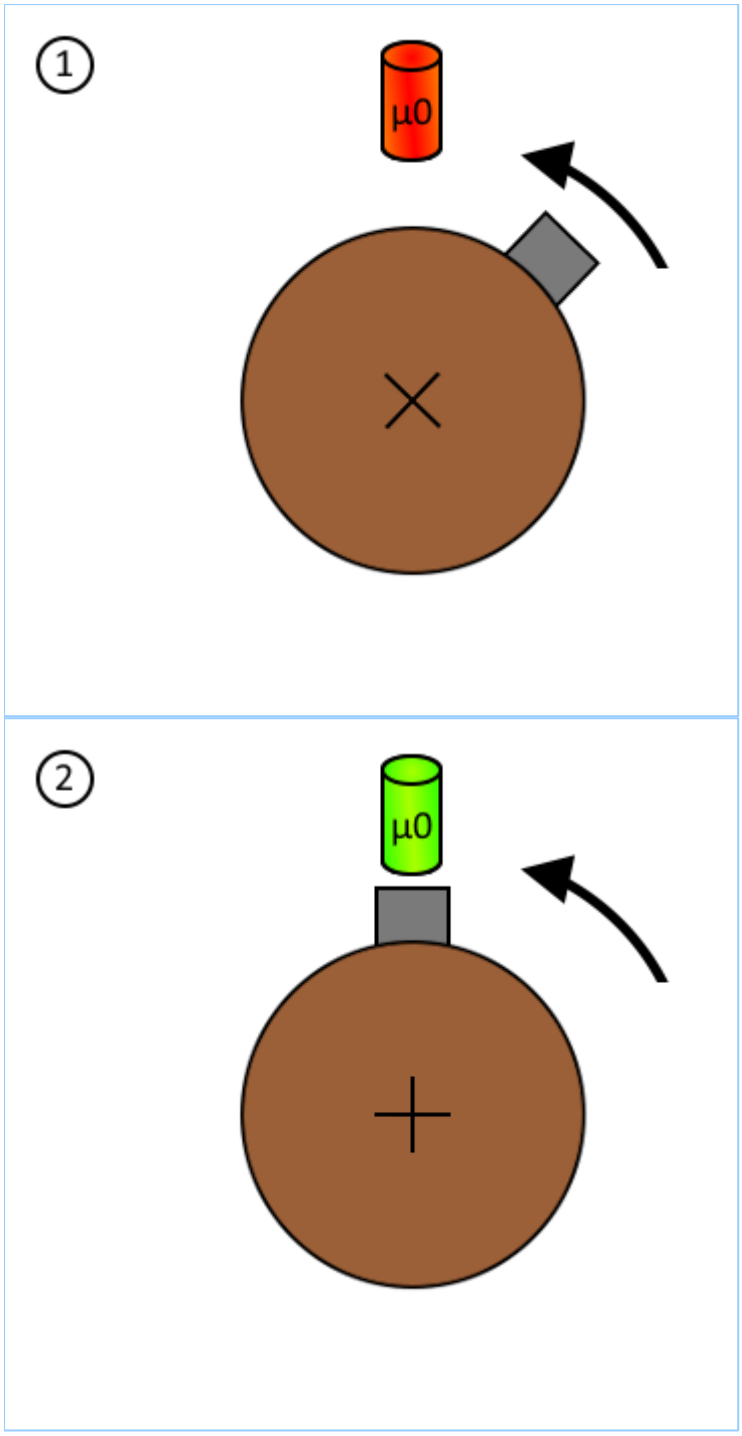
d) Si assegna come nuova quota reale il valore ottenuto sommando al parametro Preset il delta di quote (PF-quota letta sul primo CH0 encoder col micro in 1).

e) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse.

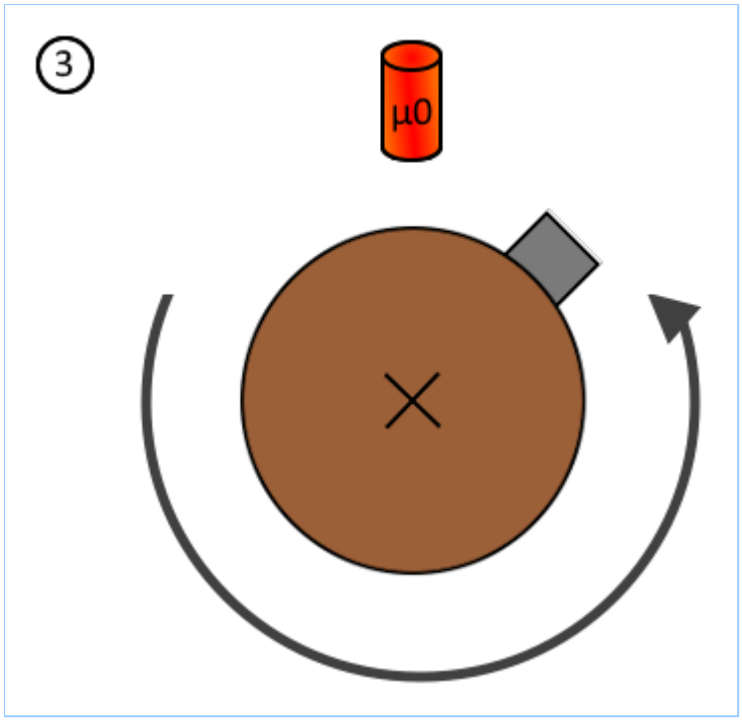
NOTA: la lunghezza della camma B letta dal micro deve essere superiore ad un giro encoder x garantire di vedere il CH0 col micro in 1

Esempio grafico

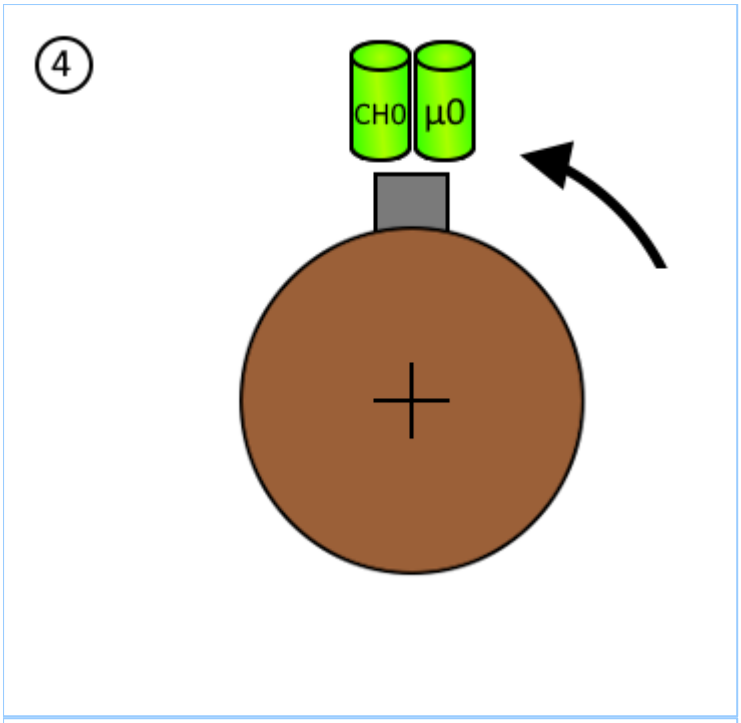
© 2025 Robox SpA

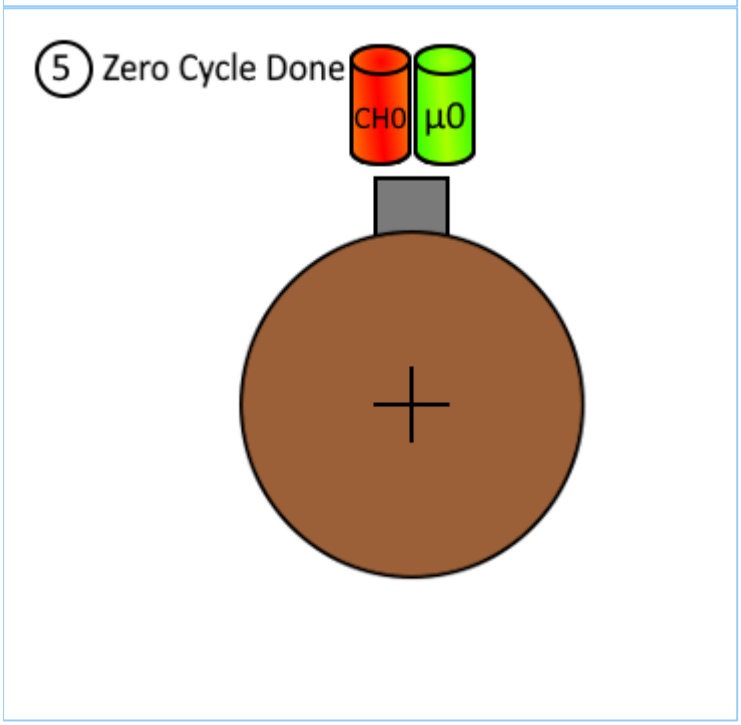


© 2025 Robox SpA



© 2025 Robox SpA





Esempi

Esempio di utilizzo:

```
INT esito
STRU_MVZC 892 zc
```

© 2025 Robox SpA

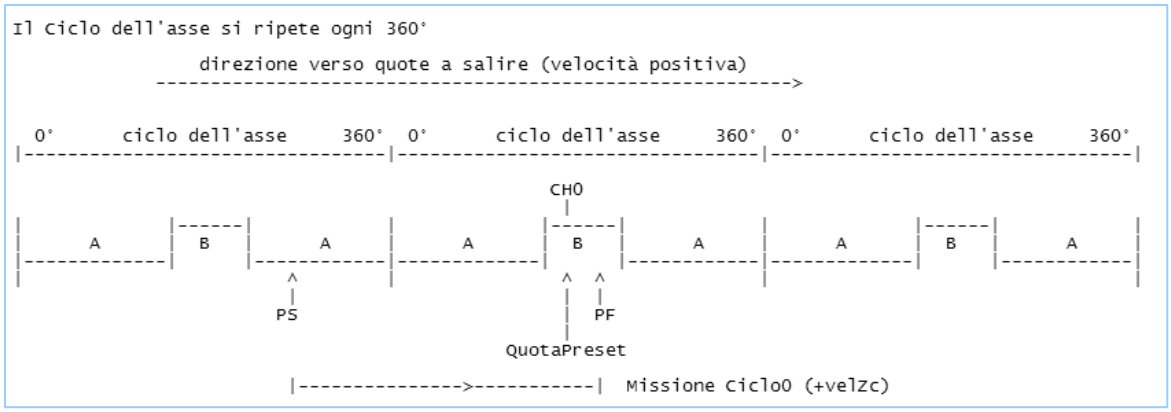
```
INT Flag
...
rule 123
    axes 1
    motion
        ; First time initialization
        if (first_time346())
            Flag = true
            zc.nAx    = 1 ; Axis number
            zc.preset = 0 ; Preselection quotas
            zc.velPos = rVelc0 ; Speed for preliminary movements mva_z
            zc.velZC  = rVelc0 * 0.5 ; Speed for final movement CYCLE0
            zc.acc    = rMaxAcc * 0.5 ; Acceleration to be used in all
            zc.jerk   = rMaxAcc * 10 ; Derivative of acceleration
            zc.spSCh  = 20000 ; Maximum space of the final movement CY
            zc.filter = 0 ; Time (sec) of micro signal filtering
            zc.zcFlag.0 = 1  ; Positive speed (since one-way)
            zc.zcFlag.1 = 1  ; Wants micro in 1 when looking for CH0
            zc.zcFlag.2 = 1  ; Enable full-precision
            zc.zcFlag.3 = 0  ; Enabled use of the micro
            zc.zcFlag.4 = 1  ; One-way search
            zc.zcFlag.5 = 0  ; Preliminary unrestricted excursion miss
            zc.zcFlag.6 = 0  ; Enable search first raising transition
            zc.zcFlag.7 = 0  ; Disable end of stroke search (because m
            zc.zcFlag.8 = 0  ; Disabilita ciclo di zero ON_POSITION
            zc.zcFlag.9 = 0  ; Disable positioning on preset elevation
            zc.zcFlag.10 = 0 ; Disable repositioning within software l
        endif

        ; Call move function
        esito = mva_zc511(Flag, zc)
    end_motion
end_rule
```

Asse rotativo monodirezionale: ciclo0 a micro + index SENZA ricerca preliminare della variazione del micro

Validità: **Rule**

Caso 1: l'asse parte dal punto PS in zona A.

© 2025 Robox SpA



Fasi:

a) Viene letto lo stato del micro per poter definire da quale movimentazione partire. Dato che il micro è in 0 e il ciclo0 si completa con micro in 1 non servono missioni preliminari.

b) Missione finale (CICLO0): si esegue il movimento finale con una corsa massima pari al valore del parametro spSch, con la velocità imposta (+velZc). Nel momento in cui viene trovato il primo canale di 0 encoder col micro in 1, la missione viene fermata e l'asse si arresta in PF.

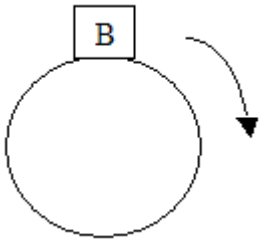
c) Si assegna come nuova quota reale il valore ottenuto sommando al parametro Preset il delta di quote (PF-quota letta sul primo CH0 encoder col micro in 1).

d) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse.

NOTA: la lunghezza della camma B letta dal micro deve essere superiore ad un giro encoder x garantire di vedere il CH0 col micro in 1

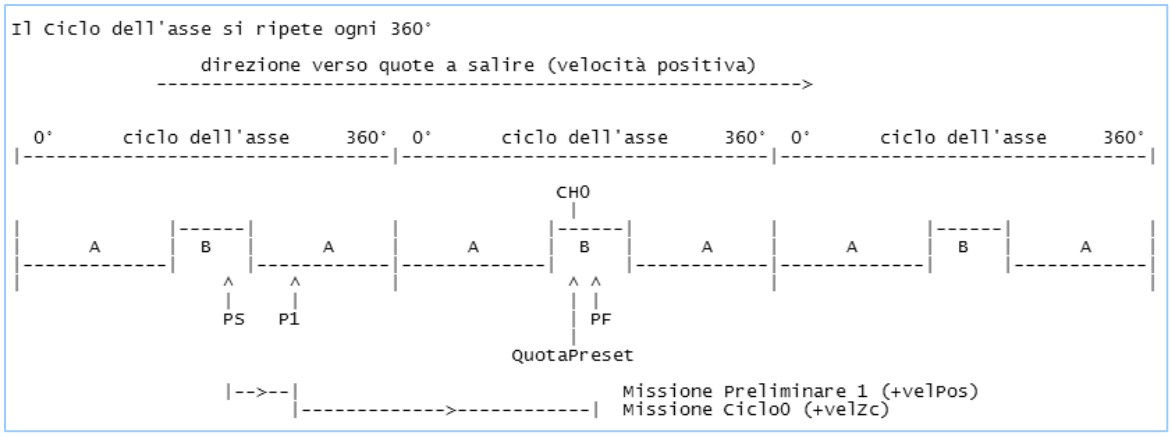
Legenda:

- *A, zona in cui il micro legge 0*
- *B, zona in cui il micro legge 1*
- *PS, punto di start iniziale per la mva_zc()*
- *PF, punto di arresto finale per la mva_zc()*



Caso 2: l'asse parte dal punto PS in zona B.

© 2025 Robox SpA



Fasi:

a) Viene letto lo stato del micro per poter definire da quale movimentazione partire. Dato che il micro è in 1 e il ciclo0 si completa con micro in 1 servono missioni preliminari.

b) Missione preliminare 1: l'asse muove alla velocità (+velPos) fino a riscontrare il fronte a scendere del micro, quindi decelera e si arresta in posizione P1.

c) Missione finale (CICLO0): si esegue il movimento finale con una corsa massima pari al valore del parametro spSch, con la velocità imposta (+velZc). Nel momento in cui viene trovato il primo canale di 0 encoder col micro in 1, la missione viene fermata e l'asse si arresta in PF.

d) Si assegna come nuova quota reale il valore ottenuto sommando al parametro Preset il delta di quote (PF-quota letta sul primo CH0 encoder col micro in 1).

e) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse.

NOTA: la lunghezza della camma B letta dal micro deve essere superiore ad un giro encoder x garantire di vedere il CH0 col micro in 1

Leggenda:

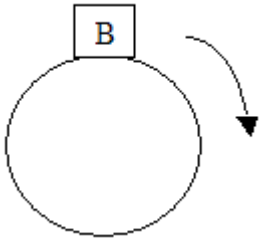
A, zona in cui il micro legge 0

B, zona in cui il micro legge 1

PS, punto di start iniziale per la mva_zc511()

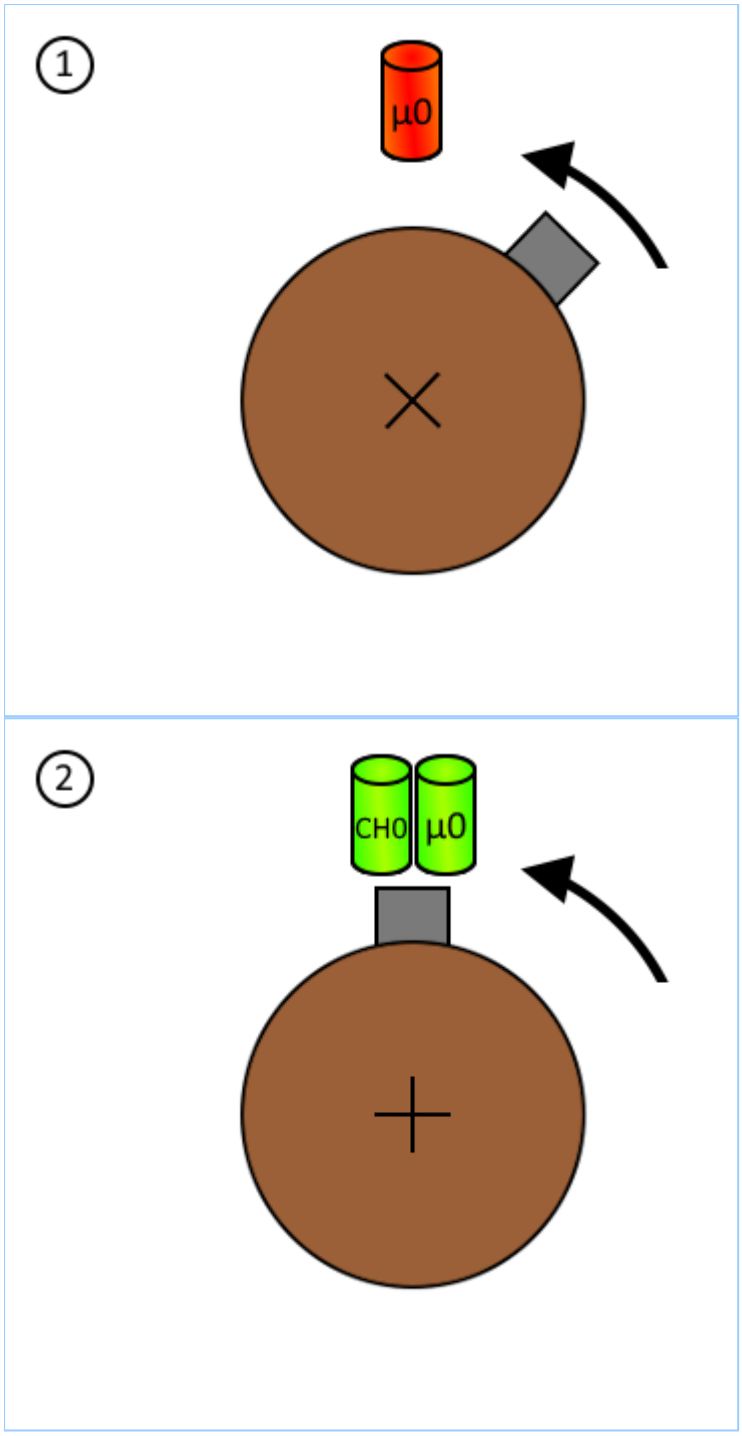
PF, punto di arresto finale per la mva_zc511()

P1, punto intermedio 1

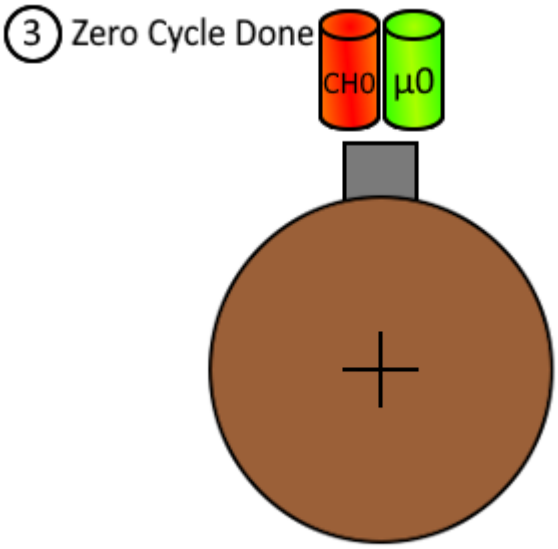


Esempio grafico

© 2025 Robox SpA



© 2025 Robox SpA



Esempi

Esempio di utilizzo:

```
INT esito
STRU_MVZC 892 zc
INT Flag
...
rule 123
    axes 1
    motion
        ; First time initialization
        if (first_time 346())
            Flag = true
            zc.nAx    = 1 ; Axis number
            zc.preset = 0 ; Preselection quotas
            zc.velPos = rVelc0 ; Speed for preliminary movements mva_z 511
            zc.velZC  = rVelc0 * 0.5 ; Speed for final movement CYCLE0
            zc.acc    = rMaxAcc * 0.5 ; Acceleration to be used in all
            zc.jerk   = rMaxAcc * 10 ; Derivative of acceleration
            zc.spSCh  = 20000 ; Maximum space of the final movement CY
            zc.filter = 0 ; Time (sec) of micro signal filtering
            zc.zcFlag.0 = 1 ; Positive speed (since one-way)
            zc.zcFlag.1 = 1 ; Wants micro in 1 when looking for CH0
            zc.zcFlag.2 = 1 ; Enable full-precision
            zc.zcFlag.3 = 0 ; Enable use of micro
            zc.zcFlag.4 = 1 ; One-way search
```

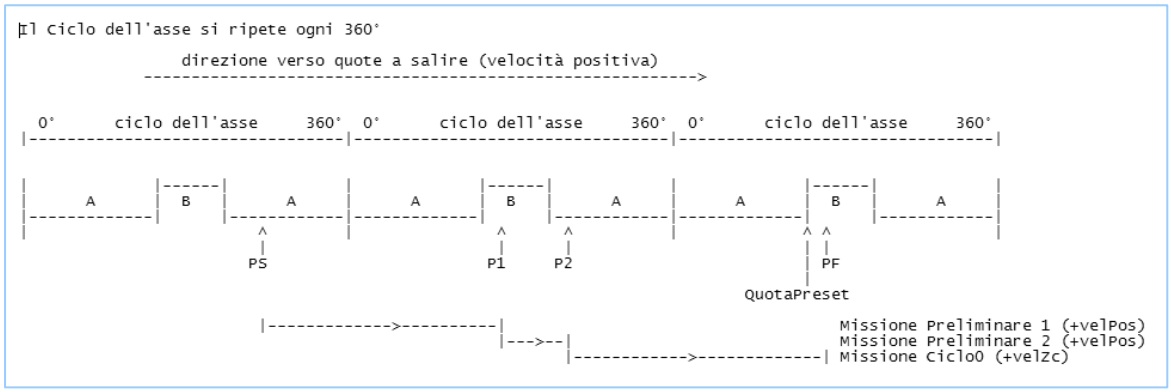
© 2025 Robox SpA

```
            zc.zcFlag.5 = 0 ; Preliminary unrestricted excursion missi
            zc.zcFlag.6 = 1 ; Disable search first transition to micro
            zc.zcFlag.7 = 0 ; Disable paracart search (because micro e
            zc.zcFlag.8 = 0 ; Disable zero cycle ON_POSITION
            zc.zcFlag.9 = 0 ; Disable positioning on preset elevation
            zc.zcFlag.10 = 0 ; Disable repositioning within software l
        endif

        ; Call move function
        esito = mva_zc511(Flag, zc)
    end_motion
end_rule
```

Asse rotativo monodirezionale: ciclo0 solo a micro con ricerca preliminare della variazione del micro

Validità: **Rule**

Caso 1: l'asse parte dal punto PS in zona A.



Fasi:

a) Dato che è abilitata ricerca preliminare della variazione del micro, missione preliminare 1: l'asse muove alla velocità (+velPos) fino a riscontrare il primo fronte del micro (in questo caso a salire), quindi decelera e si arresta in posizione P1.

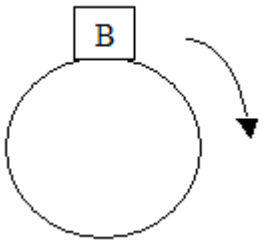
b) Dato che il micro è in 1 e il ciclo0 si completa con micro in 1 serve un'ulteriore missione preliminare.
Missione preliminare 2: l'asse muove alla velocità (+velPos) fino a riscontrare il fronte a scendere del micro, quindi decelera e si arresta in posizione P2.

Legenda:

- *A, zona in cui il micro legge 0*
- *B, zona in cui il micro legge 1*
- *PS, punto di start iniziale per la mva_zc511()*
- *PF, punto di arresto finale per la mva_zc511()*
- *P1, punto intermedio 1*
- *P2, punto intermedio 2*

© 2025 Robox SpA

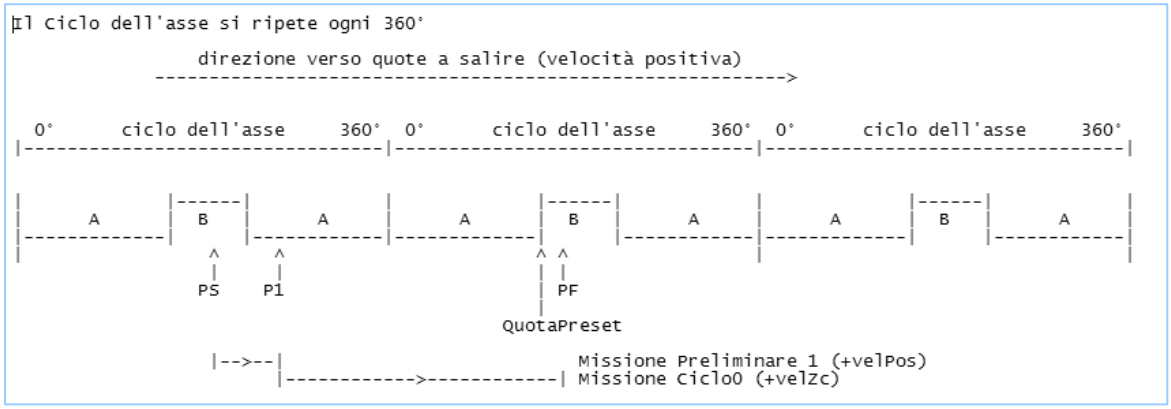
c) Missione finale (CICLO0): si esegue il movimento finale con una corsa massima pari al valore del parametro spSch, con la velocità imposta (+velZc). Nel momento in cui viene trovato il fronte a salire del micro la missione viene fermata e l'asse si arresta in PF.



d) Si assegna come nuova quota reale il valore ottenuto sommando al parametro Preset il delta di quote (PF-quota letta sul fronte a salire del micro).

e) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse.

Caso 2: l'asse parte dal punto PS in zona B.



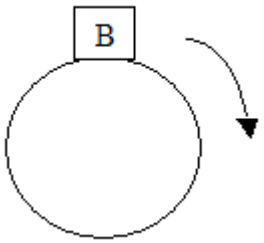
Fasi:

Leggenda:

- *A, zona in cui il micro legge 0*
- *B, zona in cui il micro legge 1*
- *PS, punto di start iniziale per la mva_zc511()*
- *PF, punto di arresto finale per la mva_zc511()*
- *P1, punto intermedio 1*

a) Dato che è abilitata ricerca preliminare della variazione del micro, missione preliminare 1: l'asse muove alla velocità (+velPos) fino a riscontrare il primo fronte del micro (in questo caso a scendere), quindi decelera e si arresta in posizione P1.

b) Dato che il micro è in 0 e il ciclo0 si completa con micro in 1 non servono ulteriori missioni preliminari.
Missione finale (CICLO0): si esegue il movimento finale con una corsa massima pari al valore del parametro spSch, con la velocità imposta (+velZc). Nel momento in cui viene trovato il fronte a salire del



© 2025 Robox SpA

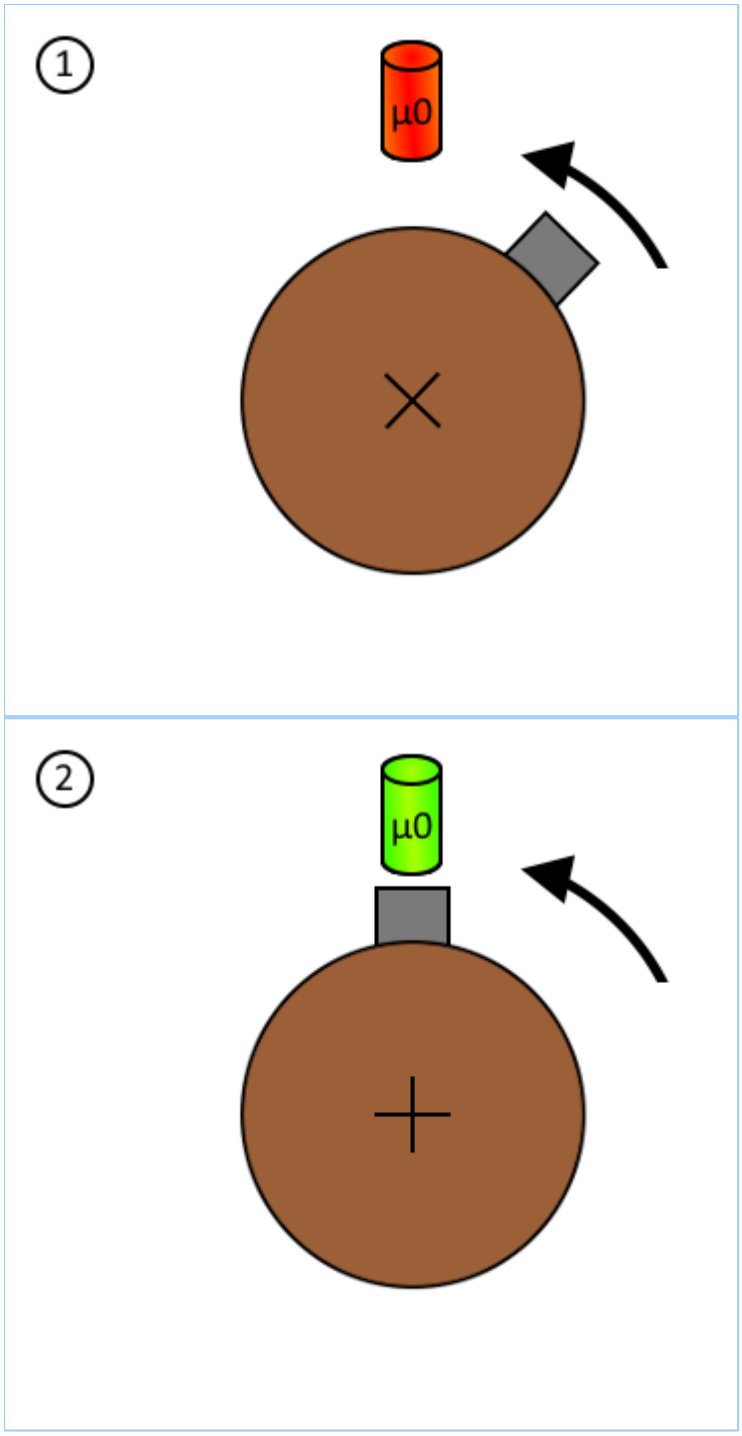
micro la missione viene fermata e l'asse si arresta in PF.

c) Si assegna come nuova quota reale il valore ottenuto sommando al parametro Preset il delta di quote (PF-quota letta sul fronte a salire del micro).

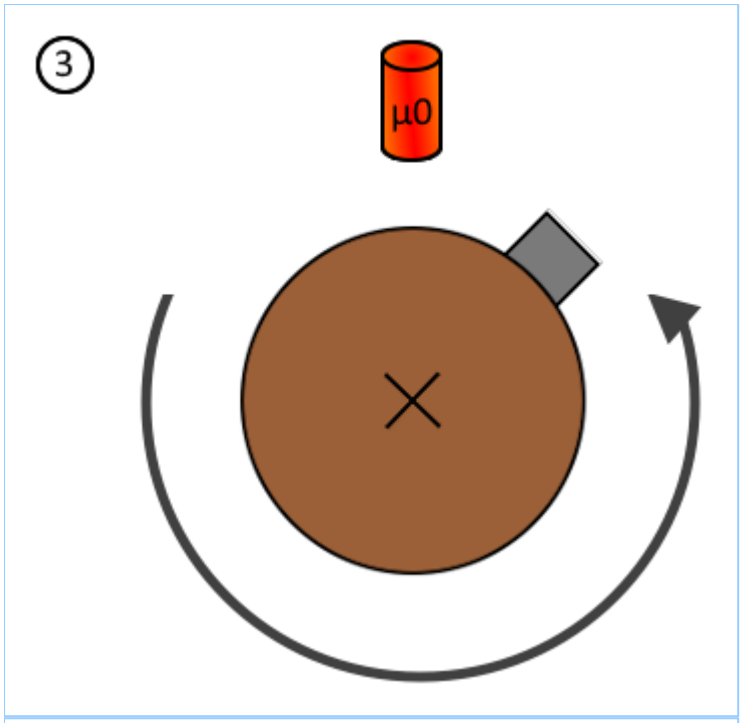
d) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse.

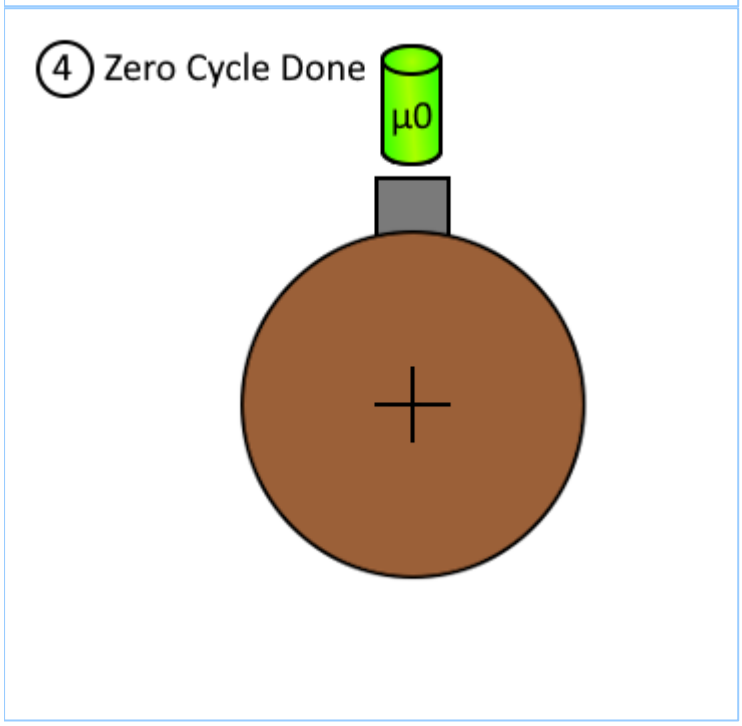
Esempio grafico

© 2025 Robox SpA



© 2025 Robox SpA





Esempi

Esempio di utilizzo:

```
INT esito
STRU_MVZC 892 zc
```

© 2025 Robox SpA

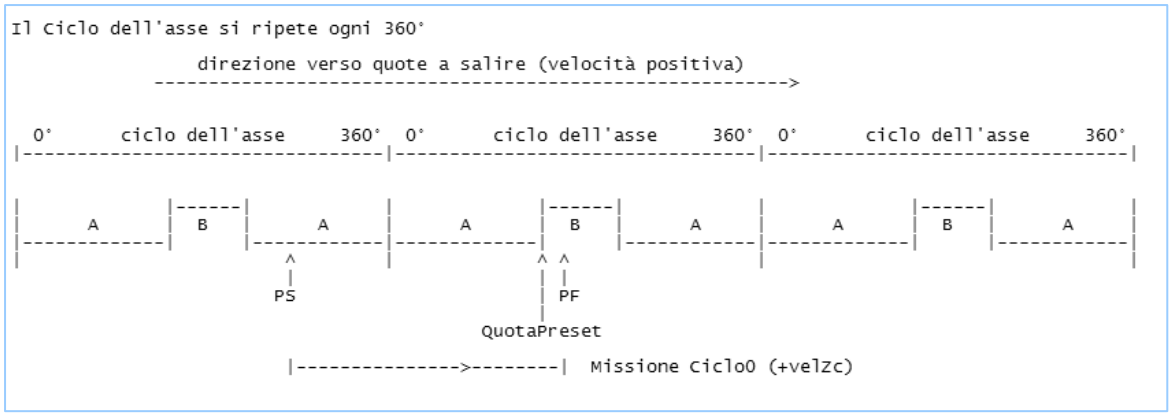
```
INT Flag
...
rule 123
    axes 1
    motion
        ; First time initialization
        if (first_time346())
            Flag = true
            zc.nAx    = 1 ; Axis number
            zc.preset = 0 ; Preselection quotas
            zc.velPos = rVelc0 ; Speed for preliminary movements mva_z 511
            zc.velZC  = rVelc0 * 0.5 ; Speed for final movement CYCLE0 511
            zc.acc    = rMaxAcc * 0.5 ; Acceleration to be used in all
            zc.jerk   = rMaxAcc * 10 ; Derivative of acceleration
            zc.spSCh  = 20000 ; Maximum space of the final movement CY
            zc.filter = 0 ; Time (sec) of micro signal filtering
            zc.zcFlag.0 = 1 ; Positive speed (given that one-way)
            zc.zcFlag.1 = 1 ; Selects the micro's rising edge
            zc.zcFlag.2 = 0 ; Disables full-precision
            zc.zcFlag.3 = 0 ; Enables use of the micro
            zc.zcFlag.4 = 1 ; One-way search
            zc.zcFlag.5 = 0 ; Preliminary missions of unrestricted exc
            zc.zcFlag.6 = 0 ; Enable search first upward transition of
            zc.zcFlag.7 = 0 ; Disable paracarro search (because micro
            zc.zcFlag.8 = 0 ; Disables zero cycle ON_POSITION
            zc.zcFlag.9 = 0 ; Disable positioning on preset elevation
            zc.zcFlag.10 = 0 ; Disables repositioning within the softw
        endif

        ; Call move function
        esito = mva_zc511(Flag, zc)
    end_motion
end_rule
```

Asse rotativo monodirezionale: ciclo0 solo a micro senza ricerca preliminare della variazione del micro

Validità: **Rule**

Caso 1: l'asse parte dal punto PS in zona A.

© 2025 Robox SpA



Fasi:

a) Viene letto lo stato del micro per poter definire da quale movimentazione partire. Dato che il micro è in 0 e il ciclo0 si completa con micro in 1 non servono missioni preliminari.

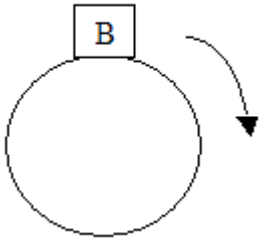
b) Missione finale (CICLO0): si esegue il movimento finale con una corsa massima pari al valore del parametro spSch, con la velocità imposta (+velZc). Nel momento in cui viene trovato il fronte a salire del micro la missione viene fermata e l'asse si arresta in PF.

c) Si assegna come nuova quota reale il valore ottenuto sommando al parametro Preset il delta di quote (PF-quota letta sul fronte a salire del micro).

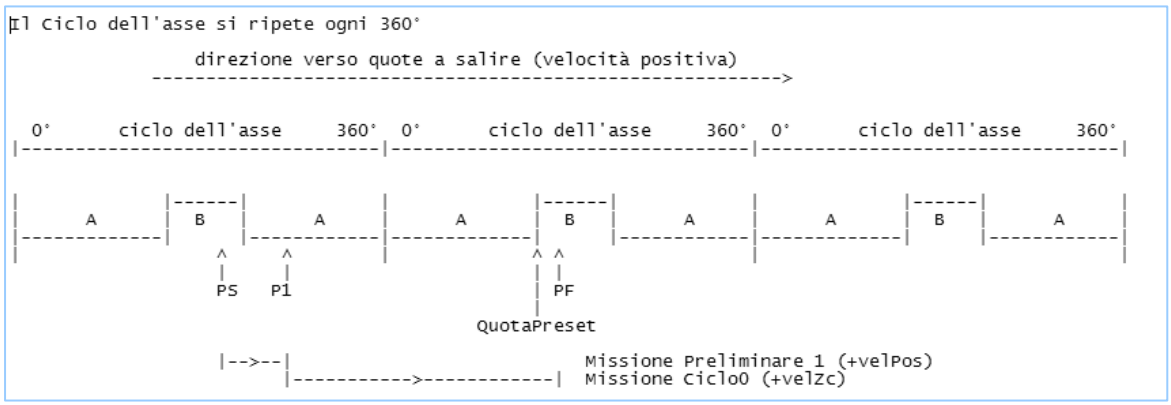
d) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse.

Legenda:

- *A, zona in cui il micro legge 0*
- *B, zona in cui il micro legge 1*
- *PS, punto di start iniziale per la mva_zc511()*
- *PF, punto di arresto finale per la mva_zc511()*



Caso 2: l'asse parte dal punto PS in zona B.



Fasi:

Leggenda:

© 2025 Robox SpA

a) Viene letto lo stato del micro per poter definire da quale movimentazione partire. Dato che il micro è in 0 e il ciclo0 si completa con micro in 1 non servono missioni preliminari.

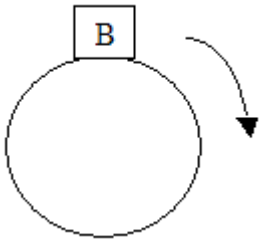
b) Missione preliminare 1: l'asse muove alla velocità (+velPos) fino a riscontrare il fronte a scendere del micro, quindi decelera e si arresta in posizione P1.

c) Missione finale (CICLO0): si esegue il movimento finale con una corsa massima pari al valore del parametro spSch, con la velocità imposta (+velZc). Nel momento in cui viene trovato il fronte a salire del micro la missione viene fermata e l'asse si arresta in PF.

d) Si assegna come nuova quota reale il valore ottenuto sommando al parametro Preset il delta di quote (PF-quota letta sul fronte a salire del micro).

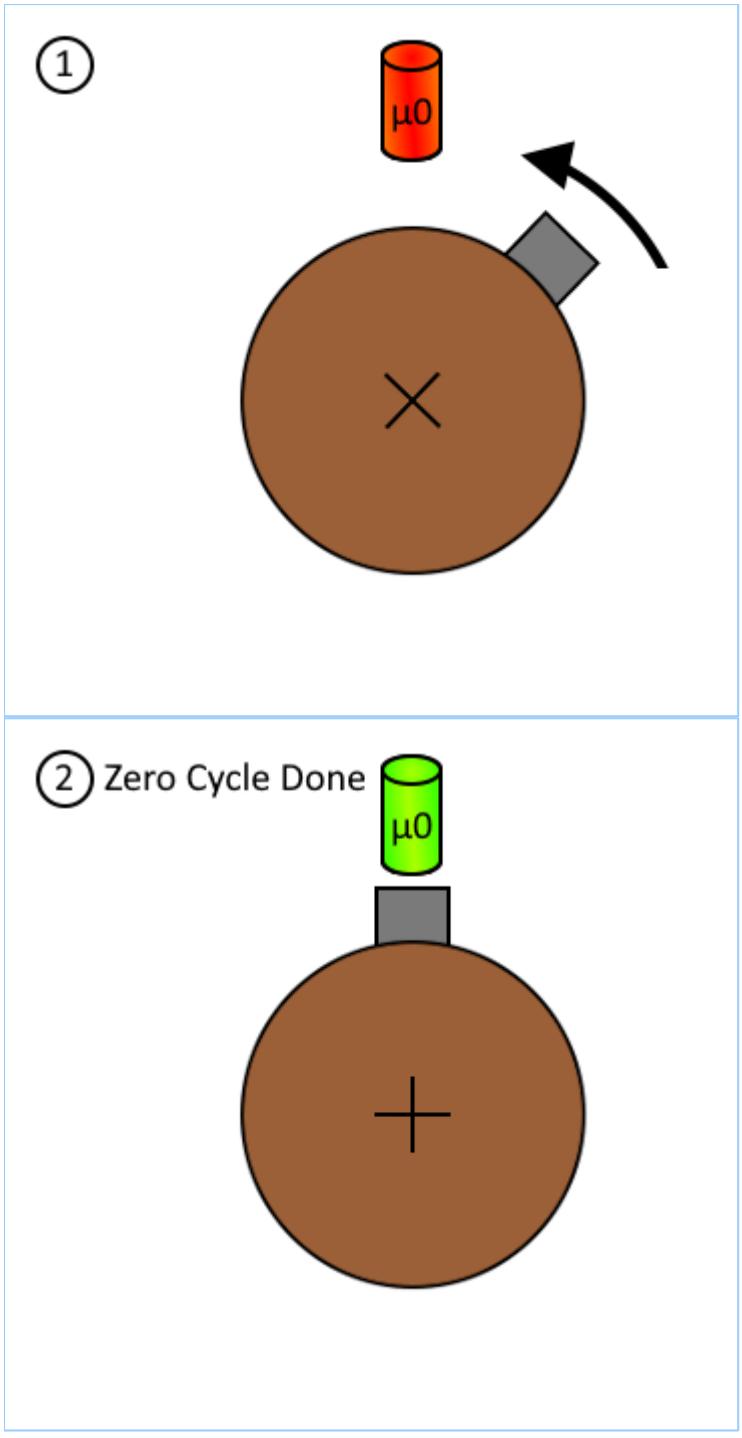
e) Si aggiorna la variabile predefinita c0_done ponendo a 1 il bit relativo all'asse

- *A, zona in cui il micro legge 0*
- *B, zona in cui il micro legge 1*
- *PS, punto di start iniziale per la mva_zc()*
- *PF, punto di arresto finale per la mva_zc()*
- *P1, punto intermedio 1*

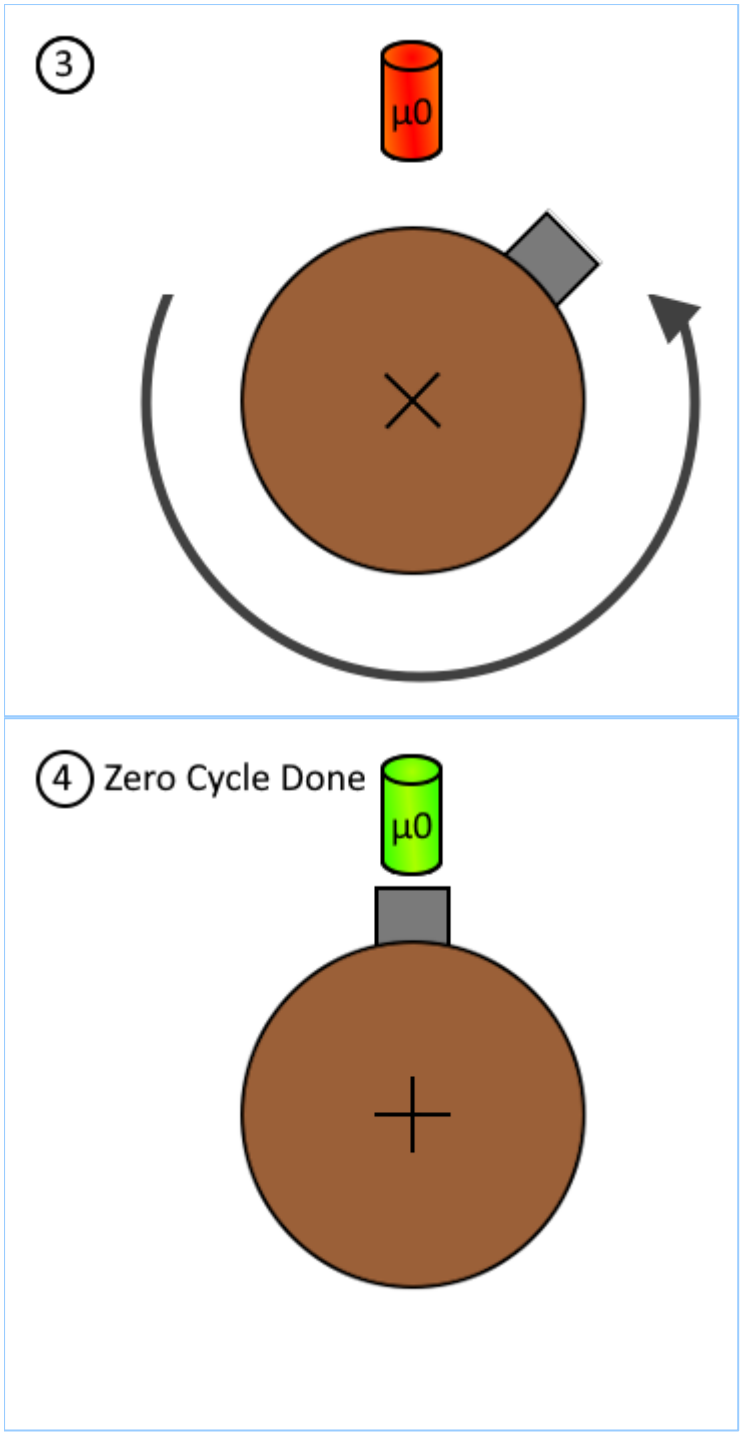


Esempio grafico

© 2025 Robox SpA



© 2025 Robox SpA



Esempi

Esempio di utilizzo:

```
INT esito
STRU_MVZC 892 zc
```

© 2025 Robox SpA

```
INT Flag
...
rule 123
    axes 1
    motion
        ; First time initialization
        if (first_time346())
            Flag = true
            zc.nAx    = 1 ; Axis number
            zc.preset = 0 ; Preselecton quotas
            zc.velPos = rVelc0 ; Speed for preliminary handling mva_zc
            zc.velZC = rVelc0 * 0.5 ; Speed for final movement CYCLE0
            zc.acc = rMaxAcc * 0.5 ; Acceleration to be used in all mo
            zc.jerk = rMaxAcc * 10 ; Derivative of acceleration
            zc.spSCh = 20000 ; Maximum space of the final movement CYC
            zc.filter = 0 ; Time (sec) for filtering the micro signal
            zc.zcFlag.0 = 1 ; Positive velocity (since one-way)
            zc.zcFlag.1 = 1 ; Selects the rising edge of the micro
            zc.zcFlag.2 = 0 ; Disables full-precision
            zc.zcFlag.3 = 0 ; Enables use of the micro
            zc.zcFlag.4 = 1 ; One-way search
            zc.zcFlag.5 = 0 ; Preliminary missions of unrestricted exc
            zc.zcFlag.6 = 1 ; Disable search first upward transition o
            zc.zcFlag.7 = 0 ; Disable paracarro search (because micro
            zc.zcFlag.8 = 0 ; Disable zero cycle ON_POSITION
            zc.zcFlag.9 = 0 ; Disable positioning on preset elevation
            zc.zcFlag.10 = 0 ; Disables repositioning within the softw
        endif

        ; Call move function
        esito = mva_zc511(Flag, zc)
    end_motion
end_rule
```

Categorie

Funzioni/istruzioni matematiche

Elenco di tutte le funzioni/istruzioni matematiche/trigonometriche messe a disposizione dal linguaggio di programmazione R3.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)

Keyword	R	T	F/I	Descrizione
abs156()	R	T	F	Valore assoluto di una grandezza
abs_mod157()	R	T	F	Resto della divisione

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
acos 157()	R	T	F	Arcocoseno
asin 178()	R	T	F	Arcoseno
atan 179()	R	T	F	Arcotangente
atan2 180()	R	T	F	Arcotangente
cos 290()	R	T	F	Coseno
fract 350()	R	T	F	Restituisce parte frazionaria dell'argomento
log 388()	R	T	F	Restituisce il logaritmo naturale dell'argomento
log10 388()	R	T	F	Restituisce il logaritmo base10 dell'argomento
mod 439()	R	T	F	Rende il resto della divisione fra variabili numeriche
pow 525()	R	T	F	Elevazione a potenza
rand32 556()	R	T	F	Restituisce un numero casuale (i32)
random 557()	R	T	F	Restituisce un numero casuale (i16)
random_value 557()	R	T	F	Restituisce un numero casuale (real)
range 557()	R	T	F	Rende TRUE se la variabile è nel range
range_adj 558()	R	T	F	Scaling di una variabile
round 565()	R	T	F	Rende il numero intero più vicino
sign 600()	R	T	F	Restituisce il segno della

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				variabile
sin 602()	R	T	F	Seno
sqrt 620()	R	T	F	Radice quadrata
tan 641()	R	T	F	Tangente
trunc 650()	R	T	F	Rende la parte intera dell'argomento

Funzioni/istruzioni safety

Elenco di tutte le funzioni/istruzioni safety messe a disposizione dal linguaggio di programmazione R3.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
safe_appl_ack 567()	R	T	F	Setta il valore dell'Application Acknowledge di un'isola safety
safe_appl_diag 568()	R	T	F	Ritorna lo stato degli Application Diagnostic di un'isola safety
safe_diag 569()	R	T	F	Ritorna il registro di diagnostica del satellite desiderato
safe_diag_reset 572()	R	T	F	Invia un Diagnostic Reset al satellite selezionato
safe_enab_out 573()	R	T	F	Imposta il valore del registro di abilitazione degli output dell'isola selezionata
safe_oper_ack 573()	R	T	F	Impone un Operation Acknowledge sul satellite selezionato

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
safe_prj_info 574()	-	T	F	Ottieni informazioni riguardo il progetto safety in esecuzione sull'isola specificata
safe_sts 575()	R	T	F	Ritorna lo stato degli input/output safety del satellite selezionato

Funzioni/istruzioni supporto rule/task

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la gestione delle rule e dei task.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
group 721()	R	T	I	Attiva gli esecutori delle RULE (RC)
int_after 725()	R	T	I	Abilita RULE su evento (priorità 1) ritardata su evento
int_inp 726()	R	T	I	Abilita RULE su evento (priorità 1) su variazione input veloce
int_timer 728()	R	T	I	Abilita RULE su evento (priorità 1) a frequenza fissa
mt_ats 449()	R	T	F	Costruisce un semaforo atomico
mt_ats2 450()	R	T	F	Costruisce un semaforo atomico
mt_en 734()	R	T	I	Abilita l'esecuzione da

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				parte di RTE del task o dei task in time-sharing specificati (priorità 7)
mt_dis 733()	R	T	I	Disabilita l'esecuzione da parte di RTE del task o dei task in time-sharing specificati (priorità 7)
mt_wait 735()	R	T	I	Arresta (sleep) il task finché la condizione espressa non diventa vera.
order 735()	R	T	I	Stabilisce l'ordine di esecuzione (OC) degli esecutori delle RULE (RC)
ref_prologue 745()	R	-	I	RULE sempre attiva a prescindere dai rule counter (RC)
rule 748()	R	-	I	Definizione di una regola
rule_epilogue 756()	R	-	I	Rule sempre attiva a prescindere dai rule counter (RC)
rule_freq 756()	R	T	I	Definizione intervallo di campionamento (SI) RULE (priorità 2)
rule_init 757()	R	-	I	Regola di inizializzazione (gira solo all'accensione)
rule_periodic 758()	R	T	I	Abilita RULE periodica (priorità 4) a frequenza fissa

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
rule_prologue 759()	R	-	I	Rule sempre attiva a prescindere dai rule counter (RC)

Gestione allarmi

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la gestione dell'allarmistica.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
al2am 166()	R	T	F	Restituisce informazioni sull'AM in funzione di un allarme
alarm_set 175()	R	T	F	Set di un allarme
alarm_set_us 176()	R	T	F	Set di un allarme con variabile stringa utente USV
alarm_reset 169()	R	T	F	Reset di un allarme
alarm_reset_par 170()	R	T	F	Reset di un allarme specificando anche il parametro
alarm_reset_par_us 172()	R	T	F	Reset di un allarme specificando anche il parametro USV
ah_cmd 158()	R	T	F	Comandi per storico allarmi
ah_get_d 159()	R	T	F	rende l'allarme indicato nello storico allarmi
ah_get_s 160()	R	T	F	Rende il testo dell'allarme indicato nello storico allarmi

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
ah_info 162()	R	T	F	Rende il numero di allarmi presenti nello storico
ah_log 163()	R	T	F	Inserisce un messaggio nello storico allarmi
ah_log_us 164()	R	T	F	Inserisce un messaggio nello storico allarmi con parametro USV
as_get_d 178()	R	T	F	Rende l'allarme indicato nello stack allarmi
inf_report 724()	R	T	I	Permette di inserire una informazione nel report di sistema
is_alarm_present 383()	R	T	F	Controlla la presenza dell'allarme nello stack
is_alarm_par_present 379()	R	T	F	Controlla la presenza dell'allarme e del parametro nello stack
is_alarm_par_present_us 381()	R	T	F	Verifica la presenza dell'allarme con parametro USV nello stack

NOTA: Utilizzare la keyword sys_log per l'autogenerazione in CF di un file di log.

NOTA: Utilizzare la keyword sys_flag per aggiungere informazioni alla diagnostica del sistema ottenuta tramite il comando MREPORT.

© 2025 Robox SpA

Gestione Axioline

Elenco di tutte le funzioni/istruzioni Axioline messe a disposizione dal linguaggio di programmazione R3 per la gestione del bus Axioline.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)

Keyword	R	T	F/I	Descrizione
axio_robj 186()	-	T	F	Funzione per leggere oggetti su fieldbus Axioline
axio_wobj 199()	-	T	F	Funzione per scrivere oggetti su fieldbus Axioline
axio_pm_r 183()	-	T	F	Funzione per leggere i process data input di un modulo AXL_F_PM_EF_1 F
axio_iol_r 181()	-	T	F	Funzione per leggere un'area dati da IOlink master noto
axio_iol_w 182()	-	T	F	Funzione per scrivere un'area dati su IOlink master noto
get_hw_info 351()	-	T	F	Permette di avere informazioni relative alle schede direttamente connesse alla CPU

Gestione canali seriali

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la gestione dei canali seriali RS232/RS422/RS485.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)

Keyword	R	T	F/I	Descrizione
ser_485_tx_sts 585()	R	T	F	Funzione che rende lo stato dell'ultima

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				trasmissione fatta sul dispositivo specificato
ser_bcc_h_init 587()	R	T	F	Inizializza la seriale per la comunicazione BCC3 tra controlli
ser_bin_in 588()	R	T	F	Funzione che permette di leggere un numero specifico di caratteri dal dispositivo specificato.
ser_bin_out 588()	R	T	F	Invia i caratteri sul canale seriale
ser_bin_ready 589()	R	T	F	Segnala se ci sono caratteri sul canale seriale
ser_buff_free 590()	R	T	F	Numero di Byte liberi sul buffer TX
ser_buff_queue 590()	R	T	F	Numero di Byte occupati sul buffer TX
ser_con 765()	R	T	I	Imposta i parametri del canale seriale
ser_flush 590()	R	T	F	Pulisce tutti i dati nei buffer di TX e RX della seriale specificata
ser_in 591()	R	T	F	Riceve da seriale la stringa terminata
ser_mod 592()	R	T	F	Imposta la modalità di lavoro del canale seriale specificato
ser_out 593()	R	T	F	Invia sul seriale la stringa terminata

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
ser_r_sts 593()	R	T	F	Legge la parola di stato del seriale
ser_ready 594()	R	T	F	Segnala la presenza di una stringa terminata sul seriale
ser_w_sts 766()	R	T	I	Imposta la parola di stato del seriale

Gestione CANopen/EtherCAT

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la gestione del protocollo CANopen DS301/DS401/DS402.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
co_send_sdo 266()	R	T	F	Invia un messaggio SDO (service data object)
co_asw_sdo 239()	R	T	F	Restituisce la risposta alla precedente istruzione co_send_sdo
co_controlword_om 241()	R	T	F	Funzione per scrivere alcuni bit della controlword
co_controlword_state 243()	R	T	F	Funzione per effettuare transizioni nella macchina a stati DS 402
co_statusword 270()	R	T	F	Restituisce il contenuto della status word della workstation indicata
co_par_download 248()	-	T	F	Funzione per caricare sul drive i parametri leggendoli da

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				Compact flash controllo
co_par_upload 251 ()	-	T	F	Funzione per salvare su un file in CF i parametri leggendoli da drive.
co_get_ws_diagno 245()	R	T	F	Funzione per leggere informazioni di stato della workstation indicata (solo canOpen).
co_rerr 255()	R	T	F	Restituisce informazioni di errore della workstation indicata
co_rnmt 256()	R	T	F	Funzione che rende il modo operativo NMT della WS indicata
co_wnmt 272()	R	T	F	Impone lo stato NMT della workstation indicata
co_robj 258()	R	T	F	Funzione per leggere oggetti fieldbus
co_wobj 273()	R	T	F	Funzione per scrivere oggetti fieldbus
co_rpdo 265()	R	T	F	Legge un'oggetto canOpen tramite PDO (solo canOpen)
co_sync 272()	R	T	F	Funzione per abilitare o disabilitare la trasmissione del messaggio di SYNC
co_wpdo 278()	R	T	F	Scrive un'oggetto

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				canOpen tramite PDO (solo canOpen)
coe_robj 279()	R	T	F	Funzione per leggere oggetti fieldbus COE
coe_wobj 284()	R	T	F	Funzione per scrivere oggetti fieldbus COE
ecat_dev_on_off 309()	-	T	F	Funzione per abilitare/disabilitare un dispositivo su linea EtherCAT
ecat_get_state 312()	R	T	F	Funzione per leggere lo stato della linea EtherCAT
ecat_probe 313()	-	T	F	Funzione per leggere le informazioni di un dispositivo su linea EtherCAT
ecat_set_state 314()	R	T	F	Funzione per imporre lo stato della linea EtherCAT
soe_ridn 603()	R	T	F	Funzione per leggere oggetti fieldbus SOE
soe_widn 615()	R	T	F	Funzione per scrivere oggetti fieldbus SOE

Gestione CNET

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la gestione del protocollo CNET.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
cnet_diag()	R	T	F	Funzione gestione CNET.

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				Riservata
cnet_get_time_sync()	R	T	F	Funzione gestione CNET. Riservata
cnet_recive()	R	T	F	Funzione gestione CNET. Riservata
cnet_receive_status()	R	T	F	Funzione gestione CNET. Riservata
cnet_rq_sync()	R	T	F	Funzione gestione CNET. Riservata
cnet_send()	R	T	F	Funzione gestione CNET. Riservata
cnet_status()	R	T	F	Funzione gestione CNET. Riservata

Gestione comunicazione BCC

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per gestire una comunicazione con il protocollo BCC.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
bcc_h_setup 225()	R	T	F	Prepara la struttura dati per la risposta ad un messaggio
bcc_receive 226()	R	T	F	Legge un messaggio BCC3
bcc_send 229()	R	T	F	Invia un messaggio BCC3
ser_bcc_h_init 587()	-	T	F	Inizializza una porta seriale per comunicare con il protocollo BCC3
bcc_get_double 219()	R	T	F	Legge un double da una struttura

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				BCC_MSGUSER
bcc_get_float220()	R	T	F	Legge un float da una struttura BCC_MSGUSER
bcc_get_i16220()	R	T	F	Legge un i16 da una struttura BCC_MSGUSER
bcc_get_i32221()	R	T	F	Legge un i32 da una struttura BCC_MSGUSER
bcc_get_i8221()	R	T	F	Legge un i8 da una struttura BCC_MSGUSER
bcc_get_string222()	R	T	F	Legge una stringa da una struttura BCC_MSGUSER
bcc_get_tcp_clients_info222()	R	T	F	Legge un i32 da una struttura STRU_BCC_TCP_CLIENT
bcc_get_u16223()	R	T	F	Legge un u16 da una struttura BCC_MSGUSER
bcc_get_u32224()	R	T	F	Legge un u32 da una struttura BCC_MSGUSER
bcc_get_u8224()	R	T	F	Legge un u8 da una struttura BCC_MSGUSER
bcc_reply_ack227()	R	T	F	Invia una risposta positiva (ACK) ad un messaggio ricevuto
bcc_reply_msg228()	R	T	F	Invia una risposta ad un messaggio ricevuto
bcc_reply_nack229()	R	T	F	Invia una risposta negativa (NACK) ad un messaggio ricevuto

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
bcc_set_double 683 ()	R	T	I	Scrive un double in una struttura BCC_MSGUSER
bcc_set_float 684()	R	T	I	Scrive un float in una struttura BCC_MSGUSER
bcc_set_i16 684()	R	T	I	Scrive un i16 in una struttura BCC_MSGUSER
bcc_set_i32 685()	R	T	I	Scrive un i32 in una struttura BCC_MSGUSER
bcc_set_i8 685()	R	T	I	Scrive un i8 in una struttura BCC_MSGUSER
bcc_set_string 686()	R	T	I	Scrive una stringa in una struttura BCC_MSGUSER
bcc_set_u16 686()	R	T	I	Scrive un u16 in una struttura BCC_MSGUSER
bcc_set_u32 687()	R	T	I	Scrive un u32 in una struttura BCC_MSGUSER
bcc_set_u8 688()	R	T	I	Scrive un u8 in una struttura BCC_MSGUSER

Gestione controllo di flusso

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per gestire il flusso del programma.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
break 694	R	T	I	Uscita da un blocco di selezione
call 694	R	T	I	Chiamata di un sottoprogramma

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
case 696	R	T	I	costrutto Select variabile
cond_val 289	R	T	I	Costrutto if-then-else semplificato
continue 697	R	T	I	Costrutto for - while
default 702	R	T	I	Costrutto select
do 703	R	T	I	Ciclo do
dwell 704	-	T	I	Arresta il task per n secondi
_else 678	R	T	I	Istruzione controllo di flusso altrimenti
else 704	R	T	I	Scelta alternativa blocco if
first_time 346()	R	-	F	Gira solo la prima volta che la rule 748 viene attivata
for 713	R	T	I	Ciclo for
function 715	R	T	I	Funzione (sottoprogramma)
goto 721	R	T	I	Istruzione di salto alla label
_if 678	R	T	I	Istruzione controllo di flusso SE
if 723	R	T	I	Costrutto if
__main_loop__ 677	R	T	I	Loop del task
restart_point 746	-	T	I	Punto di interruzione esecuzione task prima di modifica live 933
select 764	R	T	I	Costrutto select end_select
while 789	R	T	I	Costrutto while end_while

© 2025 Robox SpA

Gestione conversione endianess

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la gestione della conversione endianess.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
Funzioni di lettura:				
get_i8 355()	R	T	F	Leggi un byte (I8) da un buffer in un offset specificato
get_u8 355()	R	T	F	Leggi un byte (U8) da un buffer in un offset specificato
get_i16 353()	R	T	F	Leggi una word (I16) da un buffer in un offset specificato
get_u16 356()	R	T	F	Leggi una word (U16) da un buffer in un offset specificato
get_i32 354()	R	T	F	Leggi una dword (I32) da un buffer in un offset specificato
get_u32 357()	R	T	F	Leggi una dword (U32) da un buffer in un offset specificato
get_i64 719()	R	T	I	Leggi una qword (I64) da un buffer in un offset specificato
get_u64 720()	R	T	I	Leggi una qword (U64) da un buffer in un offset specificato
get_float 350()	R	T	F	Leggi un float (x32) da un buffer in un offset specificato
get_real 355()	R	T	F	Leggi un real (x64) da un buffer in un offset specificato

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
Funzioni di scrittura:				
set_i8 771()	R	T	I	Scrivi un byte (I8) da un buffer in un offset specificato
set_u8 775()	R	T	I	Scrivi un byte (U8) da un buffer in un offset specificato
set_i16 768()	R	T	I	Scrivi una word (I16) da un buffer in un offset specificato
set_u16 772()	R	T	I	Scrivi una word (U16) da un buffer in un offset specificato
set_i32 769()	R	T	I	Scrivi una dword (I32) da un buffer in un offset specificato
set_u32 773()	R	T	I	Scrivi una dword (U32) da un buffer in un offset specificato
set_i64 770()	R	T	I	Scrivi una qword (I64) da un buffer in un offset specificato
set_u64 774()	R	T	I	Scrivi una qword (U64) da un buffer in un offset specificato
set_float 767()	R	T	I	Scrivi un float (x32) da un buffer in un offset specificato
set_real 772()	R	T	I	Scrivi un real (x64) da un buffer in un offset specificato
Funzioni di conversioni endian x16:				
wbe2le 674()	R	T	F	Converti big endian word

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				(x16 o x32) in little endian word (x16)
wle2be 675()	R	T	F	Converti little endian word (x16 o x32) in big endian word (x16)
Funzioni di conversione endian x32:				
dwbe2le 302()	R	T	F	Converti big endian dword (x32) in little endian dword (x32)
dwle2be 304()	R	T	F	Converti little endian dword (x32) in big endian wdord (x32)
dwbe2lef 302()	R	T	F	Converti big endian dword (x32) in little endian float (x32)
dwle2bef 304()	R	T	F	Converti little endian float (4 bytes) in big endian dword (U32)
Funzioni di conversione middle endian x32:				
dwle2mbe 305()	R	T	F	Converti little endian dword (x32) in middle big endian dword (x32)
dwle2mbef 305()	R	T	F	Converti little endian float (4 bytes) in middle big endian dword (U32)
dwmbe2le 307()	R	T	F	Converti middle big endian dword (x32) in little endian dword (x32)

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
dwmbe2lef308()	R	T	F	Converti middle big endian (U32) in little endian float (4 bytes)
dwle2mle306()	R	T	F	Converti little endian dword (x32) in middle little endian dword (x32)
dwle2mlef306()	R	T	F	Converti little endian float (4 bytes) in middle little endian dword (U32)
dwbe2mbe303()	R	T	F	Converti big endian dword (x32) in middle big endian dword (x32)
dwbe2mle303()	R	T	F	Converti big endian dword (x32) in middle little endian dword (x32)
dwmbe2be307()	R	T	F	Converti middle big endian dword (x32) in big endian dword (x32)
dwmle2be308()	R	T	F	Converti middle little endian dword (x32) in big endian dword (x32)
Funzioni di conversione endian x64, con uso di stru_large_int e stru_large_uint:				
qwbe2le738()	R	T	I	Converti big endian qword (x64) in little endian qword (x64)
qwbe2lef547()	R	T	F	Converti big endian qword (x64) in little endian real (x64)
qwle2be739()	R	T	I	Converti little endian qword

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				(x64) in big endian qword (x64)
qwle2bef 741()	R	T	I	Converti little eendian real (x64) in big endian qword (x64)
Funzioni di conversione middle endian x64, con uso di stru_large_int e stru_large_uint:				
qwmbe2le 743()	R	T	I	Converti middle big endian qword (x64) in little endian qword (x64)
qwmbe2lef 548()	R	T	F	Converti middle big endian qword (x64) in little endian real (x64)
qwle2mbe 741()	R	T	I	Converti little endian qword (x64) in middle big endian qword (x64)
qwle2mbef 743()	R	T	I	Converti little endian real (x64) in middle big endian qword (x64)
Funzioni di conversione dword exchange endian x64, con uso di stru_large_int e stru_large_uint:				
qwbedwx2le 739()	R	T	I	Converti dword exchange big endian qword (x64) in little endian qword (x64)
qwbedwx2lef 547()	R	T	F	Converti dword exchange big endian qword (x64) in little endian real (x64)
qwle2bedwx 740()	R	T	I	Converti little endian qword (x64) in dword exchange big

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				endian qword (x64)
qwle2bedwxf740()	R	T	I	Converti little endian real (x64) in dword exchange big endian real (x64)
Funzioni di conversione dword exchange middle endian x64, con uso di stru_large_int e stru_large_uint:				
qwmbedwx2le744()	R	T	I	Converti dword exchange middle big endian qword (x64) in little endian qword (x64)
qwmbedwx2lef548()	R	T	F	Converti dword exchange middle big endian qword (x64) in little endian real (x64)
qwle2mbedwx740()	R	T	I	Converti little endian qword (x64) in dword exchange middle big endian qword (x64)
qwle2mbedwxf740()	R	T	I	Converti little endian real (x64) in dword exchange middle big endian qword (x64)

Gestione counter

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la gestione dei counter.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
ct_cd697()	R	T	I	Counter DOWN
ct_cdr698()	R	T	I	Counter DOWN ring

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
ct_cu 699()	R	T	I	Counter UP
ct_cur 700()	R	T	I	Counter UP ring
ct_cv 294() / ct_cv 701()	R	T	F/I	Valore corrente del counter
ct_load 701()	R	T	I	Carica il PV nel CV del counter
ct_pv 294() / ct_pv 701()	R	T	F/I	Valore di preset del counter
ct_qd 295()	R	T	F	Esito del Counter down
ct_qu 295()	R	T	F	Esito del Counter Up
ct_reset 702()	R	T	I	Reset del CV del counter

Gestione dati

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la manipolazione dei dati.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
b0 213() / b0 681()	R	T	F/I	Funzione che restituisce/scrive il byte 0 dell'argomento
b1 215() / b1 681()	R	T	F/I	Funzione che restituisce/scrive il byte 1 dell'argomento
b2 215() / b2 681()	R	T	F/I	Funzione che restituisce/scrive il byte 2 dell'argomento
b3 216() / b3 682()	R	T	F/I	Funzione che restituisce/scrive il byte 3 dell'argomento
b4 216() / b4 682()	R	T	F/I	Funzione che restituisce/scrive

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				il byte 4 dell'argomento
b5 217() / b5 682()	R	T	F/I	Funzione che restituisce/scrive il byte 5 dell'argomento
b6 218() / b6 683()	R	T	F/I	Funzione che restituisce/scrive il byte 6 dell'argomento
b7 218() / b7 683()	R	T	F/I	Funzione che restituisce/scrive il byte 7 dell'argomento
crc16_be 290()	R	T	F	Funzione per il calcolo del CRC (big endian)
crc16_le 292()	R	T	F	Funzione per il calcolo del CRC (little endian)
dw0 300() / dw0 703()	R	T	F/I	Funzione che restituisce/scrive la dword 0 dell'argomento
dw1 301() / dw1 704()	R	T	F/I	Funzione che restituisce/scrive la dword 1 dell'argomento
fill_mem 712()	R	T	I	Permette di inizializzare un area di memoria
move_byte 731()	R	T	I	Copia una zona di memoria
move_mem 732()	R	T	I	Copia una zona di memoria
r2m16 553()	R	T	F	Conversione da real a U16
r2m32 554()	R	T	F	Conversione da real a U32
r2m8 554()	R	T	F	Conversione da real a U8

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
r_ror 550()	R	T	F	Rotazione a destra del numero di bit indicati
r_rol 549()	R	T	F	Rotazione a sinistra del numero di bit indicati
r_shl 551()	R	T	F	Shift aritmetico a sinistra del numero di bit indicati
r_shr 552()	R	T	F	Shift aritmetico a destra del numero di bit indicati
rbit 559()	R	T	F	Reset di bit in una variabile intera
sb0 577() / sb0 760()	R	T	F/I	Funzione che restituisce/scrive il byte 0 dell'argomento con l'estensione del segno
sb1 578() / sb1 761()	R	T	F/I	Funzione che restituisce/scrive il byte 1 dell'argomento con l'estensione del segno
sb2 578() / sb2 761()	R	T	F/I	Funzione che restituisce/scrive il byte 2 dell'argomento con l'estensione del segno
sb3 579() / sb3 761()	R	T	F/I	Funzione che restituisce/scrive il byte 3 dell'argomento con l'estensione del segno
sb4 580() / sb4 762()	R	T	F/I	Funzione che restituisce/scrive il byte 4 dell'argomento

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				con l'estensione del segno
sb5 580() / sb5 762()	R	T	F/I	Funzione che restituisce/scrive il byte 5 dell'argomento con l'estensione del segno
sb6 581() / sb6 762()	R	T	F/I	Funzione che restituisce/scrive il byte 6 dell'argomento con l'estensione del segno
sb7 582() / sb7 763()	R	T	F/I	Funzione che restituisce/scrive il byte 7 dell'argomento con l'estensione del segno
sbit 582()	R	T	F	Set di bit in una variabile intera
sdw0 583() / sdw0 763()	R	T	F/I	Funzione che restituisce/scrive la dword 0 dell'argomento con l'estensione del segno
sdw1 584() / sdw1 763()	R	T	F/I	Funzione che restituisce/scrive la dword 1 dell'argomento con l'estensione del segno
shift_left 596()	R	T	F	Shift di un bit a sinistra dell'area di memoria indicata
shift_right 597()	R	T	F	Shift di un bit a destra dell'area di memoria indicata
shl 599()	R	T	F	Shift logico a sinistra
shr 599()	R	T	F	Shift logico a destra

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
sizeof 602()	R	T	F	Rende la dimensione della variabile
sum8_mem 637()	R	T	F	Restituisce la somma di n elementi di un array (u8)
sum16_mem 638()	R	T	F	Restituisce la somma di n elementi di un array (u16)
sum32_mem 638()	R	T	F	Restituisce la somma di n elementi di un array (u32)
sumr_mem 638()	R	T	F	Restituisce la somma di n elementi di un array (real)
sw0 639() / sw0 777()	R	T	F/I	Funzione che restituisce/scrive la word 0 dell'argomento con l'estensione del segno
sw1 639() / sw1 777()	R	T	F/I	Funzione che restituisce/scrive la word 1 dell'argomento con l'estensione del segno
sw2 640() / sw2 778()	R	T	F/I	Funzione che restituisce/scrive la word 2 dell'argomento con l'estensione del segno
sw3 641() / sw3 778()	R	T	F/I	Funzione che restituisce/scrive la word 3 dell'argomento con l'estensione del segno
tobcd 645()	R	T	F	Converte il valore da binario a BCD

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
tobin 646()	R	T	F	Converte il valore da BCD a binario
todeg 646()	R	T	F	Converte il valore da radianti a gradi
torad 647()	R	T	F	Converte il valore da gradi a radianti
w0 671() / w0 787()	R	T	F/I	Funzione che restituisce/scrive la word 0 dell'argomento
w1 672() / w1 787()	R	T	F/I	Funzione che restituisce/scrive la word 1 dell'argomento
w2 672() / w2 788()	R	T	F/I	Funzione che restituisce/scrive la word 2 dell'argomento
w3 673() / w3 788()	R	T	F/I	Funzione che restituisce/scrive la word 3 dell'argomento
wbit 674()	R	T	F	Set/Reset di bit in una variabile intera

NOTA: Se si è alla ricerca di funzioni/istruzioni per leggere/scrivere dei dati in un buffer vedi gestione conversione endianess 125.

Gestione flash

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la gestione della flash (indipendente dal tipo di hardware).

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
f_avail 318()	-	T	F	Memoria disponibile su CF
f_close 318()	-	T	F	Chiusura di un file aperto in CF

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
f_copy 319()	-	T	F	Copia il contenuto di un file in un file nuovo
f_del 320()	-	T	F	Cancellazione di un file sulla CF
f_dir 321()	-	T	F	Informazioni sui file contenuti nella CF
f_format 323()	-	T	F	Formattazione partizioni Flash
f_gets 324()	-	T	F	Legge da CF un file aperto in modo testo
f_mkdir 326()	-	T	F	Crea una nuova directory in CF
f_open 327()	-	T	F	Apertura di un file in CF
f_puts 329()	-	T	F	Scrive in un file sulla CF aperto in modo testo
f_read 330()	-	T	F	Legge da CF un file aperto in modo binario
f_read_table_s 333()	-	T	F	Legge da CF un file ascii e carica i valori in esso contenuti in variabili (controlla eventuali sbordamenti dal buffer)
f_read_table 331()	-	T	F	Legge da CF un file ascii e carica i valori in esso contenuti in variabili
f_rename 335()	-	T	F	Rinomina il nome di un file
f_seek 336()	-	T	F	Sposta il puntatore all'interno di un file sulla CF

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
f_write 337()	-	T	F	Scrive in un file sulla CF aperto in modo binario

Gestione forzatura I/O

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la forzatura degli I/O.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
force 347()	R	T	F	Funzione per forzare lo stato degli IO
release 560()	R	T	F	Funzione per rilasciare la forzatura degli IO
is_forced 384()	R	T	F	Funzione per controllare la forzatura degli IO

Gestione modbus/TCP client

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la gestione del protocollo Modbus/TCP client.

Legenda: FC = Function Code - R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)					
Keyword	FC	R	T	F/I	Descrizione
mbc_connect 391()	-	-	T	F	Connette il client al server Modbus/TCP
mbc_init 391()	-	-	T	F	Inizializza client Modbus/TCP
mbc_mask_write_reg 394()	22	-	T	F	Scrittura registers con maschera

© 2025 Robox SpA

Legenda: FC = Function Code - R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)

Keyword	FC	R	T	F/I	Descrizione
mbc_mask_write_reg_s396()	22	-	T	F	Scrittura registers con maschera tramite struttura di appoggio
mbc_read_coils398()	01	-	T	F	Lettura coils
mbc_read_coils_s399()	01	-	T	F	Lettura coils con struttura di appoggio
mbc_read_discr_inp400()	02	-	T	F	Lettura discrete inputs
mbc_read_discr_inp_s402()	02	-	T	F	Lettura discrete inputs con struttura di appoggio
mbc_read_hold_reg403()	03	-	T	F	Lettura holding registers
mbc_read_hold_reg_s405()	03	-	T	F	Lettura holding registers con struttura di appoggio
mbc_read_inp_reg406()	04	-	T	F	Lettura input registers
mbc_read_inp_reg_s408()	04	-	T	F	lettura input registers con struttura di appoggio
mbc_shutdown410()	-	-	T	F	Chiude la connessione con il server Modbus/TCP
mbc_write_coils410()	05 (singolo coil) 15 (coils > 1)	-	T	F	Scrittura coils
mbc_write_coils_s412()	05 (singolo coil) 15 (coils > 1)	-	T	F	Scrittura coils tramite struttura di appoggio

© 2025 Robox SpA

Legenda: FC = Function Code - R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)					
Keyword	FC	R	T	F/I	Descrizione
mbc_write_multiple_coils 413()	15	-	T	F	Scrittura coils usando codice funzione Modbus delle scritture multiple corrispettive
mbc_write_multiple_coils_s 414()	15	-	T	F	Scrittura coils tramite struttura di appoggio usando codice funzione Modbus delle scritture multiple corrispettive
mbc_write_multiple_regs 416()	16	-	T	F	Scrittura registers usando codice funzione Modbus delle scritture multiple corrispettive
mbc_write_multiple_regs_s 418()	16	-	T	F	Scrittura registers tramite struttura di appoggio usando codice funzione Modbus delle scritture multiple corrispettive
mbc_write_reg 419()	06 (singolo reg) 16 (reg > 1)	-	T	F	Scrittura registers
mbc_write_reg_s 421()	06 (singolo reg) 16 (reg > 1)	-	T	F	Scrittura registers tramite struttura di appoggio
mbcu_mask_write_reg 422()	22	-	T	F	Scrittura registers con maschera specificando lo unitID

© 2025 Robox SpA

Legenda: FC = Function Code - R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)					
Keyword	FC	R	T	F/I	Descrizione
mbcu_read_coils 424()	01	-	T	F	Lettura coils specificando lo unitID
mbcu_read_discr_inp 426()	02	-	T	F	Lettura discrete inputs specificando lo unitID
mbcu_read_hold_reg 427()	03	-	T	F	Lettura holding registers specificando lo unitID
mbcu_read_inp_reg 429()	04	-	T	F	Lettura input registers specificando lo unitID
mbcu_write_coils 431()	05 (singolo coil) 15 (coils > 1)	-	T	F	Scrittura coils specificando lo unitID
mbcu_write_multiple_coils 433()	15	-	T	F	Scrittura coils specificando lo unitID usando codice funzione Modbus delle scritture multiple corrispettive
mbcu_write_multiple_regs 434()	16	-	T	F	Scrittura registers specificando lo unitID usando codice funzione Modbus delle scritture multiple corrispettive
mbcu_write_reg 436()	06 (singolo reg) 16 (reg > 1)	-	T	F	Scrittura registers specificando lo unitID

NOTA: Il Function Code (FC) è un codice specifico utilizzato in una richiesta Modbus per dire al dispositivo slave Modbus a quale tipo di memoria (HOLDING REGISTERS, INPUT COILS, ...) accedere e quale azione eseguire su quella memoria (lettura o scrittura).

© 2025 Robox SpA

Gestione MOT

Elenco di tutte le funzioni/istruzioni per la gestione delle MOT (Multiple Output Table) disponibili nel linguaggio di programmazione R3.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)

Keyword	R	T	F/I	Descrizione
mot_append 439()	R	T	F	Aggiunta di un nuovo punto alla MOT
mot_clear 443()	R	T	F	Rimozione di tutti i punti della MOT
mot_create 443()	R	T	F	Creazione di una nuova MOT
mot_delete 447()	R	T	F	Cancellazione della MOT
mot_get 447()	R	T	F	Lettura dei dati relativi ad un punto della MOT
mot_remove 449()	R	T	F	Rimozione di uno o più punti della MOT
mv_mot_exec 470()	R	T	F	Esecuzione della MOT

Gestione movimento

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la movimentazione degli assi.

NOTA: Fare riferimento al manuale della libreria RPE (Robox Path Executor) se si utilizza un gruppo di assi.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)

Keyword	R	T	F/I	Descrizione
mv_cam 451()	R	-	F	Esegue una camma cubica per tratti a,b,c
mv_crimper 453()	R	-	F	Funzione crimper per flow-pack
mv_follow 454()	R	-	F	Taglio al volo (è consigliata la follow2)
mv_follow2 458()	R	-	T	Taglio al volo (master e slave

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				non associati a un asse)
mv_mot_exec 470()	R	T	F	Funzione di interpolazione mediante tabella MOT (multiple output table)
mv_phase_adj 472()	R	-	F	Correzione di fase tra due assi
mv_phase_adj2 474()	R	-	F	Correzione di fase tra due assi
mv_ramp 477()	R	T	F	Deprecata. Utilizzare ramp 555()
mv_reach_target 477()	R	-	F	Aggancio tra due assi
mv_sine_cam 478()	R	-	F	Esecuzione di una camma (y in funzione di x)
mv_synchro 479()	R	-	F	Aggancio tra due assi
mv_table 481()	R	-	F	Esegue una Camma per punti
mv_tracking 491()	R	-	F	Deprecata. Usare tracking 648()
mv_to 482()	R	-	F	Movimento verso un target
mv_to_vel 489()	R	-	F	Movimento verso un target con una velocità finale specificata
mv_to_cj 486()	R	-	F	Movimento verso un target con jerk controllato (parametri acquisiti solo su start)
mv_to_cjv 487()	R	-	F	Movimento verso un target con jerk controllato (parametri modificabili run-time)

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
mv_to_cjv_info 489 ()	R	T	F	Funzione di ausilio a MV_TO_CJV
mva_follow2 467()	R	-	F	Taglio al volo (solo slave associato a un asse)
mva_follow2_m 469 ()	R	-	F	Taglio al volo (master e slave associati a un asse)
mva_jog 491()	R	-	F	Movimentazione JOG (manuale)
mva_jog2 493()	R	-	F	Movimentazione JOG (manuale)
mva_open_loop 496()	R	-	F	Apertura dell'anello
mva_to_n 496()	R	-	F	Movimento verso un target con più assi
mva_to_n_cj 500()	R	-	F	Movimento verso un target con più assi
mva_to_n_v 504()	R	-	F	Movimento verso un target con più assi anche già in movimento
mva_zc 511()	R	-	F	Movimento di Homing
kin_conv 386()	R	T	F	Calcola i valori di accelerazione e di jerk
set_kinematics 595 ()	R	-	F	Impone le grandezze cinematiche contemporaneamente
tracking 648()	R	T	F	Porta la grandezza controllata verso il target specificato

© 2025 Robox SpA

Gestione power set

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per l'interazione con il Power set.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)

Keyword	R	T	F/I	Descrizione
exl_lam316()	R	T	F	Mostra gli assi logici inclusi/esclusi di un Power set
excl_lam711	R	T	I	Include/esclude gli assi logici di un Power set
excl_pam317()	R	T	F	Mostra gli assi fisici inclusi/esclusi di un Power set
excl_pam712	R	T	I	Include/esclude gli assi fisici di un Power set
ps_exclude_lax538()	R	T	F	Esclude gli assi logici di un Power set
ps_exclude_pax539()	R	T	F	Esclude gli assi fisici di un Power set
ps_reinclude_lax541()	R	T	F	Include gli assi logici di un Power set
ps_reinclude_pax(542()	R	T	F	Include gli assi fisici di un Power set
ps_power_enable540()	R	T	F	Power set (gestione potenza) -Abilita la potenza del Power set-
ps_channel_enable526()	R	T	F	Power set (gestione potenza) -Abilita gli assi del Power set-
ps_channel_reset_fault532()	R	T	F	Power set (gestione potenza) -Invia un comando di reset allarme-

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
ps_drive_reset 535 ()	R	T	F	Power set (gestione potenza) -Invia un comando di reset drive-
ps_status 543()	R	T	F	Power set (gestione potenza) -Rende informazioni sullo stato del Power set-
ps_channel_enabled 529()	R	T	F	Power set (gestione potenza) -Rende informazioni circa gli assi abilitati di un Power set-
ps_channel_fault 531()	R	T	F	Power set (gestione potenza) -Rende informazioni circa gli assi in fault di un Power set-

Gestione stringhe

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la manipolazione delle stringhe ASCII.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
ipaddr_to_str 378()	R	T	F	Converte un indirizzo IP da intero a stringa
str_asc 624()	R	T	F	Converte un carattere in numero
str_chr 625()	R	T	F	Converte un numero in carattere
str_compact 625()	R	T	F	Compatta gli spazi
str_equal 626()	R	T	F	Confronta due stringhe

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
str_find 627()	R	T	F	Cerca all'interno di una stringa
str_format 627()	R	T	F	Formatta la stringa (sprintf)
str_i 629()	R	T	F	Converte un valore numerico intero in ASCII
str_insert 630()	R	T	F	Inserisce all'interno di una stringa
str_len 631()	R	T	F	Calcola la lunghezza della stringa
str_mid 631()	R	T	F	Prende nella stringa
str_r 632()	R	T	F	Converte un valore numerico reale in ASCII
str_replace 633()	R	T	F	Sostituisce all'interno di una stringa
str_val 635()	R	T	F	Rende, in formato numerico, un valore contenuto in una stringa.
str_to_ipaddr 634()	R	T	F	Converte una stringa contenente l'IP in un valore numerico
tolower 647()	R	T	F	Converte un carattere in minuscolo
toupper 648()	R	T	F	Converte un carattere in maiuscolo

© 2025 Robox SpA

Gestione timer

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la gestione dei timer.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
tm_on 782()	R	T	I	Timer ON
tm_of 780()	R	T	I	Timer OFF
tm_tp 786()	R	T	I	Timer impulso
tm_sqw 784()	R	T	I	Timer onda quadra
tm_int 779()	R	T	I	Timer integrale
Funzioni di interazione:				
tm_et 642() / tm_et 779()	R	T	F/I	Legge/setta il tempo del timer
tm_pt 643() / tm_pt 784()	R	T	F/I	Legge/setta il preset del timer
tm_pt2 644() / tm_pt2 784()	R	T	F/I	Legge/setta il secondo preset del timer (timer square)
tm_q 645()	R	T	F	Uscita del timer (bobina)
tm_reset 784()	R	T	I	Resetta il tempo del timer (timer integrale)

Gestione UDP/IP

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la gestione del protocollo UDP/IP.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
ipaddr_to_str 378()	R	T	F	Converte indirizzo IP da intero a stringa
str_to_ipaddr 634()	R	T	F	Converte indirizzo IP da stringa a intero

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
udp_open_client 651()	R	T	F	Apre una connessione client UDP
udp_open_server 654()	R	T	F	Apre una connessione server UDP
udp_close 651()	R	T	F	Chiude una connessione UDP
udp_send 662()	R	T	F	Invia un pacchetto dati UDP
udp_send_to 667()	R	T	F	Invia un pacchetto dati UDP a...
udp_recv 657()	R	T	F	Riceve un pacchetto dati UDP
udp_recv_from 658()	R	T	F	Riceve un pacchetto dati UDP da...
udp_send_notify 665()	R	T	F	Invia un pacchetto dati UDP con notifica
udp_recv_notify 660()	R	T	F	Riceve un pacchetto dati UDP con notifica

Gestione variabili globali

Elenco di tutte le funzioni/istruzioni messe a disposizione dal linguaggio di programmazione R3 per la gestione delle variabili globali.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
gv_create 359()	-	T	F	Crea una nuova variabile globale. Crea le informazioni in memoria e, se necessario, nella memoria ritentiva

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
gv_destroy363()	-	T	F	Rimuove una variabile globale. elimina il file di configurazione e lo spazio in memoria
gv_link368()	R	T	F	Link a una variabile globale esistente. Crea un'interfaccia a una GV
gv_unlin373	R	T	F	Rimuove la connessione dell'interfaccia a una variabile globale
gv_read370()	R	T	F	Aggiorna il dato della variabile locale associata tramite l'interfaccia, leggendo dalla GV
gv_read_partial371()	R	T	F	Aggiorna parzialmente il dato della variabile locale associata tramite l'interfaccia, leggendo dalla GV
gv_write374()	R	T	F	Aggiorna il valore della variabile globale con il valore della variabile locale associata tramite l'interfaccia
gv_write_partial375()	R	T	F	Aggiorna parzialmente il valore della variabile globale con il valore della variabile locale associata tramite l'interfaccia
gv_lock369()	R	T	F	Blocca l'accesso della variabile globale -

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				l'accesso non è permesso da nessun'altra interfaccia
gv_unlock 373()	R	T	F	Sblocca l'accesso della variabile globale - l'accesso è quindi permesso da tutte le interfacce
gv_exist 364()	R	T	F	Controlla se la stringa specificata coincide con una variabile globale definita
gv_name 370()	R	T	F	Ottiene il nome di una variabile globale
gv_datatype 362()	R	T	F	Ottiene il tipo di una variabile globale
gv_datatype_byname 362()	R	T	F	Ottiene il tipo di una variabile globale
gv_flags 364()	R	T	F	Ottiene il flag associato a una variabile globale
gv_flags_byname 366()	R	T	F	Ottiene il flag associato a una variabile globale
gv_size 372()	R	T	F	Ottiene la dimensione di una variabile globale
gv_size_byname 372()	R	T	F	Ottiene la dimensione di una variabile globale
gv_update_file 374()	-	T	F	Aggiorna le informazioni nel file di configurazione

© 2025 Robox SpA

Gestioni miscellanee

Elenco di tutte le funzioni/istruzioni miscellanee messe a disposizione dal linguaggio di programmazione R3.

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)

Keyword	R	T	F/I	Descrizione
check_mv_follow2 230()	R	T	F	Controlla se la funzione follow2() può essere eseguita
check_mva_follow2 233()	R	T	F	Controlla se la funzione follow2() può essere eseguita
check_mva_follow2_m 236()	R	T	F	Controlla se la funzione follow2() può essere eseguita
command 288()	R	T	F	Esegue un comando (direttiva)
dec 295()	R	T	F	Decrementa una variabile di una quantità specificata
delta_enc16 296()	R	T	F	Calcola differenza di due grandezze a 16 bit gestendo lo scaravolto
delta_enc32 297()	R	T	F	Calcola differenza di due grandezze a 32 bit gestendo lo scaravolto
diff 299()	R	T	F	Rende vero se il valore della variabile è cambiato
fall 338()	R	T	F	Coglie il fronte di discesa di una condizione
fall_filt 339()	R	T	F	Coglie il fronte di discesa di una condizione filtrato
filter_lp 341()	R	-	F	Filtro passa basso sulla

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				variabile
filter_lp_st 342()	R	-	F	Filtro passa basso sulla variabile utilizzabile in un ciclo for
filter_notch 343()	R	-	F	Filtro notch sulla variabile
filter_notch_st 345()	R	-	F	Filtro notch sulla variabile utilizzabile in un ciclo for
get_hw_info 351()	-	T	F	Permette di avere informazioni relative alle schede direttamente connesse alla CPU
get_virt_ow 359()	R	T	F	Ottiene lo stato di output word virtuale
in_range 376()	R	T	F	Restituisce il valore nel range indicato
inc 377()	R	T	F	Incrementa una variabile di una quantità specificata
index_of 377()	R	T	F	restituisce l'indice della variabile simbolica
lit 729	R	T	I	Associazione di un simbolico ad una stringa di testo
loop_time 389()	R	T	F	Calcola la durata temporale tra due chiamata successive
maximum 390()	R	T	F	Restituisce il più grande tra due valori

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
master_filter 389()	R	T	F	Filtra la velocità di un master
minimum 438()	R	T	F	Restituisce il più piccolo tra due valori
normalize 513()	R	T	F	Restituisce valore normalizzato tra 0 e 1
onoff_filt 514()	R	T	F	Coglie i fronti di salita e discesa filtrati di una condizione
out_m 737()	R	T	I	Modifica lo stato dei canali di output
par_even 516()	R	T	F	Funzione che rende vero se il numero di bit in 1 del byte passatogli è pari
par_odd 517()	R	T	F	Funzione che rende vero se il numero di bit in 1 del byte passatogli è dispari
planar_comp_cp 518()	R	-	F	Applica la compensazione inversa nella gestione della Planar Compensation Table
planar_comp_ip 519()	R	-	F	Applica la compensazione diretta nella gestione della Planar Compensation Table
pos_capture_cmd 520()	R	T	F	Funzione per catturare hardware la quota di un asse

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
pos_capture_sts 524()	R	T	F	Funzione di supporto per catturare hardware la quota di un asse
ramp 555()	R	T	F	Rampa la grandezza verso un valore target
rise 562()	R	T	F	Coglie il fronte di salita di una condizione
rise_filt 563()	R	T	F	Coglie il fronte di salita di una condizione filtrato
rtc_date 565()	R	T	F	Setta la data del clock calender
rtc_time 566()	R	T	F	Setta l'orario del clock calender
set_virt_iw 595()	R	T	F	Setta lo stato di input word virtuale
set_version 776()	R	T	I	Permette di scrivere la versione software dell'applicativo
similar 601()	R	T	F	Confronta due variabili e stabilisce se sono simili
statistics 621()	R	T	F	Esegue statistiche sulla variabile indicata
store_nvr_nopf 622()	R	T	F	Salva i NVR specificati nella memoria statica
store_nvrr_nopf 623()	R	T	F	Salva i NVRR specificati nella memoria statica
time_micro 642()	R	T	F	Restituisce il tempo in microsecondi

© 2025 Robox SpA

Legenda: R = ammesso uso nelle Rule - T = ammesso uso nei Task - I = Istruzione - F = Funzione (ha quindi un valore di ritorno)				
Keyword	R	T	F/I	Descrizione
				dall'accensione del controllo assi
virt_io 669()	-	T	F	Definisce una zona di input e output word virtuali
write_dis 676()	R	T	F	Scrive una stringa sul display 4*20

Funzioni

Funzione abs()

Questa funzione esegue il valore assoluto del valore specificato.

Sintassi	**i8|i16|i32|real abs (i8|i16|i32|real value)**
value	Argomento della funzione.
Risultato	Rende il valore assoluto di *value*.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; il risultato della divisione 1230/(-5) viene salvato nel registro re
rr(3) = 1230.0 / (-5)
; il valore assoluto del contenuto in rr(3) viene salvato nel registro
rr(4) = abs 156(rr(3))
; valore dei registri dopo l'esecuzione delle istruzioni:
; rr (3) = -246
; rr (4) = 246
```

© 2025 Robox SpA

Funzione abs_mod()

Questa funzione rende il modulo del valore di ingresso, opportunamente modificato in caso di argomento entrante negativo.

Sintassi	**real abs_mod (real value, real step)**
value	È la grandezza reale di cui si vuole calcolare il modulo (può essere un'espressione reale generica).
step	Valore modulante (può essere un'espressione reale generica, ma deve sempre dare un valore positivo).
Risultato	Rende: Condizione: *value* >= 0 — mod(*value*, *step*) Altrimenti. — mod(*value*, *step*) + *step*
Validità	Rule, Task.
Note	Questa formula permette di costruire in uscita un "dente di sega" che si prolunga uniformemente anche per valori negativi della grandezza entrante.

Esempi

Esempio di utilizzo:

```
; viene incrementato di 1 il contenuto del registro rr(1)
rr(1) = inc(rr(1))
; in rr(2) viene salvato il modulo di rr(1) con 234
rr(2) = abs_mod(rr(1), 234)
```

Funzione acos()

Esegue l'arcocoseno di un valore di tipo reale.

Sintassi	**real acos (real value)**
value	Argomento dell'arcocoseno (compreso tra -1 e 1).
Risultato	Rende l'arcocoseno di *value*. Il risultato è reso in radianti ed è compreso fra 0 e K_PI.
Validità	Rule, Task.
Note	

© 2025 Robox SpA

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se argomento < -1 (dove x = numero di task e xxx numero di step)
- Genera 9900 ill. Arg A:2 T:x ST:xxx se argomento > 1 (dove x = numero di task e xxx numero di step)

Esempi

Esempio di utilizzo:

```
; in rr(10) viene messo l'arcocoseno del valore di rr(5) convertito da
rr(10) = todeg(acos(rr(5)))
; in rr(10) viene messo l'arcocoseno di 0.5
rr(10) = acos(0.5)
```

Funzione ah_cmd()

Questa funzione interagisce con la gestione dello storico allarmi di RTE.

Sintassi	**i32 ah_cmd (i32 cmdMsk)**
cmdMsk	Significato maschera a bit:

Bit	Valore esadecimale	Descrizione
0	0x00000001	Comando di clear alarm history.
1 -:- 31	...	Riservati.

Risultato	Rende:

Valore	Descrizione
0 [ACK_OK]	Tutti i comandi sono stati riconosciuti ed eseguiti.
101 [NACK_ILL_ARGS]	I comandi riconosciuti sono stati eseguiti, ma ci sono dei comandi sconosciuti.
133 [NACK_NOT_FOUND]	Storico allarmi non abilitato.

Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**i32 ah_cmd (i32 cmdMsk)**
Note	

Esempi

Esempio di utilizzo:

```
; richiesta di reset storico allarmi
_if(resetAH)
    ah_cmd 158(1)
```

Funzione ah_get_d()

Questa funzione permette di avere informazioni relative agli allarmi memorizzati nello storico allarmi con restituzione delle informazioni in una variabile di tipo stru_ah_d 823 o stru_ah_d_ext 824.

Quest'ultima struttura supporta variabili stringa utente USV.

Sintassi	**i32 ah_get_d (i32 ahId, stru_ah_d| stru_ah_d_ext stru)**
ahId	Numero dell'ID dell'allarme di cui si vuole informazioni (questo tipo di informazione si ottiene con ah_info 162()).
stru	Nome della struttura stru_ah_d 823 o stru_ah_d_ext 824 in cui RTE mette i dati di risposta.
Risultato	Rende:

Valore	Descrizione
0 [ACK_OK]	Esito ok.
107 [NACK_RES_BUSY]	Gestore storico allarmi occupato.
127 [NACK_READ_ERROR]	Dati non validi in memoria ritentiva.
133 [NACK_NOT_FOUND]	Storico non abilitato.
143 [NACK_DIFFERENT_DATA]	Il dato richiesto è stato sovrascritto da un nuovo allarme.

Validità	Rule, Task.
Note	

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
; dichiarazione variabile di tipo STRU_AH_D
stru_ah_d 823 sInfoAll

; ...
; richiesta informazioni circa l'id 123 con restituzione delle informa
ah_get_d 159(123, sInfoAll)
```

Funzione ah_get_s()

Questa funzione permette di avere informazioni relative agli allarmi memorizzati nello storico allarmi con restituzione delle informazioni in una variabile di tipo string 929.

Sintassi	**i32 ah_get_s (i32 ahId, string str [, i32 lenMax=str])**
ahId	Numero dell'ID dell'allarme di cui si vuole informazioni (questo tipo di informazione si ottengono con ah_info 162()).
str	Nome della variabile di tipo stringa.
lenMax	Lunghezza massima della stringa. (opzionale, default la dimensione della variabile stringa *str*)
Risultato	Rende:
Validità	Rule, Task.
Note	

Valore	Descrizione
0 [ACK_OK]	Esito ok.
107 [NACK_RES_BUSY]	Gestore storico allarmi occupato.
127 [NACK_READ_ERROR]	Dati non validi in memoria ritentiva.
133 [NACK_NOT_FOUND]	Storico non abilitato.
143 [NACK_DIFFERENT_DATA]	Il dato richiesto è stato sovrascritto da un nuovo allarme.

In caso di esito positivo la stringa ottenuta ha il seguente formato :"n - d(dd) m(mm) y(yyyy) hh:ii:ss alarmText".

Parametro	Descrizione
n	Id dell'allarme (solo le cifre necessarie).

© 2025 Robox SpA

Parametro	Descrizione
dd	Giorno.
mm	Mese.
yyyy	Anno.
hh	Ora.
ii	Minuti.
ss	Secondi.
alarmText	Testo dell'allarme.

Nota: è possibile personalizzare il formato di data e ora tramite la variabile predefinita date_format.

Esempi

Esempio di utilizzo:

```
bool get_alarm_info
string alarm_info
int result_ah_get
int id_alarm

lit NACK_RES_BUSY 107
lit NACK_READ_ERROR 127
lit NACK_NOT_FOUND 133
lit NACK_DIFFERENT_DATA 143

; ...
if(get_alarm_info)
    get_alarm_info = false
    result_ah_get = ah_get_s(id_alarm, alarm_info)

    if(not result_ah_get)
        ; everything okay
        inf_report(0x20, "Alarm info retrived from AH with ah_get_s
        inf_report(0x20, alarm_info)
    else
        select(result_ah_get)

            case NACK_RES_BUSY
                inf_report(0x80, "ERROR: ah_get_s() AH handler busy")
                break

            case NACK_READ_ERROR
                inf_report(0x80, "ERROR: ah_get_s() error reading rete
                break

            case NACK_NOT_FOUND
                inf_report(0x80, "ERROR: ah_get_s() alarm history not
                break
```

© 2025 Robox SpA

```
            case NACK_DIFFERENT_DATA
                inf_report(0x80, "ERROR: ah_get_s() info requested hav
                break

            default
                str_format(rep_msg, "ERROR: ah_get_s() returned unkv
                inf_report(0x80, rep_msg)

        endselect
    endif
endif
```

Funzione ah_info()

Questa funzione permette di ottenere informazioni relative agli allarmi memorizzati nello storico allarmi con restituzione dei dati nella variabile stru_ah_info.

Sintassi	**i32 ah_info (stru_ah_info stru)**
stru	Nome della struttura stru_ah_info dove RTE mette i dati di risposta.
Risultato	Rende:

Valore	Descrizione
0 [ACK_OK]	Esito ok.
133 [NACK_NOT_FOUND]	Storico non abilitato.

Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bool get_ah_info
int result_ah_info
stru_ah_info info_alarm_histrory
string rep_msg

; ...
if(get_ah_info)
    get_ah_info = false
    result_ah_info = ah_info(info_alarm_histrory)

    if(not result_ah_info)
        inf_report(0x20, "Alarm histrory info:")
        str_format(rep_msg, "AH old id: %i", info_alarm_histrory.old
```

© 2025 Robox SpA

```
        inf_report(0x20, rep_msg)
        str_format(rep_msg, "AH new id: %i", info_alarm_histrory.newId
        inf_report(0x20, rep_msg)
        str_format(rep_msg, "AH num al: %i", info_alarm_histrory.nal)
        inf_report(0x20, rep_msg)
        str_format(rep_msg, "AH size: %i", info_alarm_histrory.ahSize)
        inf_report(0x20, rep_msg)
    else
        inf_report(0x80, "### Alarm history not enabled! ###")
    endif
endif
```

Funzione ah_log()

Istruzione che permette di aggiungere dei messaggi direttamente nello storico allarmi.

Sintassi	**i32 ah_log (i32 code [, par1=0] [, par2=0] [, par3=0])**
code	Numero dell'allarme. Deve essere compreso nel range di validità dei messaggi (10000-:-12999).
par1	Primo parametro passato al messaggio. (opzionale, default 0)
par2	Secondo parametro passato al messaggio. (opzionale, default 0)
par3	Terzo parametro passato al messaggio. (opzionale, default 0)
Risultato	Rende: Valore / Descrizione 0 [ACK_OK] — Esito ok. 101 [NACK_ILL_ARGS] — Il numero di allarme non è nel range previsto. In questo caso viene generato l'allarme 9900 illegal Arg. 133 [NACK_NOT_FOUND] — Storico non abilitato.
Validità	Rule, Task.
Note	L'utilizzo dei tre parametri e/o il loro relativo tipo dipendono da come è stato configurato il messaggio.

© 2025 Robox SpA

La struttura dei messaggi è equivalente a quella degli allarmi (vedi user_alarm_file per la configurazione).

Esempi

Esempio di utilizzo:

```
; in alarm definition file
10007 0  -1 0x23002000 "Info with a INT parameter [%d]" -4  ; in this

; in r3 software
int resoAHlog
int parameter

;...
    if(inp(1))
        parameter = 1
    else
        parameter = 2
    endif
resoAHMsg = ah_log(10007, parameter)
```

Funzione ah_log_us()

Istruzione che permette di aggiungere dei messaggi con parametri stringa utente USV direttamente nello storico allarmi.

Sintassi	**i32 ah_log_us (i32 nAl [, par1=0] [, par2=0] [, par3=0])**
nAl	Numero dell'allarme. Deve essere compreso nel range di validità dei messaggi (10000-:-11999).
s1	Stringa utente da passare come primo parametro. (opzionale, default 0)
s2	Stringa utente da passare come secondo parametro. (opzionale, default 0)
s3	Stringa utente da passare come terzo parametro. (opzionale, default 0)
Risultato	Rende: Valore / Descrizione 0 — Esito ok. 101 — Il numero di allarme non è nel range previsto. In questo caso viene generato l'allarme 9900 illegal Arg A:1.

© 2025 Robox SpA

Sintassi	**i32 ah_log_us (i32 nAl [, par1=0] [, par2=0] [, par3=0])**
	Valore: 103 — Descrizione: Gestore allarmi non attivo.
Validità	Rule, Task.
Requisiti di sistema	Da RTE 34.30.0.
Note	

Esempi

Esempio di utilizzo:

```
string s1
string s2
int result
...
; set an axis log
str_format (s1, "ax:%d Broken Brake", 3)
result = ah_log_us(11101, s1)
...
...
; set an axis log
str_format (s1, "ax:%d Broken Drive", 3)
result = ah_log_us(11101, s1)
...
...
; set log about Emergency push botton pressed
str_format (s1, "Emergency Pressed")
...
str_format (s2, "near first door")
result = ah_log_us(11102, s1, s2)
...
str_format (s2, "end of line")
result = ah_log_us(11102, s1, s2)


; alarm definition in user file
11101 0 -1 0x23004000 "Log 1 <%s>" 17 0 0
11102 0 -1 0x23004000 "Log 2 <%s:%s>" 17 17 0
```

© 2025 Robox SpA

Funzione al2am()

Questa istruzione permette di avere informazioni sulla maschera allarmi AM, in funzione del numero dell'allarme e dell'eventuale numero di asse.

<table>
<tr><td>Sintassi</td><td>i32 al2am (i32 nAl , i32 nAx, i32 whichInfo)</td></tr>
<tr><td>nAl</td><td>Numero dell'allarme.</td></tr>
<tr><td>nAx</td><td>Numero dell'asse (1-:-32) o 0 se l'allarme non è di asse.</td></tr>
<tr><td>whichInfo</td><td>Tipo di informazione sulla maschera allarmi (AM) voluta:
<table>
<tr><td>Valore whichInfo</td><td>Descrizione</td></tr>
<tr><td>0</td><td>Stato del bit dell'AM relativa all'allarme.</td></tr>
<tr><td>1</td><td>Indice dell'AM relativa all'allarme.</td></tr>
<tr><td>2</td><td>Bit dell'indice dell'AM relativa all'allarme.</td></tr>
<tr><td>3</td><td>Bit dell'indice dell'AM relativa all'allarme, sottoforma di maschera.</td></tr>
</table></td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><td>Condizione</td><td>Valore</td><td>Descrizione</td></tr>
<tr><td rowspan="6">Errore.</td><td>-1</td><td>Numero allarme illegale.</td></tr>
<tr><td>-2</td><td>Numero asse illegale.</td></tr>
<tr><td>-3</td><td>Tipo di informazione richiesta illegale.</td></tr>
<tr><td>-4</td><td>L'allarme specificato non è un allarme di asse.</td></tr>
<tr><td>-5</td><td>L'allarme specificato è un allarme di asse, o non è associata alcuna AM.</td></tr>
<tr><td>-6</td><td>L'associazione al bit dell'AM</td></tr>
</table></td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 al2am (i32 nAl , i32 nAx, i32 whichInfo)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Condizione</th><th>Valore</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>non è definita.</td></tr>
<tr><td>whichInfo = 0</td><td>0</td><td>Stato del bit dell'AM relativo all'allarme basso (ovvero allarme non presente).</td></tr>
<tr><td></td><td>1</td><td>Stato del bit dell'AM relativo all'allarme alto (ovvero allarme presente).</td></tr>
<tr><td>whichInfo = 1</td><td>>= 0</td><td>Indice dell'AM corrispondente all'allarme scelto.</td></tr>
<tr><td>whichInfo = 2</td><td>>= 0</td><td>Bit dell'indice dell'AM corrispondente all'allarme scelto.</td></tr>
<tr><td>whichInfo = 3</td><td>>= 0</td><td>Bit dell'indice dell'AM corrispondente all'allarme scelto, sottoforma di maschera.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td></td></tr>
</table>

Esempi

Esempio di utilizzo:

```
; da task chiamo la funzione per avere informazioni sull'allarme 9900,
result = al2am(9900, 0, whichInfo)

; casi possibili
• whichInfo = 0 -> result = 1 se l'allarme 9900 è presente, altrimenti
• whichInfo = 1 -> result = 49
• whichInfo = 2 -> result = 27
• whichInfo = 3 -> result = 134217728 (0x8000000)
```

Esempio di utilizzo:

© 2025 Robox SpA

```
u32 SUM_MAC_DIAGNO, appo, mB
i32 nAX, iB
; ...
appo = 0
; ...
; ottengo l'indice del bit del TRANSDUCER ALARM (allarme 10)
iB = al2am 166(10, 1, 2)

for(nAX = 1 , nAX <= 6 , nAX)
    ; controllo se è presente l'allarme trasduttore
    if(AM(nAX).(iB))
        ; setto un bit che indica che almeno un asse ha allarme trasdu
        appo.(10) = true
        break
    endif
endfor

; ottengo l'indice del bit del FOLLOWING ALARM (allarme 11)
iB = al2am(11, 1, 2)

for(nAX = 1 , nAX <= 6 , nAX)
    ; controllo se è presente l'allarme inseguimento
    if(AM(nAX).(iB))
        ; setto un bit che indica che almeno un asse ha allarme insegu
        appo.(11) = true
        break
    endif
endfor

; la gestione dell'informazione è la stessa di prima, ma ora viene usa
; Prendo il bit dell'allarme UTENTE 1500 sottoforma di maschera
mB = al2am(1500, 1, 3)

for(nAX = 1 , nAX <= 6 , nAX)
    ; controllo se è presente l'allarme 1500
    if(AM(nAX) r_and mB)
        ; setto un bit che indica che almeno un asse ha allarme 1500
        appo.(12) = true
        break
    endif
endfor
```

© 2025 Robox SpA

Funzione alarm_reset()

Questa funzione permette di cancellare, tutti o uno in particolare, gli allarmi dallo stack allarmi.

Sintassi	**bool alarm_reset ([i32 nAl=0] [, i32 nAx=0])**
nAl	Numero dell'allarme (1-:-9999). (opzionale, default 0)
nAx	Numero dell'asse in allarme (1-:-32). (opzionale, default 0)
Risultato	Rende: Valore: TRUE — Allarme resettato. Valore: FALSE — Riscontrati parametri illegali.
Validità	Rule, Task.
Note	L'operazione di reset comporta: • cancellazione dell'allarme nello stack (se presente) ed eventuale compattazione • azzeramento del bit dell'allarme nell'AM • inizializzazione del tempo di filtro per accedere allo storico allarmi (ah_filter_time()) • adeguamento delle variabili predefinite al_pres(), al_stack_mask() e ax_in_alarm() (se necessario)

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se numero di allarme è in campo riservato al sistema (dove x = numero di task e xxx numero di step)
- Genera 9900 ill. Arg A:2 T:x ST:xxx se eventuale numero di asse non è in range 0-:-32 (dove x = numero di task e xxx numero di step)

Esempi

Esempi di utilizzo:

```
; reset di tutti gli allarmi presenti
alarm_reset 169()
; reset dell'allarme 123 per tutti gli assi
alarm_reset(123)
; reset dell'allarme 199 solo per il decimo asse
alarm_reset(199, 10)
; reset di tutti gli allarmi per l'asse 30
```

© 2025 Robox SpA

```
alarm_reset(0, 30)
```

Funzione alarm_reset_par()

Questa funzione permette di resettare un allarme specificando anche il valore di uno o più parametri.

Sintassi	**bool alarm_reset_par (i32 nAl, i32 nAx, u32 parMask, real par1 [, real par2=0] [, real par3=0])**
nAl	Numero dell'allarme (1-:-9999). (opzionale, default 0)
nAx	Numero dell'asse in allarme (1-:-32). (opzionale, default 0)
parMask	Maschera per indicare quali parametri sono interessati:

Bit	Valore esadecimale	Descrizione
0	0x00000001	Nessuno. Equivalente alla funzione alarm_reset 169 ().
1	0x00000002	par1
2	0x00000004	par2
3	0x00000008	par1 e par2
4	0x00000010	par3
5	0x00000020	par1 e par3
6	0x00000040	par2 e par3
7	0x00000080	par1, par2 e par3

par1	Valore del primo parametro dell'allarme.
par2	Valore del secondo parametro dell'allarme. (opzionale, default 0)
par3	Valore del terzo parametro dell'allarme. (opzionale, default 0)
Risultato	Rende:

Valore	Descrizione
TRUE	Allarme resettato.

© 2025 Robox SpA

Sintassi	**bool alarm_reset_par (i32 nAl, i32 nAx, u32 parMask, real par1 [, real par2=0] [, real par3=0])**
	<table><tr><th>Valore</th><th>Descrizione</th></tr><tr><td>FALSE</td><td>Riscontrati parametri illegali.</td></tr></table>
Validità	Rule, Task.
Note	L'operazione di reset comporta: • cancellazione dell'allarme nello stack (se presente) ed eventuale compattazione • azzeramento del bit dell'allarme nell'AM • inizializzazione del tempo di filtro per accedere allo storico allarmi (ah_filter_time()) • adeguamento delle variabili predefinite al_pres(), al_stack_mask() e ax_in_alarm() (se necessario)

Diagnostica

Genera allarme 9900:

Codice	Descrizione
1	Numero di allarme illegale.
443	Numero di allarme illegale.
2	Numero di asse illegale.
444	Numero di asse illegale.
445	Numero di asse illegale.
3	*nAl*=0 *nAx*=0 *parMask*<>0
446	*nAl*=0 *nAx*=0 *parMask*<>0
4	*nAl*=0 *nAx*<>0 *parMask*<>0
447	*nAl*=0 *nAx*<>0 *parMask*<>0
5	*nAl*<>0 *nAx*=0 (*nAl* è allarme d'asse) *parMask*<>0
448	*nAl*<>0 *nAx*=0 (*nAl* è allarme d'asse) *parMask*<>0

Esempi

Esempio di utilizzo:

© 2025 Robox SpA

```
; sample code
int nAl, nAx, a
u32 parMask
real pAl1, pAl2, pAl3

; ...
a = alarm_reset_par(nAl, nAx, parMask, pAl1, pAl2, pAl3)
```

Funzione alarm_reset_par_us()

Questa funzione permette di resettare un allarme dallo stack specificando il valore di uno o più parametri stringa utente USV.

Tutti gli allarmi nello stack che corrispondono ai parametri specificati verranno rimossi.

Se:

- il numero dell'allarme, il numero dell'asse e la parMask sono 0, viene eseguito un reset completo dello stack
- il numero dell'allarme e la parMask sono 0, vengono resettati tutti gli allarmi dell'asse specifico
- il numero dell'asse e la parMask sono 0 e il numero dell'allarme è un allarme di asse, tutti gli allarmi dei diversi assi vengono cancellati
- il numero dell'asse e la parMask sono 0 e il numero dell'allarme non è un allarme di asse, viene resettato l'allarme specifico
- la parMask è diversa da 0, l'allarme specifico viene resettato se il numero dell'allarme, il numero dell'asse (compreso 0) e i parametri corrispondono

<table>
<tr><td>Sintassi</td><td>bool alarm_reset_par_us (i32 nAl, i32 nAx, u32 parMask [, string s1=0] [, string s2=0] [, string s3=0])</td></tr>
<tr><td>nAl</td><td>Numero dell'allarme (0 o 1-:-9999).</td></tr>
<tr><td>nAx</td><td>Numero dell'asse in allarme (0 o 1-:-32).</td></tr>
<tr><td>parMask</td><td>Maschera per indicare quali parametri sono interessati:
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td>0</td><td>0x00000001</td><td>Nessun parametro USV</td></tr>
<tr><td>1</td><td>0x00000002</td><td>s1</td></tr>
<tr><td>2</td><td>0x00000004</td><td>s2</td></tr>
<tr><td>3</td><td>0x00000008</td><td>s1 e s2</td></tr>
<tr><td>4</td><td>0x00000010</td><td>s3</td></tr>
<tr><td>5</td><td>0x00000020</td><td>s1 e s3</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>bool alarm_reset_par_us (i32 nAl, i32 nAx, u32 parMask [, string s1=0] [, string s2=0] [, string s3=0])</td></tr>
<tr><td></td><td><table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td>6</td><td>0x00000040</td><td>s2 e s3</td></tr>
<tr><td>7</td><td>0x00000080</td><td>s1, s2 e s3</td></tr>
</table></td></tr>
<tr><td>s1</td><td>Stringa utente da usare come primo parametro dell'allarme. (opzionale, default 0)</td></tr>
<tr><td>s2</td><td>Stringa utente da usare come secondo parametro dell'allarme. (opzionale, default 0)</td></tr>
<tr><td>s3</td><td>Stringa utente da usare come terzo parametro dell'allarme. (opzionale, default 0)</td></tr>
<tr><td>Risultato</td><td>Rende:<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td>TRUE</td><td>• allarme cancellato
• allarme disabilitato
• asse non presente</td></tr>
<tr><td>FALSE</td><td>• numero allarme illegale
• numero asse illegale o non definito
• numero allarme o asse sono 0 ma parMask diversa da 0
• allarme non ha parametri USV</td></tr>
</table></td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Requisiti di sistema</td><td>Da RTE 34.30.0.</td></tr>
<tr><td>Note</td><td></td></tr>
</table>

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se numero di allarme non è nell'intervallo degli allarmi utente (dove x = numero di task e xxx numero di step)
- Genera 9900 ill. Arg A:2 T:x ST:xxx se eventuale numero di asse non è valido (dove x = numero di task e xxx numero di step)
- Genera 9900 ill. Arg A:3 T:x ST:xxx se il numero dell'allarme e dell'asse sono 0 ma parMask diversa da 0 (dove x = numero di task e xxx numero di step)

© 2025 Robox SpA

- Genera 9900 ill. Arg A:4 T:x ST:xxx se il numero dell'allarme è 0 ma il numero dell'asse e parMask sono diversi da 0 (dove x = numero di task e xxx numero di step)
- Genera 9900 ill. Arg A:5 T:x ST:xxx se il numero dell'asse è 0 ma il numero dell'allarme e parMask sono diversi da 0 (dove x = numero di task e xxx numero di step)
- Genera 9987 wrong reset_alarm funct. on alarm:xxx se l'allarme non ha una USV definita (dove xxx numero di allarme)

Esempi

Esempio di utilizzo:

```
string s1
string s2
int result
...
; reset an axis mayor alarm
str_format (s1, "Broken Brake")
result = alarm_reset_par_us(1101, 3, 1, s1)
...
...
; reset an axis mayor alarm
str_format (s1, "Broken Drive")
result = alarm_reset_par_us(1101, 3, 1, s1)
...
...
; reset all warnings about Emergency push botton pressed
str_format (s1, "Emergency Pressed")
result = alarm_reset_par_us(4101, 0, 1, s1)
...
; reset specific warning about Emergency push botton pressed
...
str_format (s2, "near first door")
result = alarm_reset_par_us(4101, 0, 2, 0, s2)
...
str_format (s2, "end of line")
result = alarm_set_us(4101, 0, 2, 0, s2)


; alarm definition in user file
1101 -1 14 0x22001000 "Fault <%s>" 17 0 0
4101 53 24 0x22001000 "Warning <%s:%s>" 17 17 0
```

© 2025 Robox SpA

Funzione alarm_set()

Questa funzione permette di inserire un allarme nello stack allarmi.

NOTA: Per inserire un messaggio nello storico allarmi usare la funzione ah_log 163().

Sintassi	**bool alarm_set (i32 nAl [, i32 nAx=0] [, real par1=0] [, real par2=0] [, real par3=0])**
nAl	Numero allarme (1000-:-1999, 3000-:-3999, 4000-:-8999).
nAx	Numero asse in allarme (1÷32). (opzionale, default 0)
par1	Primo parametro da inserire. (opzionale, default 0)
par2	Secondo parametro da inserire. (opzionale, default 0)
par3	Terzo parametro da inserire. (opzionale, default 0)
Risultato	Rende: Valore / Descrizione TRUE / Operazione effettuata correttamente. FALSE / Riscontrati parametri illegali (allarme non inserito).
Validità	Rule, Task.
Note	I tipi dati dei parametri opzionali par1, par2, par3 dipendono comunque dal tipo di allarme interessato. Da RTE 34.24.8 non è più possibile usare questa funzione per settare i warning (9000÷9999).

NOTA: Consultare il file allarmi utente per l'associazione del testo dell'allarme.

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se numero di allarme è in campo riservato al sistema (dove x = numero di task e xxx numero di step)
- Genera 9900 ill. Arg A:2 T:x ST:xxx se eventuale numero di asse non è in range 0-:-32 (dove x = numero di task e xxx numero di step)
- Genera 9998 illegal axes number in alarm <nAll> se l'allarme prevede un numero di asse, ma l'asse non è stato specificato sulla generazione dell'allarme

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
alarm_set(1100, 0, r(1), r(2), r(3))
alarm_set(5000, 1, ip(1), iv(1), ia(1))
alarm_set(4000, ps_1)


; il corrispondente file user_alarm_file.stp potrà essere:
1100 37 0 0x02000100 "il valore di R(1) è:[%d], di R(2) è:[%d] mentre
5000 53 0 0x02000100 "ip =[%f],iv=[%f], ia=[%f]" 8 8 8
1002 53 1 0x02000100 "powerset name= %s"  11
```

Funzione alarm_set_us()

Questa funzione permette di inserire un allarme con variabile stringa utente USV nello stack allarmi e nello storico allarmi.

L'allarme verrà aggiunto nello stack e nello storico quando:

- non è già presente nello stack
- ha un flag di parametro prominente impostato e i parametri sono diversi

Sintassi	**bool alarm_set_us (i32 nAl, i32 nAx, string s1 [, string s2=0] [, string s3=0])**
nAl	Numero allarme (1000-:-1999, 3000-:-3999, 4000-:-8999).
nAx	Numero asse (1-:-32). Mettere 0 se non è un allarme di asse.
s1	Primo parametro da inserire come stringa utente.
s2	Secondo parametro da inserire come stringa utente. (opzionale, default 0)
s3	Terzo parametro da inserire come stringa utente. (opzionale, default 0)
Risultato	Rende: Valore: TRUE — Descrizione: • allarme generato • allarme disabilitato • asse non presente Valore: FALSE — Descrizione: • numero di allarme illegale • numero di asse illegale • allarme senza parametri USV

© 2025 Robox SpA

Sintassi	**bool alarm_set_us (i32 nAl, i32 nAx, string s1 [, string s2=0] [, string s3=0])**
Validità	Rule, Task.
Requisiti di sistema	Da RTE 34.30.0.
Note	

NOTA: Consultare il file allarmi utente per l'associazione del testo dell'allarme.

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se numero di allarme non è nell'intervallo degli allarmi utente (dove x = numero di task e xxx numero di step)
- Genera 9900 ill. Arg A:2 T:x ST:xxx se eventuale numero di asse non è in range 0-:-32 e sys_flag_2.17=0 (dove x = numero di task e xxx numero di step)
- Genera 99 ill. Arg A:2 T:x ST:xxx se eventuale numero di asse non è in range 0-:-32 e sys_flag_2.17=1 (dove x = numero di task e xxx numero di step)
- Genera 9988 wrong set_alarm funct. on alarm:xxx se l'allarme non ha una USV definita (dove xxx numero di allarme)

Esempi

Esempio di utilizzo:

```
string s1
string s2
int result
...
; set an axis mayor alarm
str_format (s1, "Broken Brake")
result = alarm_set_us(1101, 3, s1)
...
...
; set an axis mayor alarm
str_format (s1, "Broken Drive")
result = alarm_set_us(1101, 3, s1)
...
...
; set warning about Emergency push button pressed
str_format (s1, "Emergency Pressed")
...
str_format (s2, "near first door")
result = alarm_set_us(4101, 0, s1, s2)
...
str_format (s2, "end of line")
result = alarm_set_us(4101, 0, s1, s2)

; alarm definition in user file
1101 -1 14 0x22001000 "Fault <%s>" 17 0 0
4101 53 24 0x22001000 "Warning <%s:%s>" 17 17 0
```

© 2025 Robox SpA

Funzione as_get_d()

Questa funzione permette di avere informazioni relative agli allarmi memorizzati nello stack allarmi, con restituzione delle informazioni in una variabile di tipo stru_as_d 826 o stru_as_d_ext 827. Quest'ultima struttura supporta le variabili stringa utente USV.

Sintassi	**bool as_get_d (i32 pos, stru_as_d| stru_as_d_ext stru)**
pos	Posizione nello stack (1-:-32).
stru	Nome della struttura stru_as_d 826 o stru_as_d_ext 827 dove RTE mette i dati di risposta.
Risultato	Rende:

Valore	Descrizione
FALSE	Esito non OK. Genera l'allarme 9900 ill. Arg A:1 se l'indice della posizione nello stack è fuori i limiti.
TRUE	Esito OK. Informazioni sull'allarme rese nella struttura.

Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; dichiarazione variabile di tipo STRU_AS_D
stru_as_d 826 sInfoAll

; ...
; richiesta allarme in posizione 1 dello stack con restituzione delle
as_get_d 178(1, sInfoAll)
```

Funzione asin()

Questa funzione segue l'arcoseno di un valore di tipo reale.

Sintassi	**real asin (real val)**
val	Argomento dell'arcoseno (compreso tra -1 e 1).

© 2025 Robox SpA

Sintassi	**real asin (real val)**
Risultato	Rende l'arcoseno dell'argomento *val*. Il risultato è reso in radianti ed è compreso tra -K_PIMEZZI 17 e K_PIMEZZI 17.
Validità	Rule, Task.
Note	

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se argomento < -1 (dove x = numero di task e xxx numero di step)
- Genera 9900 ill. Arg A:2 T:x ST:xxx se argomento > 1 (dove x = numero di task e xxx numero di step)

Esempi

Esempio di utilizzo:

```
; in rr(10) viene messo l'arcoseno del valore di rr(5) convertito da r
rr(10) = todeg 646 (asin 178 (rr(5)))
; in rr(10) viene messo l'arcoseno di 0.5
rr(10) = asin(0.5)
```

Funzione atan()

Esegue l'arcotangente di un valore di tipo reale.

Sintassi	**real atan (real val)**
val	Argomento dell'arcotangente.
Risultato	Rende l'arcotangente dell'argomento *val*. Il risultato è reso in radianti ed è compreso fra -K_PIMEZZI 17 e K_PIMEZZI 17.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
rr(10) = atan 179 (rr(5))
; se rr(5) -> infinito   rr(10) -> K_PIMEZZI 17

rr(10) = atan(1)
; rr(10) = 0.7853 (K_PI 17/4)
```

© 2025 Robox SpA

```
rr(10) = atan(0)
; rr(10) = 0
```

Funzione atan2()

Esegue l'arcotangente del rapporto tra arg1 e arg2 (arg1/arg2).

Sintassi	**real atan2 (real arg1, real arg2)**
arg1	Primo argomento dell'arcotangente, ovvero valore del seno (o numeratore).
arg2	Secondo argomento dell'arcotangente, ovvero valore del coseno (o denominatore).
Risultato	Contrariamente alla atan179(), la funzione atan2 calcola un risultato compreso tra -K_PI17 e K_PI17 a seconda dei segni dei suoi due argomenti.
Validità	Rule, Task.
Note	

Nel caso particolare in cui arg2 sia uguale a 0 il risultato sarà:

- +K_PIMEZZI17 se arg1 > 0
- - K_PIMEZZI17 se arg1 < 0
- 0 se arg1 = 0

Esempi

Esempio di utilizzo:

```
pippo = atan2180(1,1)
; pippo = 0,7853 (K_PI17/4)

pippo = atan2(0,1)
; pippo = 0

pippo = atan2(1,0)
; pippo = K_PIMEZZI17
```

Esempio di utilizzo:

```
alfaRad = atan2180(sinAlfa, cosAlfa)
betaDeg = todeg646(atan2(120, 43))
```

© 2025 Robox SpA

Funzione axio_iol_r

Questa funzione permette di leggere un'area dati da un IOlink master noto (AXL_SE_IOL4).

Nota: In RDE, ad oggi, non è presente un configuratore IOlink.

Sintassi	**i32 axio_iol_r (i32 rid, i32 slot, i32 buff[], i32 buffSize)**
rid	Robox ID.
slot	Posizione del dispositivo (1-:-63).
buff	Array di i32 dove verranno letti i dati IOlink.
buffSize	Dimensione in numero di byte di buff.
Risultato	Rende: Valore / Descrizione > 0 — Operazione eseguita con successo. -1 — Parametro illegale: *rid.* -2 — Parametro illegale: *slot*. -3 — Parametro illegale: *slot* non IOlink. -4 — Parametro illegale: *buff* == NULL. -5 — Errore in lettura.
Validità	Task.
Note	Per capire i parametri rid e slot associati ai moduli, leggere il capitolo 'Configurazione' del paragrafo comunicazione axioline. Il contenuto e la lunghezza effettiva del buffer dipendono solo dai dispositivi collegati al master IOlink. Anche il significato e il formato dei singoli dati dipendono solo dai dispositivi collegati.

Esempi

Esempio di utilizzo:

```
; assume an AXL_SE_IOL configured as
; ch1 - digital input - it doesn't use any informtion in data area
; ch2 - digital output- it doesn't use any informtion in data area
; ch3 - encoder (velocity - position)- it use 8 bytes in data area 4 b
; ch4 - analog output - it use 2 bytes in data area (in big endian)
INT esitoiol
```

© 2025 Robox SpA

```
INT rid, slot ; rid and slot of device AXl_SE_IOL4
U8 bufiol[64] ; buffer used to store iolink information
INT encVel, encPos ; information
...
...
esitoiol = AXIO_IOL_R (ridiol, slotiol, bufiol, sizeof(bufiol))
IF (esitoiol < 0)
; write here your diagnostic
ENDIF
; get now the information from the buffer
encVel = GET_I32(bufiol, 0); get velocity information from buffer in B
encVel = DWBE2LE(encVel)   ; convert information from Big Endian to Li
encPos = GET_I32(bufiol, 4); get position information from buffer in B
encPos = DWBE2LE(encPos)   ; convert information from Big Endian to Li
```

Funzione axio_iol_w

Questa funzione permette di scrivere un'area dati su un IOlink master noto (AXL_SE_IOL4).

Nota: In RDE, ad oggi, non è presente un configuratore IOlink.

Sintassi	**i32 axio_iol_w (i32 rid, i32 slot, i32 buff[], i32 buffSize)**
rid	Robox ID.
slot	Posizione del dispositivo (1-:-63).
buff	Array di i32 dove verranno scritti i dati IOlink.
buffSize	Dimensione in numero di byte di buff.
Risultato	Rende:

Valore	Descrizione
> 0	Operazione eseguita con successo.
-1	Parametro illegale: *rid.*
-2	Parametro illegale: *slot*.
-3	Parametro illegale: *slot* non IOlink.
-4	Parametro illegale: *buff* == NULL.
-5	Errore in scrittura.

Validità	Task.

© 2025 Robox SpA

Sintassi	**i32 axio_iol_w (i32 rid, i32 slot, i32 buff[], i32 buffSize)**
Note	Per capire i parametri rid e slot associati ai moduli, leggere il capitolo 'Configurazione' del paragrafo comunicazione axioline. Il contenuto e la lunghezza effettiva del buffer dipendono solo dai dispositivi collegati al master IOlink. Anche il significato e il formato dei singoli dati dipendono solo dai dispositivi collegati.

Esempi

Esempio di utilizzo:

```
; assume an AXL_SE_IOL configured as
; ch1 - digital input - it doesn't use any informtion in data area
; ch2 - digital output- it doesn't use any informtion in data area
; ch3 - encoder (velocity - position)- it use 8 bytes in data area 4 b
; ch4 - not used - it doesn't use any informtion in data area
INT esitoiol
INT rid, slot     ; rid and slot of device AXl_SE_IOL4
U8 bufiolOut[64] ; buffer used to write to iolink
INT refOut        ; information to be written in the analogic output
INT refoutD       ; value reduced in the I16 value (+32767 -:- -32768)
I16 refoutBe
...
...
refOutD = IN_RANGE(refOut, -32758, 32767) ; reduce the value in 16 bit
refoutBe = refOutD ; store the informatoin in the correct datatype var
refoutBe = WLE2BE (refoutBe) ; convert from Little Endian to Big Endia
SET_I16 (bufiolOut, 8, refOutBe) ; write to outpute data buffer
; write to IOLINK data output buffer
esitoiol = AXIO_IOL_W (ridiol, slotiol, bufiolOut, SIZEOF(bufiolOut))
IF (esitoiol < 0)
; write here your diagnostic
ENDIF
```

Funzione axio_pm_r()

Questa funzione permette di leggere i dati acquisiti da un dispositivo AXL_F_PM_EF_1F e salvarli in un'apposita struttura di tipo stru_pm_pd 904 oppure in un array di i32.

La lettura delle informazioni di processo avviene se il modulo è connesso direttamente al controllo o se connesso alla testa EtherCAT e poi definito come device ECAT_PM.

NOTA: Questa istruzione può essere utilizzata solo se in rte.cfg è inserita la keyword ECAT_PM.

© 2025 Robox SpA

Sintassi	**i32 axio_pm_r (i32 rid, i32 slot, stru_pm_pd stru, i32 buffSize)** **i32 axio_pm_r (i32 rid, i32 slot, i32 buff[], i32 buffSize)**
rid	Robox ID.
slot	Posizione del dispositivo (1-:-63).
stru *buff*	Struttura di tipo stru_pm_pd 904 oppure array di i32 dove verranno scritti i dati letti dal dispositivo.
buffSize	Dimensione in numero di byte di buff, deve essere un multiplo di 4.
Risultato	Rende:

Valore	Descrizione
> 0	Operazione eseguita con successo.
-1	Parametro illegale: *rid.*
-2	Parametro illegale: *slot*.
-3	Parametro illegale: *slot* non PM.
-4	Parametro illegale: *buff* == NULL.
-5	Parametro illegale: *buffSize* non è un multiplo di 4.
-6	Errore in lettura.

Validità	Task.
Note	Per capire i parametri rid e slot associati ai moduli, leggere il capitolo 'Configurazione' del paragrafo comunicazione axioline.

I dati letti da questa funzione sono i seguenti (con impostazioni di default):

Word	0, 1	2, 3	4, 5	6, 7	8, 9	10, 11	12, 13	14, 15	16, 17	18, 19	20, 21	22, 23	24, 25	26, 27	28, 29	30, 31
Phase	L1	L2	L3	L1	L2	L3	N	L1	L2	L3	L1	L2	L3	L1	L2	L3
Meaning	Voltage			Current				Active Power active energy			Reactive power reactive energy			Apparent power apparent energy		

© 2025 Robox SpA

Word	0, 1	2, 3	4, 5	6, 7	8, 9	10, 11	12, 13	14, 15	16, 17	18, 19	20, 21	22, 23	24, 25	26, 27	28, 29	30, 31
Signal	L1_Voltage L2_Voltage L3_Voltage			L1_Current L2_Current L3_Current NeutralConductor_Current				L1_Real_PE L2_Real_PE L3_Real_PE			L1_Reactive_PE L2_Reactive_PE L3_Reactive_PE			L1_Apparent_PE L2_Apparent_PE L3_Apparent_PE		
Unit	V			A				kW kWh			kVAr kVArh			kVA kVAh		
Resolution	1 mV			1 mA				1 W 1 Wh			1 VAr 1 VArh			1 VA 1 VAh		
Object settings (Default = bold)	Depending on object 0E11☐: 00☐: RMS value (RMS) 01☐: instantaneous value							Depending on object 0E12☐: 00☐: power 01☐: energy								
	Depending on object 0E13☐: 00☐:phase voltage 01☐: phase-to-phase voltage															

Esempi

Esempio di utilizzo:

```
int pm_result
; rid e slot del dispositivo AXl_SE_IOL4
int rid, slot
; buffer usato per salvare i dati
stru_pm_pd 904 pm

; leggo le misure dal modulo
pm_result = axio_pm_r 183(rid, slot, pm, sizeof(pm))
if(pm_result < 0)
    ; scrivi qui la tua diagnostica
endif

; controllo se su ogni linea ci sono più di 100V
if((pm.L1_voltage > 100000) && (pm.L2_voltage > 100000) && (pm.L2_volt
    ; ...
endif
```

© 2025 Robox SpA

Funzione axio_robj()

Questa funzione permette di eseguire la lettura di un oggetto da un modulo axioline specificato.

Funzionamento

Questa funzione permette di eseguire la lettura dell'oggetto '*stru*.index + *stru*.subIndex' sul modulo axioline identificato da '*stru*.rid + *stru*.slot'.

Le informazioni ricevute vengono memorizzate in *destination*, la quale deve essere una variabile grande a sufficienza per contenere i dati che si leggeranno. La dimensione del buffer di destinazione deve essere specificata in *stru*.dataLen.

Impostare *stru*.dataType = 0 per utilizzare il tipo di dato associato all'oggetto (vedi tabella sotto dei codici tipi di dato). In questo caso l'istruzione scriverà in *stru*.dataType il codice del tipo di dato relativo all'oggetto appena letto. Volendo forzare un tipo di dato specifico, impostarne il valore in *stru*.dataType. Di norma, in caso di array è possibile leggere o tutto l'array (specificando *stru*.subIndex = 0) o il singolo elemento. A lettura avvenuta l'istruzione scrive in *stru*.dataType il codice del tipo di dato utilizzato e in *stru*.nRic il numero di byte processati.

La lettura può avvenire in modo passante o in modo bloccante (utilizzabile solo nei task 1-:-8).

Per modo passante si intende che la funzione deve essere continuamente chiamata e questa rende lo stato attuale dell'operazione.

Per modo bloccante si intende che la funzione viene invocata e ritorna solo quando ha completato (con esito positivo o con esito negativo) l'operazione di lettura.

La funzione rende un esito sullo stato dell'esecuzione.

Sintassi	**i32 axio_robj (stru_axio_pdi stru, destination)**
stru	Nome della struttura di tipo stru_axio_pdi 828.
destination	Contiene le informazioni ricevute. Deve essere un oggetto compatibile con i dati che si stanno leggendo (se si legge una VISIBLE_STRING, *destination* sarà di tipo string; se si legge un OCTET_STRING, sarà un array di byte o una struttura).
Risultato	Rende un esito sullo stato dell'esecuzione. Vedi gestione passante e/o gestione bloccante.
Validità	Task.
Note	

Gestione passante

- Imporre ***stru*.phase = 1** per inizializzare la lettura.
- Il comando di lettura viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.
  La funzione rende 0.

© 2025 Robox SpA

- Attesa della risposta.
  La funzione continua a rendere 0.
- Ricezione risposta con esito **positivo**. Il valore letto viene scritto in *destination*. RTE cambia *stru*.phase = 11.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -2, gestore comunicazione occupato
  - -5, comando a un dispositivo non presente o non di tipo axioline ('rid')
  - -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata
  - -7, informazioni dispositivo non ancora disponibili
  - -8, errore interno
  - -10, valore illegale
  - -11, valore illegale di slot
  - -12, valore illegale di objCode
  - -13, valore illegale di subIndex
  - -14, valore illegale di dataType
  - -15, valore illegale di dataLen
  - -16, valore illegale di phase (allarme 9900 arg 6)
  - -17, errore in comando di lettura (tunnel)
  - -22, destinazione cambiata (gestione a fasi)
  - 2, comando di cui si è già ricevuto risposta positiva, valore disponibile in *destination*
  - >2, codice di errore SDO, vedi tabella sotto

Gestione bloccante

- Imporre ***stru*.phase = -1** per eseguire la lettura.
- Il comando di lettura viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.
- Attesa della risposta. Se la risposta non arriva entro un massimo di 3 secondi la funzione esce con un errore (timeout).
- Ricezione risposta con esito **positivo**. Il valore letto viene scritto in *destination*. RTE cambia *stru*.phase = 11.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -2, gestore comunicazione occupato
  - -4, gestione bloccante non permessa nel task corrente (*stru*.phase = -1) (rule, ladder). Comando non accettato (allarme 9900 arg 0)
  - -5, comando a un dispositivo non presente o non di tipo axioline ('rid')
  - -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata

© 2025 Robox SpA

- -7, informazioni dispositivo non ancora disponibili
- -8, errore interno
- -10, valore illegale
- -11, valore illegale di slot
- -12, valore illegale di objcode
- -13, valore illegale di subIndex
- -14, valore illegale di dataType
- -15, valore illegale di dataLen
- -16, valore illegale di phase
- -22, destinazione cambiata (gestione a fasi)
- 2, comando di cui si è già ricevuto risposta positiva, valore disponibile in *destination*
- >2, codice di errore SDO, vedi tabella sotto

Codici di errore SDO

SDO error code	Descrizione
0x0503 0000	Toggle bit non alternato.
0x0504 0000	Timeout del protocollo SDO.
0x0504 0001	Identificatore del comando client/server non valido o sconosciuto.
0x0504 0002	Dimensione del blocco non valida (solo gestione bloccante).
0x0504 0003	Numero di sequenza non valido (solo gestione bloccante).
0x0504 0004	Errore CRC (solo gestione bloccante).
0x0504 0005	Eccesso di memoria.
0x0601 0000	Accesso non supportato a un oggetto.
0x0601 0001	Tentativo di leggere un oggetto a sola scrittura.
0x0601 0002	Tentativo di scrivere un oggetto a sola lettura.
0x0602 0000	L'oggetto non esiste nel dizionario oggetti.
0x0604 0041	Impossibile mappare l'oggetto sul PDO.
0x0604 0042	Il numero e la lunghezza degli oggetti da mappare supererebbero la lunghezza del PDO.
0x0604 0043	Incompatibilità dei parametri.

© 2025 Robox SpA

SDO error code	Descrizione
0x0604 0047	Incompatibilità interna al dispositivo.
0x0606 0000	Accesso non riuscito a causa di un errore hardware.
0x0607 0010	Il tipo di dati non corrisponde, la lunghezza del parametro di servizio non corrisponde.
0x0607 0012	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo alto.
0x0607 0013	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo basso.
0x0609 0011	Il sottoindice non esiste.
0x0609 0030	Superamento del campo di valori del parametro (solo per accesso in scrittura).
0x0609 0031	Valore del parametro scritto troppo alto.
0x0609 0032	Valore del parametro scritto troppo basso.
0x0609 0036	Il valore massimo è inferiore al valore minimo.
0x0800 0000	Errore generale.
0x0800 0020	I dati non possono essere trasferiti o archiviati nell'applicazione.
0x0800 0021	I dati non possono essere trasferiti o archiviati nell'applicazione a causa del controllo locale.
0x0800 0022	I dati non possono essere trasferiti o archiviati nell'applicazione a causa dello stato attuale del dispositivo.
0x0800 0023	La generazione dinamica del dizionario oggetti non riesce o non è presente alcun dizionario oggetti (ad esempio, il dizionario oggetti viene generato da un file e la generazione non riesce a causa di un errore di file).

Codici tipi di dato (dataType)

Codice	Tipo di dato
0	Selezione automatica (default).
2	INTEGER8
3	INTEGER16

© 2025 Robox SpA

Codice	Tipo di dato
4	INTEGER32
5	UNSIGNED8
6	UNSIGNED16
7	UNSIGNED32
9	VISIBLE_STRING
10	OCTET_STRING
21	INTEGER64
27	UNSIGNED64
-2	ARRAY INTEGER8
-3	ARRAY INTEGER16
-4	ARRAY INTEGER32
-5	ARRAY UNSIGNED8
-6	ARRAY UNSIGNED16
-7	ARRAY UNSIGNED32
-10	OCTET_STRING (usato per strutture speciali, ad es. tabella parametri CNT2/INC2)

Esempi

Esempio di utilizzo:

```
; for AXIO_ROBJ / AXIO_WOBJ

int opAXIOobj                ; operation selection
                             ; 0  noone
                             ; 1 blocking read operation
                             ; -1 continuos blocking read operation
                             ; 2 pass-through read operation
                             ; -2  continuos pass-through read operatio
                             ; NOTE the read value will be store in app
                             ;      storage depending by pdi.dataType s
                             ; 10 blocking write operation
                             ; -10 continuos blocking write operation
                             ; 11 pass-through write operation
                             ; -11  continuos pass-through write operat

; desired parameter
int aRid                     ; RID of device where
```

© 2025 Robox SpA

```
int aSlot                       ; slot number of the module (from 1 up 63)
int aIndex                      ; PDI object index number
int aSubi                       ; PDI object subindex number
int aDataType = 0               ; dataType if 0  default data type are ass
int aDataLen = 0                ; in read if 0 di aBufIn sized is used in
int aWhatWrite                  ; specify what to write and where get the
int aNElem                      ; number of array elements (if any)

; struct used to execute the instruction
stru_axio_pdi 828 pdi
u8  aBufIn[256]                 ; general destination variable
real dtRAXIO                    ; used to compute the instruction duration
real tActAxio                   ; instruction duration

; result of the operation
int resultAxioRObj
int resultAxioWObj
string a__srDatatype            ; managed dataType description

; source or destination variable
i8 a_I8                         ; used for INTEGER8
u8 a_U8                         ; used for UNSIGNED8
i16 a_I16                       ; used for INTEGER16
u16 a_U16                       ; used for UNSIGNED16
int a_I32                       ; used for INTEGER32
u32 a_U32                       ; used for UNSIGNED32
i16 a_aI16[256/2]               ; used for -INTEGER16 (ARRAY of INTEGER16)
u16 a_aU16[256/2]               ; used for -UNSIGNED16 (ARRAY of UNSIGNED1
int a_aI32[256/4]               ; used for -INTEGER32 (ARRAY of INTEGER32)
u32 a_aU32[256/4]               ; used for -UNSIGNED32 (ARRAY of UNSIGNED3
string a_VS                     ; used for VISIBLE_STRING
u8 a_Octet[256]                 ; used for OCTET_STRING

struct_p 918 MVF
    float data
    u8    sts
    u8    unit
end_struct_p

MVF a_mvf                       ; used for -OCTET_STRING - object 0x82 sub
MVF a_amvf[8]                   ; used for -OCTET_STRING - object 0x82 sub

struct_p CNT_INC_PARAMTABLE
    u16 basicConfig;
    u16 Resereved1;
    u16 outputConfig;
    u16 outputPulseTime;
    u32 startValue_convFact;
    u32 lowerLimit_moduloVal;
```

© 2025 Robox SpA

```
    u32 upperLimit_encIncVal;
    u32 reserved2_limitValOut1;
    u32 reserved3_limitValOut2;
    u32 reserved4_referenceVal;
    u32 reserved5_offsetZref;
    u32 reserved6_trasmFact;
    u32 reserved7_reserved8;
end_struct_p

CNT_INC_PARAMTABLE a_paraCntInc ; used for -OCTET_STRING - object 0x80

;<0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0>
;-----------------------------------------------------------------
function TestAXIOxOBJ()
;-----------------------------------------------------------------
;<0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0>

; function to test AXIO_ROBJE E AXIO_WOBJ instructions
select(opAXIOobj)
    case -100    ; error phase
        break

    case 0       ; nothing to do
        break

    case -1      ;opAxioObj = 1  continuous blocking Read on AXIO devic
    case 1       ;opAxioObj = 1  blocking Read on AXIO device
        pdi.rid      = aRid
        pdi.slot     = aSlot
        pdi.index    = aIndex
        pdi.subindex= aSubi
        pdi.dataType= aDataType
        pdi.dataLen = cond_val289(aDataLen = 0, sizeof602(aBufIn), aDat
        pdi.phase    = -1
        tActAxio     = time_micro642()

        resultAxioRObj = axio_robj186(pdi, abufin)
        dtRAXIO = time_micro() - tActAxio
        if(resultAxioRObj < 0)
            opAXIOobj = -100
            return
        endif

        ; prendiamo esito in funzione del datatype

        select(pdi.dataType)
            default    ; tipo non noto
                a__srDatatype = "DataType : non noto"
                break
            case 2     ; integer 8
                a__srDatatype = "DataType : integer 8"
```

© 2025 Robox SpA

```
        a_I8 = abufin[0]
        break
    case 5    ; unsigned 8
        a__srDatatype = "DataType : unsigned 8"
        a_U8 = abufin[0]
        break
    case 3    ; integer 16
        a__srDatatype = "DataType : integer 16"
        a_I16 = sw0(639)(abufin[0])
        break
    case -3   ; array integer 16
        a__srDatatype = "DataType : array integer 16"
        move_byte(731)(abufin, a_aI16, pdi.nric)
        break
    case 6    ; unsigned 16
        a__srDatatype = "DataType : unsigned 16"
        a_U16 = w0(671)(abufin[0])
        break
    case -6   ; array unsigned 16
        a__srDatatype = "DataType : array unsigned 16"
        move_byte(abufin, a_aU16, pdi.nric)
        break
    case 4    ; integer 32
        a__srDatatype = "DataType : integer 32"
        a_I32 = sdw0(583)(abufin[0])
        break
    case -4   ; array integer 32
        a__srDatatype = "DataType : array integer 32"
        move_byte(abufin, a_aI32, pdi.nric)
        break
    case 7    ; unsigned 32
        a__srDatatype = "DataType : unsigned 32"
        a_U32 = dw0(300)(abufin[0])
        break
    case -7   ; arrayunsigned 32
        a__srDatatype = "DataType : array unsigned 32"
        move_byte(abufin, a_aU32, pdi.nric)
        break
    case 9    ; visible string
        a__srDatatype = "DataType : visible string"
        move_byte(abufin, a_VS, pdi.nric)
        break
    case 10   ; Octet string
        a__srDatatype = "DataType : octet string"
        move_byte(abufin, a_Octet, pdi.nric)
        break
    case -10  ; -Octet string
        select(pdi.index)
            default
                a__srDatatype = "DataType : -Octet string - un
                move_byte(abufin, a_Octet, pdi.nric)
```

© 2025 Robox SpA

```
                    break
                case 0x80
                    a__srDatatype = "DataType : -Octet string - CN
                    move_byte(abufin, a_paraCntInc, pdi.nric)
                    break
                case 0x82
                    if(pdi.nRic > (int)sizeof(MVF))
                        a__srDatatype = "DataType : -Octet string
                        move_byte(abufin, a_aMVF[0], pdi.nric)
                    else
                        a__srDatatype = "DataType : -Octet string
                        move_byte(abufin, a_MVF, pdi.nric)
                    endif
            end_select
            break
        case 27  ; U64
            a__srDatatype = "DataType : unsigned 64 as octet"
            move_byte(abufin, a_Octet, pdi.nric)
            break
    end_select

    _if(opAXIOobj> 0)
        opAXIOobj = 0;
    break
case -2     ;opAxioObj = 1  continuous pass-through Read on AXIO d
case 2      ;opAxioObj = 1  pass-through Read on AXIO device
    pdi.rid     = aRid
    pdi.slot    = aSlot
    pdi.index   = aIndex
    pdi.subindex= aSubi
    pdi.dataType= aDataType
    pdi.dataLen = cond_val(aDataLen = 0, sizeof(aBufIn), aDataLen)
    pdi.phase   = 1
    tActAxio    = time_micro();
case -3
case 3
    resultAxioRObj = axio_robj(pdi, abufin);
    if(resultAxioRObj = 0)
        opAXIOobj = cond_val(opAXIOobj < 0, -3, 3)
        break
    endif
    dtRAXIO = time_micro() - tActAxio;
    if(resultAxioRObj < 0)
        opAXIOobj = -100;
        return
    endif

    ; prendiamo esito in funzione del datatype

    select(pdi.dataType)
        default   ; tipo non noto
            a__srDatatype = "DataType : non noto"
```

© 2025 Robox SpA

```
        break
    case 2    ; integer 8
        a__srDatatype = "DataType : integer 8"
        a_I8 = abufin[0]
        break
    case 5    ; unsigned 8
        a__srDatatype = "DataType : unsigned 8"
        a_U8 = abufin[0]
        break
    case 3    ; integer 16
        a__srDatatype = "DataType : integer 16"
        a_I16 = sw0(abufin[0])
        break
    case -3   ; array integer 16
        a__srDatatype = "DataType : array integer 16"
        move_byte(abufin, a_aI16, pdi.nric)
        break
    case 6    ; unsigned 16
        a__srDatatype = "DataType : unsigned 16"
        a_U16 = w0(abufin[0])
        break
    case -6   ; array unsigned 16
        a__srDatatype = "DataType : array unsigned 16"
        move_byte(abufin, a_aU16, pdi.nric)
        break
    case 4    ; integer 32
        a__srDatatype = "DataType : integer 32"
        a_I32 = sdw0(abufin[0])
        break
    case -4   ; array integer 32
        a__srDatatype = "DataType : array integer 32"
        move_byte(abufin, a_aI32, pdi.nric)
        break
    case 7    ; unsigned 32
        a__srDatatype = "DataType : unsigned 32"
        a_U32 = dw0(abufin[0])
        break
    case -7   ; arrayunsigned 32
        a__srDatatype = "DataType : array unsigned 32"
        move_byte(abufin, a_aU32, pdi.nric)
        break
    case 9    ; visible string
        a__srDatatype = "DataType : visible string"
        move_byte(abufin, a_VS, pdi.nric)
        break
    case 10   ; Octet string
        a__srDatatype = "DataType : octet string"
        move_byte(abufin, a_Octet, pdi.nric)
        break
    case -10  ; -Octet string
        select(pdi.index)
```

© 2025 Robox SpA

```
                    default
                        a__srDatatype = "DataType : -Octet string - un
                        move_byte(abufin, a_Octet, pdi.nric)
                        break
                    case 0x80
                        a__srDatatype = "DataType : -Octet string - CN
                        move_byte(abufin, a_paraCntInc, pdi.nric)
                        break
                    case 0x82
                        if(pdi.nRic > (int)sizeof(MVF))
                            a__srDatatype = "DataType : -Octet string
                            move_byte(abufin, a_aMVF[0], pdi.nric)
                        else
                            a__srDatatype = "DataType : -Octet string
                            move_byte(abufin, a_MVF, pdi.nric)
                        endif
                end_select
                break
            case 27  ; U64
                a__srDatatype = "DataType : unsigned 64 as octet"
                move_byte(abufin, a_Octet, pdi.nric)
                break
        end_select

        if(opAXIOobj> 0)
            opAXIOobj = 0;
        else
            opAXIOobj = -2;
        endif
        break
    case -10         ;opAxioObj = -10  continuous blocking Write on AXI
    case 10          ;opAxioObj = 10   blocking Write on AXIO device
        pdi.rid      = aRid
        pdi.slot     = aSlot
        pdi.index    = aIndex
        pdi.subindex= aSubi
        pdi.dataType= aDataType
        pdi.phase    = -1
        tActAxio     = time_micro();
        select(aWhatWrite)
            case 9     ; variable string
                a__srDatatype = "DataType : visible string"
                pdi.dataLen   = str_len(a_VS) +1
                resultAxioRObj   = axio_wobj(pdi, a_vs);
                break
            case 6     ; unsigned 16
                a__srDatatype = "DataType : unsigned 16"
                pdi.dataLen   = sizeof(a_U16)
                resultAxioRObj   = axio_wobj(pdi, a_u16);
                break
            case -6    ; array unsigned 16
                a__srDatatype = "DataType : array unsigned 16"
```

© 2025 Robox SpA

```
            pdi.dataLen  = sizeof(a_U16) * aNElem
            resultAxioRObj  = axio_wobj(pdi, a_au16);
            break
        case 3    ; integer 16
            a__srDatatype = "DataType : integer 16"
            pdi.dataLen  = sizeof(a_I16)
            resultAxioRObj  = axio_wobj(pdi, a_i16);
            break
        case -3   ; array integer 16
            a__srDatatype = "DataType : array integer 16"
            pdi.dataLen  = sizeof(a_I16) * aNElem
            resultAxioRObj  = axio_wobj(pdi, a_ai16);
            break
        case 5    ; unsigned 8
            a__srDatatype = "DataType : unsigned 8"
            pdi.dataLen  = sizeof(a_U8)
            resultAxioRObj  = axio_wobj(pdi, a_u8);
            break
        case 2    ; integer 8
            a__srDatatype = "DataType : integer 8"
            pdi.dataLen  = sizeof(a_I8)
            resultAxioRObj  = axio_wobj(pdi, a_i8);
            break
        case 10   ; Octet string
            a__srDatatype = "DataType : octet string"
            pdi.dataLen  = aNElem
            resultAxioRObj  = axio_wobj(pdi, a_octet);
            break

        case -10  ; -Octet string
            select(pdi.index)
                default
                    a__srDatatype = "DataType : -Octet string - un
                    pdi.dataLen  = aNElem
                    resultAxioRObj  = axio_wobj(pdi, a_octet);
                    break
                case 0x80
                    a__srDatatype = "DataType : -Octet string - CN
                    pdi.dataLen  = sizeof(a_paraCntInc)
                    resultAxioRObj  = axio_wobj(pdi, a_paracntinc)
                    break
            end_select
            break

    end_select

    dtRAXIO = time_micro() - tActAxio;
    if((resultAxioWObj < 0) or (resultAxioWObj > 2))
        opAXIOobj = -100;
        return
    endif
    _if (opAXIOobj> 0)
```

© 2025 Robox SpA

```
            opAXIOobj = 0;
        break    ; CASE 10 / -10


    case -11     ;opAxioObj = -11  continuous pass-through Write on AXI
    case 11      ;opAxioObj = 11   pass-through Write on AXIO device
        pdi.rid      = aRid
        pdi.slot     = aSlot
        pdi.index    = aIndex
        pdi.subindex = aSubi
        pdi.dataType = aDataType
        pdi.phase    = 1
        tActAxio     = time_micro()
    case -12
    case 12
        select(aWhatWrite)
            case 9     ; variable string
                a__srDatatype = "DataType : visible string"
                pdi.dataLen   = str_len(a_VS) +1
                resultAxioRObj   = axio_wobj(pdi, a_vs);
                break
            case 6     ; unsigned 16
                a__srDatatype = "DataType : unsigned 16"
                pdi.dataLen   = sizeof(a_U16)
                resultAxioRObj   = axio_wobj(pdi, a_u16);
                break
            case -6    ; array unsigned 16
                a__srDatatype = "DataType : array unsigned 16"
                pdi.dataLen   = sizeof(a_U16) * aNElem
                resultAxioRObj   = axio_wobj(pdi, a_au16);
                break
            case 3     ; integer 16
                a__srDatatype = "DataType : integer 16"
                pdi.dataLen   = sizeof(a_I16)
                resultAxioRObj   = axio_wobj(pdi, a_i16);
                break
            case -3    ; array integer 16
                a__srDatatype = "DataType : array integer 16"
                pdi.dataLen   = sizeof(a_I16) * aNElem
                resultAxioRObj   = axio_wobj(pdi, a_ai16);
                break
            case 5     ; unsigned 8
                a__srDatatype = "DataType : unsigned 8"
                pdi.dataLen   = sizeof(a_U8)
                resultAxioRObj   = axio_wobj(pdi, a_u8);
                break
            case 2     ; integer 8
                a__srDatatype = "DataType : integer 8"
                pdi.dataLen   = sizeof(a_I8)
                resultAxioRObj   = axio_wobj(pdi, a_i8);
                break
```

© 2025 Robox SpA

```
            case 10    ; Octet string
                a__srDatatype = "DataType : octet string"
                pdi.dataLen  = aNElem
                resultAxioRObj  = axio_wobj(pdi, a_octet);
                break

            case -10   ; -Octet string
                select(pdi.index)
                    default
                        a__srDatatype = "DataType : -Octet string - un
                        pdi.dataLen  = aNElem
                        resultAxioRObj  = axio_wobj(pdi, a_octet);
                        break
                    case 0x80
                        a__srDatatype = "DataType : -Octet string - CN
                        pdi.dataLen  = sizeof(a_paraCntInc)
                        resultAxioRObj  = axio_wobj(pdi, a_paracntinc)
                        break
                end_select
                break

        end_select

        if(resultAxioWObj = 0)
            opAXIOobj = cond_val(opAXIOobj < 0, -12, 12)
            break
        endif
        dtRAXIO = time_micro() - tActAxio;
        if((resultAxioWObj < 0) or (resultAxioWObj > 2))
            opAXIOobj = -100;
            return
        endif
        if(opAXIOobj> 0)
            opAXIOobj = 0;
        else
            opAXIOobj =  -11
        endif
        break
    end_select
endfun
```

Funzione axio_wobj()

Questa funzione permette di eseguire la scrittura di un oggetto su un modulo axioline specifico.

Funzionamento

Questa funzione permette di eseguire la scrittura dell'oggetto '*stru*.objCode + *stru*.subIndex' sul modulo axioline identificato da '*stru*.rid + *stru*.slot'.

© 2025 Robox SpA

Vengono scritte le informazioni presenti in *source*. I dati contenuti in *source* verranno assunti in base al tipo di dato (vedi tabella sotto dei codici tipi di dato). Il tipo di dato, se non specificato direttamente (*stru*.dataType <> 0), viene desunto dall'oggetto che vuole scrivere. Il numero di byte da scrivere deve essere impostato nel campo *stru*.dataLen. In caso di tipi di dato singoli (integer8/16/32, unsigned8/16/32) esso verrà utilizzato per controllare se il sorgente ha dimensione sufficiente. In caso di tipi di dato uguale a OCTET_STRING viene assunto come numero di byte da scrivere. In caso di VISIBLE_STRING il dato è ignorato e viene usata direttamente la dimensione della stringa. In caso di array, *stru*.datalen viene usato per calcolare il numero di elementi dell'array.

A scrittura avvenuta l'istruzione impone in *stru*.dataType il codice di tipo dato utilizzato e in *stru*.nRic il numero di byte processati.

Di norma, in caso di array è possibile scrivere o tutto l'array (specificando *stru*.subIndex = 0) o il singolo elemento.

La scrittura può avvenire in modo passante o in modo bloccante (utilizzabile solo nei task 1-:-8). Per modo bloccante si intende che la funzione viene invocata e ritorna solo quando ha completato (con esito positivo o negativo) l'operazione di scrittura.

La funzione rende un esito sullo stato dell'esecuzione.

Sintassi	**i32 axio_wobj (stru_axio_pdi stru, source)**
stru	Nome della struttura di tipo stru_axio_pdi 828.
source	Contiene le informazioni da inviare. *source* deve essere un oggetto compatibile con i dati che si stanno scrivendo (se si scrive una VISIBLE STRING, *source* sarà di tipo string; se si scrive un OCTET_STRING sarà un array di byte o una struttura).
Risultato	Rende un esito sullo stato dell'esecuzione. Vedi gestione passante e/o gestione bloccante.
Validità	Task.
Note	

Gestione passante

- Imporre ***stru*.phase = 1** al fine di inizializzare l'operazione di scrittura.
- Il comando di scrittura del valore *source* viene inviato al dispositivo remoto e viene attesa la risposta. La variabile *stru*.phase viene cambiata.
  La funzione rende 0.
- Attesa della risposta.
  La funzione continua a rendere 0.
- Ricezione risposta con esito **positivo**. RTE cambia *stru*.phase = 7.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -2, gestore comunicazione occupato
  - -5, comando a un dispositivo non presente o non di tipo axioline ('s.rid')

© 2025 Robox SpA

- -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata
- -7, informazioni dispositivo non ancora disponibili
- -8, errore interno
- -10, valore illegale
- -11, valore illegale di slot
- -12, valore illegale di objCode
- -13, valore illegale di subIndex
- -14, valore illegale di dataType
- -15, valore illegale di dataLen
- -16, valore illegale di phase (alarm 9900 A:6)
- -17, errore in comando di lettura (tunnel)
- -20, il valore scritto è lo stesso
- -23, object a sola lettura
- 2, comando di cui si è già avuto risposta positiva, valore disponibile in *source*
- >2, codice di errore SDO, vedi sotto tabella codici di errore SDO

Gestione bloccante

- Imporre ***stru*.phase = -1** per eseguire la scrittura.
- Il comando di scrittura del valore *source* viene inviato al dispositivo remoto e viene attesa la risposta. La variabile *stru*.phase viene cambiata.
- Attesa della risposta. Se la risposta non arriva entro un massimo di 3 secondi la funzione esce con un errore (timeout).
- Ricezione risposta con esito **positivo**. RTE cambia *stru*.phase = 7.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -2, gestore comunicazione occupato
  - -4, gestione bloccante non permessa net task corrente (*stru*.phase = -1) (rule, ladder). Comando non accettato (alarm 9900 A:0)
  - -5, comando a un dispositivo non presente o non di tipo axioline (*stru*.rid)
  - -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata
  - -7, informazioni dispositivo non ancora disponibili
  - -8, errore interno
  - -10, valore illegale
  - -11, valore illegale di slot
  - -12, valore illegale di objCode
  - -13, valore illegale di subIndex
  - -14, valore illegale di dataType

© 2025 Robox SpA

- -15, valore illegale di dataLen
- -16, valore illegale di phase (alarm 9900 A:6)
- -17, errore in comando di lettura (tunnel)
- -20, il valore scritto è lo stesso
- -23, object a sola lettura
- 2, comando di cui si è già avuta risposta positiva, valore disponibile in *source*
- >2, codice di errore SDO, vedi sotto tabella codici di errore SDO

Codici di errore SDO

SDO error code	Descrizione
0x0503 0000	Toggle bit non alternato.
0x0504 0000	Timeout del protocollo SDO.
0x0504 0001	Identificatore del comando client/server non valido o sconosciuto.
0x0504 0002	Dimensione del blocco non valida (solo gestione bloccante).
0x0504 0003	Numero di sequenza non valido (solo gestione bloccante).
0x0504 0004	Errore CRC (solo gestione bloccante).
0x0504 0005	Eccesso di memoria.
0x0601 0000	Accesso non supportato a un oggetto.
0x0601 0001	Tentativo di leggere un oggetto a sola scrittura.
0x0601 0002	Tentativo di scrivere un oggetto a sola lettura.
0x0602 0000	L'oggetto non esiste nel dizionario oggetti.
0x0604 0041	Impossibile mappare l'oggetto sul PDO.
0x0604 0042	Il numero e la lunghezza degli oggetti da mappare supererebbero la lunghezza del PDO.
0x0604 0043	Incompatibilità dei parametri.
0x0604 0047	Incompatibilità interna al dispositivo.
0x0606 0000	Accesso non riuscito a causa di un errore hardware.
0x0607 0010	Il tipo di dati non corrisponde, la lunghezza del parametro di servizio non corrisponde.

© 2025 Robox SpA

SDO error code	Descrizione
0x0607 0012	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo alto.
0x0607 0013	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo basso.
0x0609 0011	Il sottoindice non esiste.
0x0609 0030	Superamento del campo di valori del parametro (solo per accesso in scrittura).
0x0609 0031	Valore del parametro scritto troppo alto.
0x0609 0032	Valore del parametro scritto troppo basso.
0x0609 0036	Il valore massimo è inferiore al valore minimo.
0x0800 0000	Errore generale.
0x0800 0020	I dati non possono essere trasferiti o archiviati nell'applicazione.
0x0800 0021	I dati non possono essere trasferiti o archiviati nell'applicazione a causa del controllo locale.
0x0800 0022	I dati non possono essere trasferiti o archiviati nell'applicazione a causa dello stato attuale del dispositivo.
0x0800 0023	La generazione dinamica del dizionario oggetti non riesce o non è presente alcun dizionario oggetti (ad esempio, il dizionario oggetti viene generato da un file e la generazione non riesce a causa di un errore di file).

Codici tipi di dato (dataType)

Codice	Tipo di dato
0	Selezione automatica (default).
2	INTEGER8
3	INTEGER16
4	INTEGER32
5	UNSIGNED8
6	UNSIGNED16
7	UNSIGNED32

© 2025 Robox SpA

Codice	Tipo di dato
9	VISIBLE_STRING
10	OCTET_STRING
21	INTEGER64
27	UNSIGNED64
-2	ARRAY INTEGER8
-3	ARRAY INTEGER16
-4	ARRAY INTEGER32
-5	ARRAY UNSIGNED8
-6	ARRAY UNSIGNED16
-7	ARRAY UNSIGNED32
-10	OCTET_STRING (usato per strutture speciali, ad es. tabella parametri CNT2/INC2)

Esempi

Esempio di utilizzo:

```
; for AXIO_ROBJ / AXIO_WOBJ

int opAXIOobj                ; operation selection
                             ; 0  noone
                             ; 1 blocking read operation
                             ; -1 continuos blocking read operation
                             ; 2 pass-through read operation
                             ; -2  continuos pass-through read operatio
                             ; NOTE the read value will be store in app
                             ;      storage depending by pdi.dataType s
                             ; 10 blocking write operation
                             ; -10 continuos blocking write operation
                             ; 11 pass-through write operation
                             ; -11  continuos pass-through write operat

; desired parameter
int aRid                     ; RID of device where
int aSlot                    ; slot number of the module (from 1 up 63)
int aIndex                   ; PDI object index number
int aSubi                    ; PDI object subindex number
int aDataType = 0            ; dataType if 0  default data type are ass
int aDataLen = 0             ; in read if 0 di aBufIn sized is used in
int aWhatWrite               ; specify what to write and where get the
int aNElem                   ; number of array elements (if any)
```

© 2025 Robox SpA

```
; struct used to execute the instruction
stru_axio_pdi 828 pdi
u8  aBufIn[256]              ; general destination variable
real dtRAXIO                 ; used to compute the instruction duration
real tActAxio                ; instruction duration

; result of the operation
int resultAxioRObj
int resultAxioWObj
string a__srDatatype         ; managed dataType description

; source or destination variable
i8 a_I8                      ; used for INTEGER8
u8 a_U8                      ; used for UNSIGNED8
i16 a_I16                    ; used for INTEGER16
u16 a_U16                    ; used for UNSIGNED16
int a_I32                    ; used for INTEGER32
u32 a_U32                    ; used for UNSIGNED32
i16 a_aI16[256/2]            ; used for -INTEGER16 (ARRAY of INTEGER16)
u16 a_aU16[256/2]            ; used for -UNSIGNED16 (ARRAY of UNSIGNED1
int a_aI32[256/4]            ; used for -INTEGER32 (ARRAY of INTEGER32)
u32 a_aU32[256/4]            ; used for -UNSIGNED32 (ARRAY of UNSIGNED3
string a_VS                  ; used for VISIBLE_STRING
u8 a_Octet[256]              ; used for OCTET_STRING

struct_p 918 MVF
    float data
    u8    sts
    u8    unit
end_struct_p

MVF a_mvf                    ; used for -OCTET_STRING - object 0x82 sub
MVF a_amvf[8]                ; used for -OCTET_STRING - object 0x82 sub

struct_p CNT_INC_PARAMTABLE
    u16 basicConfig;
    u16 Resereved1;
    u16 outputConfig;
    u16 outputPulseTime;
    u32 startValue_convFact;
    u32 lowerLimit_moduloVal;
    u32 upperLimit_encIncVal;
    u32 reserved2_limitValOut1;
    u32 reserved3_limitValOut2;
    u32 reserved4_referenceVal;
    u32 reserved5_offsetZref;
    u32 reserved6_trasmFact;
    u32 reserved7_reserved8;
end_struct_p
```

© 2025 Robox SpA

```
CNT_INC_PARAMTABLE a_paraCntInc ; used for -OCTET_STRING - object 0x80

;<0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0>
;-----------------------------------------------------------------
function TestAXIOxOBJ()
;-----------------------------------------------------------------
;<0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0>

; function to test AXIO_ROBJE E AXIO_WOBJ instructions
select(opAXIOobj)
    case -100    ; error phase
        break

    case 0       ; nothing to do
        break

    case -1      ;opAxioObj = 1  continuous blocking Read on AXIO devic
    case 1       ;opAxioObj = 1  blocking Read on AXIO device
        pdi.rid      = aRid
        pdi.slot     = aSlot
        pdi.index    = aIndex
        pdi.subindex= aSubi
        pdi.dataType= aDataType
        pdi.dataLen = cond_val289(aDataLen = 0, sizeof602(aBufIn), aDat
        pdi.phase    = -1
        tActAxio     = time_micro642()

        resultAxioRObj = axio_robj186(pdi, abufin)
        dtRAXIO = time_micro() - tActAxio
        if(resultAxioRObj < 0)
            opAXIOobj = -100
            return
        endif

        ; prendiamo esito in funzione del datatype

        select(pdi.dataType)
            default    ; tipo non noto
                a__srDatatype = "DataType : non noto"
                break
            case 2     ; integer 8
                a__srDatatype = "DataType : integer 8"
                a_I8 = abufin[0]
                break
            case 5     ; unsigned 8
                a__srDatatype = "DataType : unsigned 8"
                a_U8 = abufin[0]
                break
            case 3     ; integer 16
                a__srDatatype = "DataType : integer 16"
```

© 2025 Robox SpA

```
        a_I16 = sw0(abufin[0])
        break
    case -3    ; array integer 16
        a__srDatatype = "DataType : array integer 16"
        move_byte(abufin, a_aI16, pdi.nric)
        break
    case 6     ; unsigned 16
        a__srDatatype = "DataType : unsigned 16"
        a_U16 = w0(abufin[0])
        break
    case -6    ; array unsigned 16
        a__srDatatype = "DataType : array unsigned 16"
        move_byte(abufin, a_aU16, pdi.nric)
        break
    case 4     ; integer 32
        a__srDatatype = "DataType : integer 32"
        a_I32 = sdw0(abufin[0])
        break
    case -4    ; array integer 32
        a__srDatatype = "DataType : array integer 32"
        move_byte(abufin, a_aI32, pdi.nric)
        break
    case 7     ; unsigned 32
        a__srDatatype = "DataType : unsigned 32"
        a_U32 = dw0(abufin[0])
        break
    case -7    ; arrayunsigned 32
        a__srDatatype = "DataType : array unsigned 32"
        move_byte(abufin, a_aU32, pdi.nric)
        break
    case 9     ; visible string
        a__srDatatype = "DataType : visible string"
        move_byte(abufin, a_VS, pdi.nric)
        break
    case 10    ; Octet string
        a__srDatatype = "DataType : octet string"
        move_byte(abufin, a_Octet, pdi.nric)
        break
    case -10   ; -Octet string
        select(pdi.index)
            default
                a__srDatatype = "DataType : -Octet string - un
                move_byte(abufin, a_Octet, pdi.nric)
                break
            case 0x80
                a__srDatatype = "DataType : -Octet string - CN
                move_byte(abufin, a_paraCntInc, pdi.nric)
                break
            case 0x82
                if(pdi.nRic > (int)sizeof(MVF))
                    a__srDatatype = "DataType : -Octet string
```

© 2025 Robox SpA

```
                            move_byte(abufin, a_aMVF[0], pdi.nric)
                        else
                            a__srDatatype = "DataType : -Octet string
                            move_byte(abufin, a_MVF, pdi.nric)
                        endif
                end_select
                break
            case 27   ; U64
                a__srDatatype = "DataType : unsigned 64 as octet"
                move_byte(abufin, a_Octet, pdi.nric)
                break
        end_select

        _if(opAXIOobj> 0)
            opAXIOobj = 0;
        break
    case -2      ;opAxioObj = 1  continuous pass-through Read on AXIO d
    case 2       ;opAxioObj = 1  pass-through Read on AXIO device
        pdi.rid      = aRid
        pdi.slot     = aSlot
        pdi.index    = aIndex
        pdi.subindex= aSubi
        pdi.dataType= aDataType
        pdi.dataLen = cond_val(aDataLen = 0, sizeof(aBufIn), aDataLen)
        pdi.phase    = 1
        tActAxio     = time_micro();
    case -3
    case 3
        resultAxioRObj = axio_robj(pdi, abufin);
        if(resultAxioRObj = 0)
            opAXIOobj = cond_val(opAXIOobj < 0, -3, 3)
            break
        endif
        dtRAXIO = time_micro() - tActAxio;
        if(resultAxioRObj < 0)
            opAXIOobj = -100;
            return
        endif

        ; prendiamo esito in funzione del datatype

        select(pdi.dataType)
            default    ; tipo non noto
                a__srDatatype = "DataType : non noto"
                break
            case 2     ; integer 8
                a__srDatatype = "DataType : integer 8"
                a_I8 = abufin[0]
                break
            case 5     ; unsigned 8
                a__srDatatype = "DataType : unsigned 8"
                a_U8 = abufin[0]
```

© 2025 Robox SpA

```
        break
    case 3    ; integer 16
        a__srDatatype = "DataType : integer 16"
        a_I16 = sw0(abufin[0])
        break
    case -3   ; array integer 16
        a__srDatatype = "DataType : array integer 16"
        move_byte(abufin, a_aI16, pdi.nric)
        break
    case 6    ; unsigned 16
        a__srDatatype = "DataType : unsigned 16"
        a_U16 = w0(abufin[0])
        break
    case -6   ; array unsigned 16
        a__srDatatype = "DataType : array unsigned 16"
        move_byte(abufin, a_aU16, pdi.nric)
        break
    case 4    ; integer 32
        a__srDatatype = "DataType : integer 32"
        a_I32 = sdw0(abufin[0])
        break
    case -4   ; array integer 32
        a__srDatatype = "DataType : array integer 32"
        move_byte(abufin, a_aI32, pdi.nric)
        break
    case 7    ; unsigned 32
        a__srDatatype = "DataType : unsigned 32"
        a_U32 = dw0(abufin[0])
        break
    case -7   ; arrayunsigned 32
        a__srDatatype = "DataType : array unsigned 32"
        move_byte(abufin, a_aU32, pdi.nric)
        break
    case 9    ; visible string
        a__srDatatype = "DataType : visible string"
        move_byte(abufin, a_VS, pdi.nric)
        break
    case 10   ; Octet string
        a__srDatatype = "DataType : octet string"
        move_byte(abufin, a_Octet, pdi.nric)
        break
    case -10  ; -Octet string
        select(pdi.index)
            default
                a__srDatatype = "DataType : -Octet string - un
                move_byte(abufin, a_Octet, pdi.nric)
                break
            case 0x80
                a__srDatatype = "DataType : -Octet string - CN
                move_byte(abufin, a_paraCntInc, pdi.nric)
                break
```

© 2025 Robox SpA

```
                    case 0x82
                        if(pdi.nRic > (int)sizeof(MVF))
                            a__srDatatype = "DataType : -Octet string
                            move_byte(abufin, a_aMVF[0], pdi.nric)
                        else
                            a__srDatatype = "DataType : -Octet string
                            move_byte(abufin, a_MVF, pdi.nric)
                        endif
                end_select
                break
            case 27  ; U64
                a__srDatatype = "DataType : unsigned 64 as octet"
                move_byte(abufin, a_Octet, pdi.nric)
                break
        end_select

        if(opAXIOobj> 0)
            opAXIOobj = 0;
        else
            opAXIOobj = -2;
        endif
        break
    case -10         ;opAxioObj = -10  continuous blocking Write on AXI
    case 10          ;opAxioObj = 10   blocking Write on AXIO device
        pdi.rid      = aRid
        pdi.slot     = aSlot
        pdi.index    = aIndex
        pdi.subindex= aSubi
        pdi.dataType= aDataType
        pdi.phase    = -1
        tActAxio     = time_micro();
        select(aWhatWrite)
            case 9     ; variable string
                a__srDatatype = "DataType : visible string"
                pdi.dataLen   = str_len631(a_VS) +1
                resultAxioRObj   = axio_wobj199(pdi, a_vs);
                break
            case 6     ; unsigned 16
                a__srDatatype = "DataType : unsigned 16"
                pdi.dataLen   = sizeof(a_U16)
                resultAxioRObj   = axio_wobj(pdi, a_u16);
                break
            case -6    ; array unsigned 16
                a__srDatatype = "DataType : array unsigned 16"
                pdi.dataLen   = sizeof(a_U16) * aNElem
                resultAxioRObj   = axio_wobj(pdi, a_au16);
                break
            case 3     ; integer 16
                a__srDatatype = "DataType : integer 16"
                pdi.dataLen   = sizeof(a_I16)
                resultAxioRObj   = axio_wobj(pdi, a_i16);
                break
```

© 2025 Robox SpA

```
            case -3    ; array integer 16
                a__srDatatype = "DataType : array integer 16"
                pdi.dataLen  = sizeof(a_I16) * aNElem
                resultAxioRObj  = axio_wobj(pdi, a_ai16);
                break
            case 5     ; unsigned 8
                a__srDatatype = "DataType : unsigned 8"
                pdi.dataLen  = sizeof(a_U8)
                resultAxioRObj  = axio_wobj(pdi, a_u8);
                break
            case 2     ; integer 8
                a__srDatatype = "DataType : integer 8"
                pdi.dataLen  = sizeof(a_I8)
                resultAxioRObj  = axio_wobj(pdi, a_i8);
                break
            case 10    ; Octet string
                a__srDatatype = "DataType : octet string"
                pdi.dataLen  = aNElem
                resultAxioRObj  = axio_wobj(pdi, a_octet);
                break

            case -10   ; -Octet string
                select(pdi.index)
                    default
                        a__srDatatype = "DataType : -Octet string - un
                        pdi.dataLen  = aNElem
                        resultAxioRObj  = axio_wobj(pdi, a_octet);
                        break
                    case 0x80
                        a__srDatatype = "DataType : -Octet string - CN
                        pdi.dataLen  = sizeof(a_paraCntInc)
                        resultAxioRObj  = axio_wobj(pdi, a_paracntinc)
                        break
                end_select
                break

        end_select

        dtRAXIO = time_micro() - tActAxio;
        if((resultAxioWObj < 0) or (resultAxioWObj > 2))
            opAXIOobj = -100;
            return
        endif
        _if (opAXIOobj> 0)
            opAXIOobj = 0;
        break    ; CASE 10 / -10


    case -11     ;opAxioObj = -11  continuous pass-through Write on AXI
    case 11      ;opAxioObj = 11   pass-through Write on AXIO device
        pdi.rid      = aRid
        pdi.slot     = aSlot
```

© 2025 Robox SpA

```
        pdi.index    = aIndex
        pdi.subindex = aSubi
        pdi.dataType = aDataType
        pdi.phase    = 1
        tActAxio     = time_micro()
    case -12
    case 12
        select(aWhatWrite)
            case 9     ; variable string
                a__srDatatype = "DataType : visible string"
                pdi.dataLen  = str_len(a_VS) +1
                resultAxioRObj  = axio_wobj(pdi, a_vs);
                break
            case 6     ; unsigned 16
                a__srDatatype = "DataType : unsigned 16"
                pdi.dataLen  = sizeof(a_U16)
                resultAxioRObj  = axio_wobj(pdi, a_u16);
                break
            case -6    ; array unsigned 16
                a__srDatatype = "DataType : array unsigned 16"
                pdi.dataLen  = sizeof(a_U16) * aNElem
                resultAxioRObj  = axio_wobj(pdi, a_au16);
                break
            case 3     ; integer 16
                a__srDatatype = "DataType : integer 16"
                pdi.dataLen  = sizeof(a_I16)
                resultAxioRObj  = axio_wobj(pdi, a_i16);
                break
            case -3    ; array integer 16
                a__srDatatype = "DataType : array integer 16"
                pdi.dataLen  = sizeof(a_I16) * aNElem
                resultAxioRObj  = axio_wobj(pdi, a_ai16);
                break
            case 5     ; unsigned 8
                a__srDatatype = "DataType : unsigned 8"
                pdi.dataLen  = sizeof(a_U8)
                resultAxioRObj  = axio_wobj(pdi, a_u8);
                break
            case 2     ; integer 8
                a__srDatatype = "DataType : integer 8"
                pdi.dataLen  = sizeof(a_I8)
                resultAxioRObj  = axio_wobj(pdi, a_i8);
                break
            case 10    ; Octet string
                a__srDatatype = "DataType : octet string"
                pdi.dataLen  = aNElem
                resultAxioRObj  = axio_wobj(pdi, a_octet);
                break

            case -10   ; -Octet string
                select(pdi.index)
```

© 2025 Robox SpA

```
                    default
                        a__srDatatype = "DataType : -Octet string - un
                        pdi.dataLen  = aNElem
                        resultAxioRObj  = axio_wobj(pdi, a_octet);
                        break
                    case 0x80
                        a__srDatatype = "DataType : -Octet string - CN
                        pdi.dataLen  = sizeof(a_paraCntInc)
                        resultAxioRObj  = axio_wobj(pdi, a_paracntinc)
                        break
                end_select
                break

        end_select

        if(resultAxioWObj = 0)
            opAXIOobj = cond_val(opAXIOobj < 0, -12, 12)
            break
        endif
        dtRAXIO = time_micro() - tActAxio;
        if((resultAxioWObj < 0) or (resultAxioWObj > 2))
            opAXIOobj = -100;
            return
        endif
        if(opAXIOobj> 0)
            opAXIOobj = 0;
        else
            opAXIOobj =  -11
        endif
        break
    end_select
endfun
```

Funzione b0()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 0 dell'argomento.

Lettura:

Sintassi	**u8 b0 (i8|i16|i32|u8|u16|u32|real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto del primo byte dell'argomento *arg*.
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**u8 b0 (i8|i16|i32|u8|u16|u32|real arg)**
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**b0 (i8|i16|i32|u8|u16|u32|real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Esempi

Esempio di utilizzo:

```
int pluto

; se pluto vale 0x1234
r(1) = b0[213](pluto) ; r(1)= 0x0034
r(1) = b1[215](pluto) ; r(1)= 0x0012

;-----------------------------------------------
; definisco variabile max come integer 32 bit
int max
b0(max) = 0x11
b1(max) = 0x22
b2[215](max) = 0x33
b3[216](max) = 0x44
; alla fine max sarà = 0x44332211

;-----------------------------------------------
; con RTE 33.9.0:
int pippo = 123
b4[216](pippo) = 9
; errore!
```

© 2025 Robox SpA

Funzione b1()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 1 dell'argomento.

Lettura:

Sintassi	**u8 b1 (i16|i32|u16|u32|real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto del secondo byte dell'argomento *arg*.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**b1 (i16|i32|u16|u32|real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Funzione b2()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 2 dell'argomento.

Lettura:

Sintassi	**u8 b2 (i32|u32|real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto del terzo byte dell'argomento *arg*.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

© 2025 Robox SpA

Sintassi	**b2 (i32|u32|real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Funzione b3()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 3 dell'argomento.

Lettura:

Sintassi	**u8 b3 (i32|u32|real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto del quarto byte dell'argomento *arg*.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**b3 (i32|u32|real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Funzione b4()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 4 dell'argomento.

Lettura:

Sintassi	**u8 b4 (real arg)**
arg	Variabile di cui si vuole conoscere il contenuto.

© 2025 Robox SpA

Sintassi	**u8 b4 (real arg)**
	Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto del quinto byte dell'argomento *arg*.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**b4 (real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Funzione b5()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 5 dell'argomento.

Lettura:

Sintassi	**u8 b5 (real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto del sesto byte dell'argomento *arg*.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**b5 (real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**b5 (real arg) = u8 value**
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Funzione b6()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 6 dell'argomento.

Lettura:

Sintassi	**u8 b6 (real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto del settimo byte dell'argomento *arg*.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**b6 (real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Funzione b7()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 7 dell'argomento.

Lettura:

Sintassi	**u8 b7 (real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto dell'ottavo byte dell'argomento *arg*.

© 2025 Robox SpA

Sintassi	**u8 b7 (real arg)**
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**b7 (real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Funzione bcc_get_double()

Questa funzione permette di leggere un real dal campo 'data' di una struttura bcc_msguser 820 all'offset specificato.

Sintassi	**real bcc_get_double (bcc_msguser stru, i32 offset)**
stru	Struttura di tipo bcc_msguser 820.
offset	Offset in numero di byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
real valueRead
int offset = 10

valueRead = bcc_get_double 219(bcc_str, offset)
```

© 2025 Robox SpA

Funzione bcc_get_float()

Questa funzione permette di leggere un float dal campo 'data' di una struttura bcc_msguser 820 all'offset specificato.

Sintassi	**float bcc_get_float (bcc_msguser stru, i32 offset)**
stru	Struttura di tipo bcc_msguser 820.
offset	Offset in numero di byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
float valueRead
int offset = 10

valueRead = bcc_get_float 220(bcc_str, offset)
```

Funzione bcc_get_i16()

Questa funzione permette di leggere un i16 dal campo 'data' di una struttura bcc_msguser 820 all'offset specificato.

Sintassi	**i16 bcc_get_i16 (bcc_msguser stru, i32 offset)**
stru	Struttura di tipo bcc_msguser 820.
offset	Offset in numero di byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
i16 valueRead
int offset = 10

valueRead = bcc_get_i16 220(bcc_str, offset)
```

© 2025 Robox SpA

Funzione bcc_get_i32()

Questa funzione permette di leggere un i32 dal campo 'data' di una struttura bcc_msguser 820 all'offset specificato.

Sintassi	**i32 bcc_get_i32 (bcc_msguser stru, i32 offset)**
stru	Struttura di tipo bcc_msguser 820.
offset	Offset in numero di byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
int valueRead
int offset = 10

valueRead = bcc_get_i32 221(bcc_str, offset)
```

Funzione bcc_get_i8()

Questa funzione permette di leggere un i8 dal campo 'data' di una struttura bcc_msguser 820 all'offset specificato.

Sintassi	**i8 bcc_get_i8 (bcc_msguser stru, i32 offset)**
stru	Struttura di tipo bcc_msguser 820.
offset	Offset in numero di byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
i8 valueRead
int offset = 10

valueRead = bcc_get_i8 221(bcc_str, offset)
```

© 2025 Robox SpA

Funzione bcc_get_string()

Questa funzione permette di leggere una string dal campo 'data' di una struttura bcc_msguser 820 all'offset specificato.

Sintassi	**string bcc_get_string (bcc_msguser stru, i32 offset)**
stru	Struttura di tipo bcc_msguser 820.
offset	Offset in numero di byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
string valueRead
int offset = 10

valueRead = bcc_get_string 222(bcc_str, offset)
```

Funzione bcc_get_tcp_clients_info()

Questa funzione permette di ottenere informazioni sui client connessi.

Sintassi	**i32 bcc_get_tcp_clients_info (stru_bcc_tcp_client buff, u32 nElem [, u32 idx=0])**
buff	Struttura di tipo stru_bcc_tcp_client 830 o array di strutture.
nElem	Capacità dell'array di strutture.
idx	Indice dell'offset nella lista dei client. (opzionale, default 0)
Risultato	Rende:

Valore	Descrizione
>= 0	Numero del client connesso.
-3	La dimensione della struttura del compilatore non

© 2025 Robox SpA

Sintassi	**i32 bcc_get_tcp_clients_info (stru_bcc_tcp_client buff, u32 nElem [, u32 idx=0])**
	Valore / Descrizione
	(vuoto) / corrisponde a quella di RTE.
	-4 / *nElem* = 0
	-5 / Valore illegale di *idx.*
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
lit MAX_TCP_CLIENTS 32

int num_of_client
stru_bcc_tcp_client[830] cl[MAX_TCP_CLIENTS]

; ...
num_of_client = bcc_get_tcp_clients_info[222](cl[0], MAX_TCP_CLIENTS)

if(num_of_client < 0)
    ; manage the error
    ; ...
endif
```

Funzione bcc_get_u16()

Questa funzione permette di leggere un u16 dal campo 'data' di una struttura bcc_msguser[820] all'offset specificato.

Sintassi	**u16 bcc_get_u16 (bcc_msguser stru, i32 offset)**
stru	Struttura di tipo bcc_msguser[820].
offset	Offset in numero di byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

© 2025 Robox SpA

```
bcc_msguser bcc_str
u16 valueRead
int offset = 10

valueRead = bcc_get_u16(bcc_str, offset)
```

Funzione bcc_get_u32()

Questa funzione permette di leggere un u32 dal campo 'data' di una struttura bcc_msguser all'offset specificato.

Sintassi	**u32 bcc_get_u32 (bcc_msguser stru, i32 offset)**
stru	Struttura di tipo bcc_msguser.
offset	Offset in numero di byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
u32 valueRead
int offset = 10

valueRead = bcc_get_u32(bcc_str, offset)
```

Funzione bcc_get_u8()

Questa funzione permette di leggere un u8 dal campo 'data' di una struttura bcc_msguser all'offset specificato.

Sintassi	**u8 bcc_get_u8 (bcc_msguser stru, i32 offset)**
stru	Struttura di tipo bcc_msguser.
offset	Offset in numero di byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
bcc_msguser bcc_str
u8 valueRead
int offset = 10

valueRead = bcc_get_u8(bcc_str, offset)
```

Funzione bcc_h_setup()

Questa funzione esegue l'impostazione dei dati che servono per rispondere ad una messaggio BCC.

Sintassi	**bool bcc_h_setup (bcc_msguser struRx, bcc_msguser struTx, i32 len)**
struRx	Struttura bcc_msguser in cui è stato ricevuto il messaggio.
struTx	Struttura bcc_msguser usata per i dati da inviare. **NOTA**: La struttura dei dati da spedire può essere la stessa dei dati ricevuti.
len	Lunghezza dei dati dei messaggi (max 255 byte).
Risultato	Rende: Valore: TRUE — Operazione effettuata correttamente. Valore: FALSE — Riscontrati parametri illegali.
Validità	Rule, Task.
Note	Le operazioni eseguite sono (logicamente) le seguenti: *struTx*.resSys1 = *struRx*.resSys1 *struTx*.resSys2 = *struRx*.resSys2 *struTx*.bccID = *struRx*.bccID *struTx*.DST = *struRx*.SR *struTx*.DCH = *struRx*.SCH *struTx*.SRC = *struRx*.DST *struTx*.SCH = *struRx*.DCH *struTx*.MSG = BCC_ACK (0x1) *struTx*.LEN = *len*

© 2025 Robox SpA

Sintassi	**bool bcc_h_setup (bcc_msguser struRx, bcc_msguser struTx, i32 len)**
	struTx.PID = *struRx*.PID *struTx*.DLC = 0xFF

La risposta viene indirizzata al sorgente e viene utilizzato lo stesso dispositivo fisico.

Esempi

Esempio di utilizzo:

```
; vediamo se sul canale 190 c'è un messaggio da scaricare
reso = bcc_receive 226(190, blocco) > 0
_if(reso > 0)
    call 694 scaricaDati
goto loopT1
; fine codice task 1

;---------------------------------------------
;---------------------------------------------
; F U N Z I O N E
;---------------------------------------------
;---------------------------------------------

function 715 scaricaDati
    ; copia dei dati ricevuti nella struttura blocco.data
    ; a partire dal regisro intero 2000 per 10 Bytes
    move_byte 731(blocco.data, r(2000), 5*4)
    ; mettiamo a disposizione del master il registro rr(1)
    ; che incrementa (watch-dog)
    inc 377(rr(1))
    ; copiamo il registro reale 1 nella nel campo data della
    ; struttura blocco (RR = 8 bytes)
    move_byte(rr(1), blocco.data, 8)
    ; rispondiamo al master utilizzando lo stesso header
    reso = bcc_h_setup 225(blocco, blocco, 8)
    reso = bcc_send 229(190, blocco)
end_fun
```

Funzione bcc_receive()

Questa istruzione permette di leggere i messaggi utente in arrivo su canali predefiniti BCC3.0

Sintassi	**i32 bcc_receive (i32 idCh, bcc_msguser stru)**
idCh	Indice del canale (190 ÷ 199).
stru	Struttura bcc_msguser 820 dichiarata localmente in cui verrà scaricato il messaggio ricevuto.

© 2025 Robox SpA

Sintassi	**i32 bcc_receive (i32 idCh, bcc_msguser stru)**
Risultato	Rende:

Valore	Descrizione
> 0	Messaggio ricevuto.
0	Nessun messaggio ricevuto.
-1, -2, -3	Parametri illegali.
-4, -5	Funzione illegale (nessun dispositivo configurato BCC3).

Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; definisco variabile di tipo struttura BCC_MSGUSER
bcc_msguser 820 blocco

reso = bcc_receive 226 (190, blocco) > 0
_if (reso > 0)
    call 694 scaricaDati
```

Funzione bcc_reply_ack()

Questa funzione permette di inviare una risposta positiva (Ack) ad un messaggio ricevuto.

Sintassi	**i32 bcc_reply_ack (bcc_msguser stru)**
stru	Struttura bcc_msguser 820 per la gestione dei messaggi BCC.
Risultato	Rende:

Valore	Descrizione
> 0	Messaggio ricevuto.
0	Nessun messaggio ricevuto.
-1, -2, -3	Parametri illegali.
-4, -5	Funzione illegale (nessun dispositivo configurato BCC3).

© 2025 Robox SpA

Sintassi	**i32 bcc_reply_ack (bcc_msguser stru)**
Validità	Rule, Task.
Note	

Funzione bcc_reply_msg()

Questa funzione permette di inviare una risposta ad un messaggio ricevuto.

Sintassi	**i32 bcc_reply_msg (bcc_msguser struRx, bcc_msguser struTx, i32 code=0, i32 len=-1)**
struRx	Struttura bcc_msguser 820 di gestione BCC del messaggio a cui rispondere.
struTx	Struttura bcc_msguser 820 di gestione BCC con le informazione della risposta.
code	Se uguale a 0, il codice del messaggio (campo 'msg' della struttura bcc_msguser) da utilizzare nella risposta sarà quello della struttura *struTx*, altrimenti verrà utilizzato *code*. (default 0)
len	Se uguale a -1, la lunghezza del messaggio (campo 'lendata' della struttura bcc_msguser) da utilizzare nella risposta sarà quella della struttura *struTx*, altrimenti verrà utilizzata *len*. (default -1)
Risultato	Rende: Valore / Descrizione > 0 — Messaggio ricevuto. 0 — Nessun messaggio ricevuto. -1, -2, -3 — Parametri illegali. -4, -5 — Funzione illegale (nessun dispositivo configurato BCC3).
Validità	Rule, Task.
Note	

© 2025 Robox SpA

Funzione bcc_reply_nack()

Questa funzione permette di inviare una risposta negativa (Nack) ad un messaggio ricevuto.

Sintassi	**i32 bcc_reply_nack (bcc_msguser stru, i32 nAck, i32 par=0)**
stru	Struttura bcc_msguser 820 per la gestione dei messaggi BCC.
nAck	Codice di errore.
par	Parametro ausiliario. (default 0)
Risultato	Rende:

Valore	Descrizione
> 0	Messaggio ricevuto.
0	Nessun messaggio ricevuto.
-1, -2, -3	Parametri illegali.
-4, -5	Funzione illegale (nessun dispositivo configurato BCC3).

Validità	Rule, Task.
Note	

Funzione bcc_send()

Questa funzione permette di inviare i messaggi utente sui canali predefiniti BCC3.

Sintassi	**i32 bcc_send (i32 idCh, bcc_msguser stru)**
idCh	Indice del canale (190÷199).
stru	Struttura bcc_msguser 820 dichiarata localmente da cui verrà letto il messaggio da inviare.
Risultato	Rende:

Valore	Descrizione
> 0	Messaggio inviato.
0	Nessun messaggio inviato.
-1, -2, -3	Parametri illegali.
-4, -5	Funzione illegale (nessun dispositivo

© 2025 Robox SpA

Sintassi	**i32 bcc_send (i32 idCh, bcc_msguser stru)**
	Valore: — / Descrizione: configurato BCC3).
Validità	Rule, Task.
Note	Perchè il messaggio venga trasmesso è necessario che i dati di intestazione siano stati inseriti correttamente. Questo può esser fatto con l'istruzione ser_bcc_h_init 587().

Esempi

Esempio di utilizzo:

```
reso = ser_bcc_h_init 587(blocco, 1, 190, 190, 10000, 5*4)
function 715 spedisci
    ; copia a partire da R(1000), 5*4 bytes ossia 5 registri interi, n
    ; campo data della struttura blocco
    move_byte 731(r(1000), blocco.data, 5*4)

    ; invio della struttura blocco sul canale 190 (a disposizione i ca
    ; reso da' informazioni diagnostiche
    reso = bcc_send 229(190, blocco)
end_fun
```

Funzione check_mv_follow2()

Questa funzione serve per verificare se è possibile eseguire la mv_follow2 458() con i dati specificati. La funzione se ha avuto esito negativo rende un codice di errore; altrimenti, se ok, può rendere anche il tempo di durata massimo del sincronismo.

Sintassi	**i32 check_mv_follow2 (stru_follow2 stru, real masterSpe, real syncTime)**
stru	Struttura di tipo stru_follow2 846 contenente i parametri relativi alla movimentazione.
masterSpe	Velocità del master a regime.
syncTime	Variabile in cui verrà scritto la durata massima del sincronismo.
Risultato	Rende: Valore: -5 / Descrizione: Numero asse non valido. Valore: -1 / Descrizione: *masterSpe* <= 0.

© 2025 Robox SpA

Sintassi	**i32 check_mv_follow2 (stru_follow2 stru, real masterSpe, real syncTime)**

Valore	Descrizione
0	Operazione eseguita con successo. È possibile eseguire la mv_follow2 458(), in *syncTime* è possibile leggere la durata massima del sincronismo.
1	Accelerazione massima dello slave <= 0.
2	Lo spazio di partenza (sync position / park position) oppure lo spazio di frenata sono <= 0.
3	'kSlaveMaster' <= 0.
4	Slave rotativo -> spazio master di partenza + spazio di frenata insufficienti.
5	Slave rotativo -> lo spazio master disponibile allo slave per raggiungere il prossimo sincronismo è insufficiente.
	Slave lineare -> lo spazio master disponibile allo slave per tornare indietro è insufficiente.
6	'masterCycle' <= 0.
21	Lo slave supera la velocità massima mentre si sposta dalla posizione di parcheggio alla posizione di sincronismo.
22	Lo slave supera l'accelerazione massima mentre si sposta dalla posizione di parcheggio alla posizione di sincronismo.

© 2025 Robox SpA

Sintassi	**i32 check_mv_follow2 (stru_follow2 stru, real masterSpe, real syncTime)**

Valore	Descrizione
31	Slave lineare -> lo slave supera la velocità massima durante la frenata dopo il sincronismo.
32	Slave lineare -> lo slave supera l'accelerazione massima durante la frenata dopo il sincronismo.
41	Slave lineare -> lo slave supera la velocità massima mentre torna alla posizione iniziale.
42	Slave lineare -> lo slave supera l'accelerazione massima mentre torna alla posizione iniziale.
51	Slave rotativo -> lo slave supera la velocità massima mentre si sposta alla posizione di parcheggio dopo il sincronismo.
52	Slave rotativo -> lo slave supera l'accelerazione massima mentre si sposta alla posizione di parcheggio dopo il sincronismo.
61	Slave rotativo -> lo slave supera la velocità massima mentre si sposta dalla posizione di sincronismo attuale alla successiva, lo slave si ferma per aspettare lo spazio master necessario.
62	Slave rotativo -> lo slave supera l'accelerazione massima mentre si sposta dalla posizione di sincronismo attuale alla successiva, lo slave si ferma per aspettare lo spazio master necessario.

© 2025 Robox SpA

Sintassi	**i32 check_mv_follow2 (stru_follow2 stru, real masterSpe, real syncTime)**
	Valore / Descrizione
	71 — Slave rotativo -> lo slave supera la velocità massima mentre si sposta dalla posizione di sincronismo attuale alla successiva.
	72 — Slave rotativo -> lo slave supera l'accelerazione massima mentre si sposta dalla posizione di sincronismo attuale alla successiva.
Validità	Rule, Task.
Note	

Funzione check_mva_follow2()

Questa funzione serve per verificare se è possibile eseguire la mva_follow2() con i dati specificati. La funzione se ha avuto esito negativo rende un codice di errore; altrimenti, se ok, può rendere anche il tempo di durata massimo del sincronismo.

Sintassi	**i32 check_mva_follow2 (stru_follow2_s stru, real masterSpe, real syncTime)**
stru	Struttura di tipo stru_follow2_s contenente i parametri relativi alla movimentazione.
masterSpe	Velocità del master a regime.
syncTime	Variabile in cui verrà scritto la durata massima del sincronismo.
Risultato	Rende:

Valore	Descrizione
-5	Numero asse non valido.
-1	*masterSpe* <= 0.
0	Operazione eseguita con successo. È possibile eseguire la mva_follow2(), in *syncTime* è possibile leggere la durata massima del sincronismo.

© 2025 Robox SpA

Sintassi	**i32 check_mva_follow2 (stru_follow2_s stru, real masterSpe, real syncTime)**

Valore	Descrizione
1	Accelerazione massima dello slave <= 0.
2	Lo spazio di partenza (sync position / park position) oppure lo spazio di frenata sono <= 0.
3	'kSlaveMaster' <= 0.
4	Slave rotativo -> spazio master di partenza + spazio di frenata insufficienti.
5	Slave rotativo -> lo spazio master disponibile allo slave per raggiungere il prossimo sincronismo è insufficiente.
	Slave lineare -> lo spazio master disponibile allo slave per tornare indietro è insufficiente.
6	'masterCycle' <= 0.
21	Lo slave supera la velocità massima mentre si sposta dalla posizione di parcheggio alla posizione di sincronismo.
22	Lo slave supera l'accelerazione massima mentre si sposta dalla posizione di parcheggio alla posizione di sincronismo.
31	Slave lineare -> lo slave supera la velocità massima durante la frenata dopo il sincronismo.
32	Slave lineare -> lo slave supera l'accelerazione massima durante la

© 2025 Robox SpA

Sintassi	**i32 check_mva_follow2 (stru_follow2_s stru, real masterSpe, real syncTime)**

Valore	Descrizione
	frenata dopo il sincronismo.
41	Slave lineare -> lo slave supera la velocità massima mentre torna alla posizione iniziale.
42	Slave lineare -> lo slave supera l'accelerazione massima mentre torna alla posizione iniziale.
51	Slave rotativo -> lo slave supera la velocità massima mentre si sposta alla posizione di parcheggio dopo il sincronismo.
52	Slave rotativo -> lo slave supera l'accelerazione massima mentre si sposta alla posizione di parcheggio dopo il sincronismo.
61	Slave rotativo -> lo slave supera la velocità massima mentre si sposta dalla posizione di sincronismo attuale alla successiva, lo slave si ferma per aspettare lo spazio master necessario.
62	Slave rotativo -> lo slave supera l'accelerazione massima mentre si sposta dalla posizione di sincronismo attuale alla successiva, lo slave si ferma per aspettare lo spazio master necessario.
71	Slave rotativo -> lo slave supera la velocità massima mentre si sposta dalla posizione di sincronismo attuale alla successiva.

© 2025 Robox SpA

Sintassi	**i32 check_mva_follow2 (stru_follow2_s stru, real masterSpe, real syncTime)**
	Valore / Descrizione
	72 — Slave rotativo -> lo slave supera l'accelerazione massima mentre si sposta dalla posizione di sincronismo attuale alla successiva.
Validità	Rule, Task.
Note	

Funzione check_mva_follow2_m()

Questa funzione serve per verificare se è possibile eseguire la mva_follow2_m458() con i dati specificati. La funzione se ha avuto esito negativo rende un codice di errore; altrimenti, se ok, può rendere anche il tempo di durata massimo del sincronismo.

Sintassi	**i32 check_mva_follow2 (stru_follow2_sm stru, real masterSpe, real syncTime)**
stru	Struttura di tipo stru_follow2_sm862 contenente i parametri relativi alla movimentazione.
masterSpe	Velocità del master a regime.
syncTime	Variabile in cui verrà scritto la durata massima del sincronismo.
Risultato	Rende:

Valore	Descrizione
-5	Numero asse non valido.
-1	*masterSpe* <= 0.
0	Operazione eseguita con successo. È possibile eseguire la mva_follow2_m458(), in *syncTime* è possibile leggere la durata massima del sincronismo.
1	Accelerazione massima dello slave <= 0.
2	Lo spazio di partenza (sync position / park

© 2025 Robox SpA

Sintassi	**i32 check_mva_follow2 (stru_follow2_sm stru, real masterSpe, real syncTime)**

Valore	Descrizione
	position) oppure lo spazio di frenata sono <= 0.
3	'kSlaveMaster' <= 0.
4	Slave rotativo -> spazio master di partenza + spazio di frenata insufficienti.
5	Slave rotativo -> lo spazio master disponibile allo slave per raggiungere il prossimo sincronismo è insufficiente.
	Slave lineare -> lo spazio master disponibile allo slave per tornare indietro è insufficiente.
6	'masterCycle' <= 0.
21	Lo slave supera la velocità massima mentre si sposta dalla posizione di parcheggio alla posizione di sincronismo.
22	Lo slave supera l'accelerazione massima mentre si sposta dalla posizione di parcheggio alla posizione di sincronismo.
31	Slave lineare -> lo slave supera la velocità massima durante la frenata dopo il sincronismo.
32	Slave lineare -> lo slave supera l'accelerazione massima durante la frenata dopo il sincronismo.
41	Slave lineare -> lo slave supera la velocità

© 2025 Robox SpA

Sintassi	**i32 check_mva_follow2 (stru_follow2_sm stru, real masterSpe, real syncTime)**

Valore	Descrizione
	massima mentre torna alla posizione iniziale.
42	Slave lineare -> lo slave supera l'accelerazione massima mentre torna alla posizione iniziale.
51	Slave rotativo -> lo slave supera la velocità massima mentre si sposta alla posizione di parcheggio dopo il sincronismo.
52	Slave rotativo -> lo slave supera l'accelerazione massima mentre si sposta alla posizione di parcheggio dopo il sincronismo.
61	Slave rotativo -> lo slave supera la velocità massima mentre si sposta dalla posizione di sincronismo attuale alla successiva, lo slave si ferma per aspettare lo spazio master necessario.
62	Slave rotativo -> lo slave supera l'accelerazione massima mentre si sposta dalla posizione di sincronismo attuale alla successiva, lo slave si ferma per aspettare lo spazio master necessario.
71	Slave rotativo -> lo slave supera la velocità massima mentre si sposta dalla posizione di sincronismo attuale alla successiva.
72	Slave rotativo -> lo slave supera l'accelerazione massima mentre si sposta dalla posizione di sincronismo attuale alla successiva.

© 2025 Robox SpA

Sintassi	**i32 check_mva_follow2 (stru_follow2_sm stru, real masterSpe, real syncTime)**
Validità	Rule, Task.
Note	

Funzione co_asw_sdo()

ATTENZIONE: Questa funzione non può essere usata in un task se nello stesso sono presenti anche le funzioni co_robj 258()/co_wobj 273()/coe_robj 279()/coe_wobj 284().

Questa funzione permette di:

- Testare se il precedente comando SDO co_send_sdo 266(), usato per scrivere dei dati, è stato eseguito -> co_asw_sdo(*nWs*, *handle*)
- Testare se il precedente comando SDO co_send_sdo(), usato per leggere dei dati, è stato eseguito -> co_asw_sdo(*nWs*, *handle*)
- Testare se il precedente comando SDO co_send_sdo(), usato per leggere dei dati, è stato eseguito e contemporaneamente per scaricare i dati -> co_asw_sdo(*nWs*, *handle*, *type*, *data*)

Sintassi	**i32 co_asw_sdo (i32 nWs, i32 handle)** **i32 co_asw_sdo (i32 nWs, i32 handle, i32 type, real data)**
nWs	ID stazione Robox (workstation logica 1-:-64). **NOTA**: Se si specifica 0 come ws allora si intende per tutte.
handle	Handle ricevuto dalla precedente esecuzione dell'istruzione co_send_sdo 266().
type	Permette di selezionare come interpretare i dati da scaricare in *data*:

Valore	Tipo
0	Nessun oggetto.
1	Unsigned 8 bit.
2	Unsigned 16 bit.
3	Unsigned 24 bit.
4	Unsigned 32 bit.
5	Real 32 bit.
-1	Signed 8 bit.
-2	Signed 16 bit.
-3	Signed 24 bit.

© 2025 Robox SpA

Sintassi	**i32 co_asw_sdo (i32 nWs, i32 handle)** **i32 co_asw_sdo (i32 nWs, i32 handle, i32 type, real data)**
	Valore: -4 — Tipo: Signed 32 bit.
data	Variabile locale dove vengono scaricati i dati (da formato Fix di dimensione specificata a formato real) se *type* positivo.
Risultato	Rende: -2: Riferimento di messaggio non più valido. -1: Dati illegali (viene generato l'allarme 9900). 0: Messaggio in attesa di risposta. 1: Comando eseguito e ottenuto risposta. > 1: Rappresenta il codice SDO abort code secondo lo standard CANopen.
Validità	Rule, Task.
Note	Da RTE 34.13.5 sono disponibili le funzioni co_robj() e co_wobj().

Esempi

Esempio di utilizzo:

```
; if there is a reading command
if(rise(command_read))
    command_read = false
    faseRead = SEND
endif

select(faseRead)
    case IDLE
        break

    case SEND
```

© 2025 Robox SpA

```
        handle = co_send_sdo(rNws, rObj, rSub)
        if(handle < 0)
            faseRead = ERROR
            break
        endif
        faseRead = GET_DATA
        break

    case GET_DATA
        result = co_asw_sdo(rNws, handle, rType, Data)
        ; waiting answer
        _if(result = 0)
            break
        ; test answer
        if(result = 1)
            faseRead = DONE
        else
            faseRead = ERROR
        endif
        break

    case DONE
        ; insert in the MREPORT diagnosys from SHELL
        str_format(srInfo, "SDO Read ws:%d, obj: %d, sub:%d, returne
        inf_report(0x1f,srInfo)
        faseRead = IDLE
        break

    case ERROR
        ; insert in the MREPORT diagnosys from SHELL
        str_format(srInfo, "Error SDO read value:%d", result)
        inf_report(0x1f, srInfo)
        faseRead = IDLE
        break
    end_select
```

Funzione co_controlword_om()

Questa funzione serve per imporre i bit 'manufacturer specific', 'halt' e 'operation mode specific' della controlword CAN/CoE (oggetto 6040H).

Sintassi	**i32 co_controlword_om (i32 nWs, i32 state, i32 bitMsk)**
nWs	ID stazione Robox (workstation logica 1-:-64). **NOTA**: Se si specifica 0 come *ws* allora si intende per tutte.
state	Valore da imporre ai bit specificati.
bitMsk	Maschera dei bit predisposti ad assumere lo stato.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 co_controlword_om (i32 nWs, i32 state, i32 bitMsk)</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td><> 0</td><td>Operazione avvenuta con successo.</td></tr>
<tr><td>0</td><td>Riscontrati parametri illegali.</td></tr>
</table></td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td>La maschera dei bit imponibili con questa istruzione è 0x0000FD70, che corrisponde a:
<table>
<tr><th>Bit</th><th>Valore Hex</th><th>Significato</th></tr>
<tr><td>4, 5, 6</td><td>0x00000070</td><td>Operation mode specific.</td></tr>
<tr><td>8</td><td>0x00000100</td><td>Halt.</td></tr>
<tr><td>10</td><td>0x00000400</td><td>Da RTE 34.16.6.</td></tr>
<tr><td>11, 12, 13, 14, 15</td><td>0x0000F800</td><td>Manufacturer specific.</td></tr>
</table>
Si ricorda che nel file di configurazione rte.cfg, o da configuratore grafico di progetto (proprietà assi), è possibile impostare i bit 'operation mode specific' tramite la keyword axis_can.</td></tr>
</table>

co_controlword_om() per dispositivi SOE

Da RTE 34.18.4 è possibile utilizzare questa funzione anche con dispostivi SOE. I bit scrivibili sono 8, 9, 11 e la maschera corrispondente è 0x00000B00.

Tali bit corrispondono a:

Bit	Valore Hex	Significato
-	0x00000000	Primary op mode.
8	0x00000100	Secondary mode 1.
9	0x00000200	Secondary mode 2.
8, 9	0x00000300	Secondary mode 3.
11	0x00000800	Secondary mode 4.
8, 11	0x00000900	Secondary mode 5.
9, 11	0x00000A00	Secondary mode 6.
8, 9, 11	0x00000B00	Secondary mode 7.

© 2025 Robox SpA

Definizione bit controlword

Bit	15-14-13-12-11	10	9	8	7	6-5-4	3	2	1	0
Definizione	Manufacturer specific	Reserved	Reserved	Halt	Fault reset	Operation mode specific	Enable operation	Quick stop	Enable voltage	Switch on
Bit imponibili da co_controlword_om()	X	X		X		X				

Esempi

Esempio di utilizzo:

```
; transizione up del bit 0x0010
reso = co_controlword_om(1,0x0010,0x0010)
; transizione down del bit 0x0010
reso = co_controlword_om(1,0x0000,0x0010)
```

Funzione co_controlword_state()

Funzione che impone lo stato della controlword.

Sintassi	**i32 co_controlword_state (i32 nWs , i32 state)**
nWs	ID Stazione Robox (workstation logica 1-:-64). **NOTA**: Se si specifica 0 come *ws* allora si intende per tutte.
state	Stati possibili in accordo con la macchina a stati DSP 402:

Valore	Descrizione
-2	DISABLE ENABLE (modale).
-1	POWER ON DISABLED (modale).
1	POWER ON ENABLED (modale).
2	ENABLE (modale ma resettato se si da

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 co_controlword_state (i32 nWs , i32 state)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td></td><td>quick stop).</td></tr>
<tr><td>3</td><td>DISABLE POWER (autoresettante).</td></tr>
<tr><td>4</td><td>QUICK STOP (autoresettante).</td></tr>
<tr><td>5</td><td>RESET FAULT (autoresettante).</td></tr>
</table>
</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td><> 0</td><td>Operazione avvenuta con successo.</td></tr>
<tr><td>0</td><td>Riscontrati parametri illegali.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td>La maschera dei bit imponibili con questa istruzione è 0x0000008F, che corrisponde a:
<table>
<tr><th>Bit</th><th>Valore Hex</th><th>Significato</th></tr>
<tr><td>0</td><td>0x00000001</td><td>Switch on.</td></tr>
<tr><td>1</td><td>0x00000002</td><td>Enable voltage.</td></tr>
<tr><td>2</td><td>0x00000004</td><td>Quick stop.</td></tr>
<tr><td>3</td><td>0x00000008</td><td>Enable operation.</td></tr>
<tr><td>7</td><td>0x00000080</td><td>Fault reset.</td></tr>
</table>
</td></tr>
</table>

Se l'asse è gestito mediante un PowerSet la gestione della macchina a stati DS402 viene effettuata da RTE.

L'utilizzo di questa funzione è quindi necessario quando:

- la gestione di un dispositivo non viene fatta mediante un PowerSet
- per comandare un quick stop indipendentemente dalla gestione di RTE
- per comandare un reset fault indipendentemente dalla gestione di RTE

Per sapere il comando attuale presente sulla controlword utilizzare la funzione co_statusword 270().

Definizione bit controlword

© 2025 Robox SpA

Bit	15-14-13-12-11	10-9	8	7	6-5-4	3	2	1	0
Definizione	Manufacturer specific	Reserved	Halt	Fault reset	Operation mode specific	Enable operation	Quick stop	Enable voltage	Switch on
Bit impostabili da co_controlword_state()				X		X	X	X	X

Esempi

Esempio di utilizzo:

```
; comando, a tutte le ws, di power on enable
reso = co_controlword_state(0, 1)
```

Funzione co_get_ws_diagno()

Funzione per leggere informazioni di stato della workstation indicata.

Sintassi	**i32 co_get_ws_diagno (i32 nWs)**
nWs	ID Stazione Robox (workstation logica 1-:-64). **NOTA**: Se si specifica 0 come *ws* allora si intende per tutte.
Risultato	Rende il codice di stato. Il significato dei bit che compongono il codice di stato è:

Bit	Valore esadecimale	Significato
0	0x00000001	Rx PDO 1 mancante
1	0x00000002	Rx PDO 2 mancante
2	0x00000004	Rx PDO 3 mancante
3	0x00000008	Rx PDO 4 mancante
4	0x00000010	Rx PDO 1 ricevuto prima del dovuto

© 2025 Robox SpA

Sintassi	**i32 co_get_ws_diagno (i32 nWs)**

Bit	Valore esadecimale	Significato
5	0x00000020	Rx PDO 2 ricevuto prima del dovuto
6	0x00000040	Rx PDO 3 ricevuto prima del dovuto
7	0x00000080	Rx PDO 4 ricevuto prima del dovuto
8	0x00000100	Ws mandatoria mancante
9	0x00000200	Ws mancante
10	0x00000400	Ultima richiesta RTR fallita
11	0x00000800	Ws presenza imposta
12	0x00001000	Riconfigurazione Ws
13	0x00002000	Errore durante la configurazione
14	0x00004000	-
15	0x00008000	-
16	0x00010000	Presente messaggio d'emergenza
17	0x00020000	Nessuna comunicazione (in operational mode)
18	0x00040000	Tipo Ws: mandatoria
19	0x00080000	Tipo Ws: ricaricabile alla connessione
20	0x00100000	Configurazione soddisfacente eseguita

© 2025 Robox SpA

Sintassi	**i32 co_get_ws_diagno (i32 nWs)**

Bit	Valore esadecimale	Significato
21	0x00200000	Ws non connessa alla prima configurazione
22	0x00400000	Problema durante la configurazione di un comando SDO
23	0x00800000	Problema durante la configurazione di un comando NMT
24	0x01000000	Configurazione eseguita
25	0x02000000	SI in cui i dati sono letti/scritti (solo su dispositivi fisici CANbus)
26	0x04000000	Il nodo è configurato non presente (ECAT_SLAVE_OFF)
27	0x08000000	Al nodo è stata richiesta una modalità diversa da OPERATIONAL
28	0x10000000	Il master EtherCAT non è in modo OPERATIONAL
29	0x20000000	Non c'è link sulla porta EtherCAT del master
30	0x40000000	Ws non partita
31	0x80000000	Ws non definita

Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**i32 co_get_ws_diagno (i32 nWs)**
Note	Rende le stesse informazioni del comando d_co_wsdiagno da shell RDE.

Esempi

Esempio di utilizzo:

```
int canDiagno2, canDiagno3
canDiagno2 = co_get_ws_diagno(2)
canDiagno3 = co_get_ws_diagno(3)
```

Funzione co_par_download()

Funzione per caricare su un dispositivo connesso via CANopen DS301 i parametri contenuti in un file presente nella flash del controllo Robox.

Funzionamento

Le operazioni svolte da questa funzione sono:

- Apertura e analisi del file comandi, dove l'analisi consiste in:
  - Verifica esistenza della sezione "[FileInfo]"
  - Verifica esistenza della sezione "[OperationToDownloadParameters]"
  - Verifica presenza della sezione "[NodeInfo *ws*]" ed eventuale assunzione dei suoi comandi "ParameterFileName=filename" e/o "Description=text"

  Nell'eventualità che uno dei primi due punti sopra elencati non venga soddisfatto o venga riscontrato un errore, la funzione non verrà portata a termine e sarà reso il codice di errore.
- Determinazione del nome del file dal quale attingere i parametri da caricare secondo i seguenti criteri:

nameParFile	"ParameterFileName=filename"	Nome del file
Presente	Presente	Uguale a quello specificato in *nameParFile.*
Presente	Omesso	Uguale a quello specificato in *nameParFile.*
Omesso	Presente	Uguale a quello specificato in ParameterFileName.
Omesso	Omesso	Uguale a quello specificato in *nameCmdFile.*

- Esecuzione dei comandi rilevati nella sezione "[OperationToDownloadParameters]". I parametri vengono inviati nell'ordine in cui i relativi oggetti sono stati scritti nel file. Lo stato operativo del comando viene reso disponibile nella variabile predichiarata sts_co_par_download, che ovviamente potrà essere utilizzata dal programma dopo

© 2025 Robox SpA

l'esecuzione di questo passo oppure visionata tramite gli strumenti RDE anche durante l'esecuzione. Questa variabile può assumere i seguenti valori:

Valore	Descrizione
> 0	Numero di linea del file attualmente in uso (file comandi o file parametri) nel quale è stato riscontrato l'errore che ha causato l'uscita anomala dall'istruzione.
-1	Attesa esecutore disponibile.
-2	Attesa esecuzione comando.
-3	Apertura file comandi.
-4	-
-5	-
-6	-
-7	Lettura file parametri.
-8	Esecuzioni comandi file parametri.
-9	-

Nell'eventualità che venga riscontrato un errore durante la comunicazione dei parametri via SDO l'errore restituito dal dispositivo viene memorizzato nella variabile predichiarata co_sdo_err (per la spiegazione dei codici fare riferimento alla documentazione del dispositivo).

Nota: L'esecuzione di questa funzione potrebbe richiedere molto tempo in quanto dipende dalla quantità di dati da scambiare, dal traffico attuale sulla linea Can e dalla presenza di altre istruzioni che vengono eseguite allo stesso livello.

Sintassi	**i32 co_par_download (string nameCmdFile, i32 nWs, i32 flagBrk [, string nameParFile="/FA/PAX_%d.DCF"])**
nameCmdFile	Nome completo del file comandi.
nWs	Numero di workstation sulla quale si vuole operare.
flagBrk	Locazione da utilizzarsi per abortire l'esecuzione in corso. Un qualsiasi valore diverso da 0 causa l'interruzione dell'esecuzione. **ATTENZIONE**: La gestione dell'interruzione di questa funzione mediante questa variabile deve essere fatta in un altro task.
nameParFile	Stringa contenente "-F filename", dove filename è il nome completo del file parametri. (opzionale, default "/FA/PAX_%d.DCF")
Risultato	Significato dei valori restituiti:

© 2025 Robox SpA

Sintassi	**i32 co_par_download (string nameCmdFile, i32 nWs, i32 flagBrk [, string nameParFile="/FA/PAX_%d.DCF"])**

Valore	Descrizione
< 0	Codice NACK non convertito. (segnalare a Robox)
0	Operazione ok.
1	Argomento mancante. (usare mreport -a e/o la variabile predichiarata sts_co_par_download utilizzata per il reso)
2	Argomento illegale. (usare mreport -a e/o sts_co_par_download)
3	Uno dei file specificati non esiste. (usare mreport -a)
4	Uno dei file specificati è illegale. (usare mreport -a)
5	Operazione abortita. (usare mreport -a)
6	Memoria piena (RAM o CF). (usare mreport -a)
7	Impossibile rinominare il file. File già esistente. (usare mreport -a)
8	Errore in scrittura del file. (usare mreport -a)
9	Errore in lettura del file. (usare mreport -a)
10	Impossibile aggiungere ulteriori dati al file in ram durante la gestione. (usare mreport -a)
11	Errore su comando di lettura/scrittura

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 co_par_download (string nameCmdFile, i32 nWs, i32 flagBrk [, string nameParFile="/FA/PAX_%d.DCF"])</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td></td><td>oggetti su fieldbus. (usare mreport -a e/o sts_co_par_download)</td></tr>
<tr><td>12</td><td>Errore apertura file. (usare mreport -a)</td></tr>
<tr><td>13</td><td>Gestore istruzioni/comandi occupato.</td></tr>
<tr><td>14</td><td>Operazione ok ma presenti errori di oggetto o sub index non esistenti. Si verifica quando si disabilita la generazione del codice di ritorno 11 tramite sys_cfg. (usare mreport -a)</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Task.</td></tr>
<tr><td>Note</td><td></td></tr>
</table>

Funzione co_par_upload()

Funzione per aggiornare nella flash del controllo Robox un file parametri, leggendo i valori direttamente dagli azionamenti via CANopen DS301. Partendo dagli oggetti descritti nel file comandi viene generato un file parametri contenente la definizione e il valore di tali oggetti.

Da RTE 34.31.0, settando a 1 sys_cfg_2.24, è possibile salvare solo gli oggetti il cui valore è diverso dal *DefaultValue* specificato nel file template. Insieme a questi vengono salvati anche gli oggetti con *AccessType*=W e quelli il cui *DefaultValue* non è specificato.

Funzionamento

Le operazioni svolte da questa funzione sono:

- Apertura e analisi del file comandi, dove l'analisi consiste in:
  - Verifica esistenza della sezione "[FileInfo]"
  - Verifica esistenza della sezione "[OperationToUploadParameters]"
  - Verifica presenza della sezione "[NodeInfo wsNumber]" ed eventuale assunzione dei suoi comandi "ParameterFileName=filename" e/o "Description=text"

  Nell'eventualità che uno dei primi due punti sopra elencati non venga soddisfatto o venga riscontrato un errore, la funzione non verrà portata a termine e sarà reso il codice di errore.

© 2025 Robox SpA

- Determinazione del nome del file dal quale attingere i parametri da caricare secondo i seguenti criteri:

nameParFile	"ParameterFileName=filename"	Nome del file
Presente	Presente	Uguale a quello specificato in *nameParFile.*
Presente	Omesso	Uguale a quello specificato in *nameParFile.*
Omesso	Presente	Uguale a quello specificato in ParameterFileName.
Omesso	Omesso	Uguale a PAXn.DCF.

- Esecuzione dei comandi rilevati nella sezione "[OperationToUploadParameters]". I parametri vengono letti nell'ordine in cui i relativi oggetti sono stati scritti nel file comandi. Vengono assunti solo gli oggetti nelle sezioni [MandatoryObjects], [OptionalObjects] e [ManufacturerObjects]. Lo stato operativo del comando viene reso disponibile nella variabile predichiarata sts_co_par_upload, che ovviamente potrà essere utilizzata dal programma dopo l'esecuzione di questo passo oppure visionata tramite gli strumenti RDE anche durante l'esecuzione. Questa variabile può assumere i seguenti valori:

Valore	Descrizione
> 0	Numero di linea del file attualmente in uso (file comandi o file parametri) nel quale è stato riscontrato l'errore che ha causato l'uscita anomala dall'istruzione.
-1	Attesa esecutore disponibile.
-2	Attesa esecuzione comando.
-3	Apertura file comandi.
-4	-
-5	-
-6	Analisi file comandi in upload.
-7	Lettura file parametri.
-8	Esecuzioni comandi file parametri.
-9	Scrittura file parametri in corso.

  Nell'eventualità che venga riscontrato un errore durante la comunicazione dei parametri via SDO, l'errore restituito dal dispositivo viene memorizzato nella variabile predichiarata co_sdo_err (per la spiegazione dei codici fare riferimento alla documentazione del dispositivo).

NOTA: L'esecuzione di questa funzione potrebbe richiedere molto tempo in quanto dipende dalla quantità di dati da scambiare, dal traffico attuale sulla linea Can e dalla presenza di altre istruzioni che vengono eseguite allo stesso livello.

© 2025 Robox SpA

Sintassi	**i32 co_par_upload (string nameCmdFile, i32 nWs, i32 flagBrk [, string nameParFile="/FA/PAX_%d.DCF"] [, string comment=""])**
nameCmdFile	Nome completo del file comandi.
nWs	Numero di workstation sulla quale si vuole operare.
flagBrk	Locazione da utilizzarsi per abortire l'esecuzione in corso. Un qualsiasi valore diverso da 0 causa l'interruzione dell'esecuzione. **ATTENZIONE**: La gestione dell'interruzione di questa funzione mediante questa variabile deve essere fatta in un altro task.
nameParFile	Stringa contenente il nome del file da inserire nel file DCF. (opzionale, default "/FA/PAX_%d.DCF") La sintassi per questi parametri è "-F filename", dove filename è il nome del file parametri completo di percorso.
comment	Stringa contenente un commento da inserire nel file DCF. (opzionale, default "") La sintassi per questi parametri è "-C comments" dove comments sono commenti di massimo 240 caratteri senza LF (Line Feed).
Risultato	Significato dei valori restituiti:

Valore	Descrizione
< 0	Codice NACK non convertito. (segnalare a Robox)
0	Operazione ok.
1	Argomento mancante. (usare mreport -a e/o la variabile predichiarata sts_co_par_download utilizzata per il reso)
2	Argomento illegale. (usare mreport -a e/o sts_co_par_download)
3	Uno dei file specificati non esiste. (usare mreport -a)

© 2025 Robox SpA

Sintassi	**i32 co_par_upload (string nameCmdFile, i32 nWs, i32 flagBrk [, string nameParFile="/FA/PAX_%d.DCF"] [, string comment=""])**

Valore	Descrizione
4	Uno dei file specificati è illegale. (usare mreport -a)
5	Operazione abortita. (usare mreport -a)
6	Memoria piena (RAM o CF). (usare mreport -a)
7	Impossibile rinominare il file. File già esistente. (usare mreport -a)
8	Errore in scrittura del file. (usare mreport -a)
9	Errore in lettura del file. (usare mreport -a)
10	Impossibile aggiungere ulteriori dati al file in ram durante la gestione. (usare mreport -a)
11	Errore su comando di lettura/scrittura oggetti su fieldbus. (usare mreport -a e/o sts_co_par_download)
12	Errore apertura file. (usare mreport -a)
13	Gestore istruzioni/comandi occupato.
14	Operazione ok ma presenti errori di oggetto o sub index non esistenti. Si verifica quando si disabilita la generazione del codice di ritorno 11 tramite sys_cfg. In

© 2025 Robox SpA

Sintassi	**i32 co_par_upload (string nameCmdFile, i32 nWs, i32 flagBrk [, string nameParFile="/FA/PAX_%d.DCF"] [, string comment=""])**
	Valore: Descrizione: caso di upload, la linea con il valore viene inserita come commento con la segnalazione dell'errore. (usare mreport -a)
Validità	Task.
Note	Il flag sys_cfg_2.24, per abilitare la memorizzazione dei soli oggetti il cui valore è diverso dal *DefaultValue*, è ritentivo.

Funzione co_rerr()

Funzione per leggere l'ultimo messaggio d'errore generato dalla ws indicata.

Sintassi	**i32 co_rerr (i32 nWs, stru_coerrmsg stru)**
nWs	ID stazione Robox (workstation logica 1-:-64). **NOTA**: Se si specifica 0 come ws allora si intende per tutte.
stru	Struttura stru_coerrmsg 836 in cui verranno inserite le informazioni.
Risultato	Rende: Valore: 0 — Descrizione: Nessun messaggio di errore. Valore: 1 — Descrizione: Messaggio di errore presente. Valore: -1 — Descrizione: Parametri illegali (numero di ws illegale).
Validità	Rule, Task.
Note	Rende le stesse informazioni del comando D_EMCY da shell RDE.

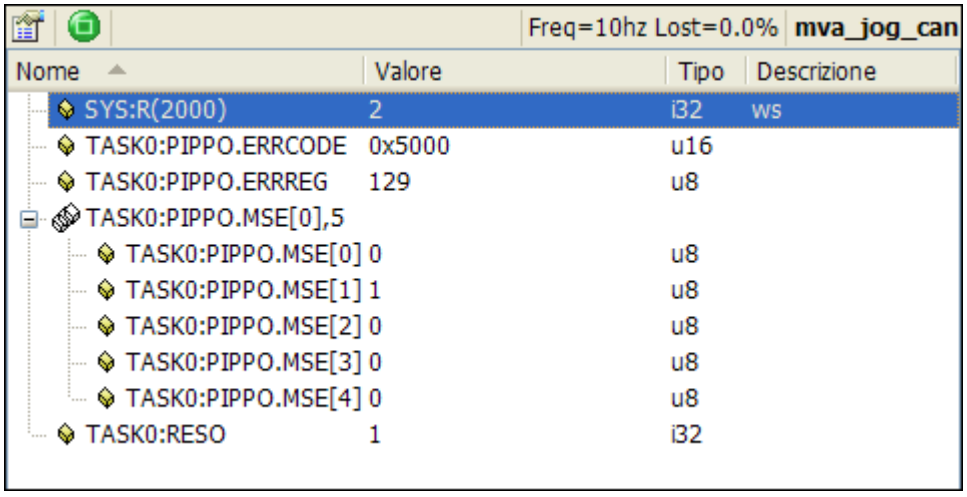
Esempi

© 2025 Robox SpA

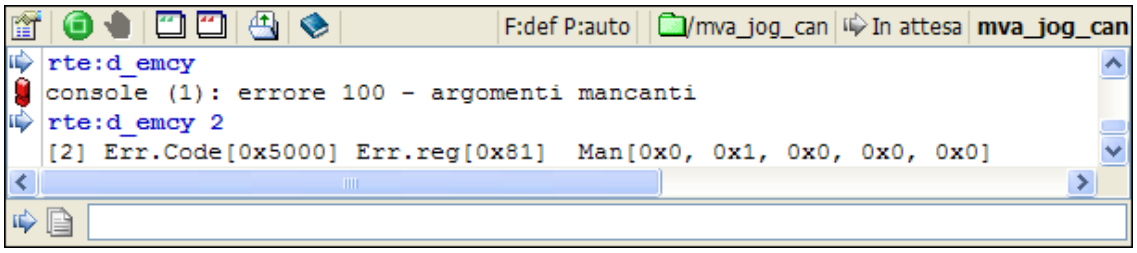
Esempio di utilizzo:

```
int result
stru_coerrmsg 836 struErr

result = co_rerr 255 (r(2000), struErr)   ; in r(2000) viene specificata
```

Da monitor RDE si vedrà:



Inviando il comando D_EMCY da shell RDE si ottengono le stesse informazioni:



Funzione co_rnmt()

Funzione che rende il modo operativo NMT della ws indicata.

Sintassi	**i32 co_rnmt (i32 nWs)**
nWs	ID stazione Robox (workstation logica 1-:-64). **NOTA**: Se si specifica 0 come ws allora si intende per tutte.
Risultato	Rende lo stato NMT (primi 8 bit):

Bit	Valore Hex	Significato
-	0x00000000	BOOT
2	0x00000004	STOP
0, 2	0x00000005	OPERATIONAL

© 2025 Robox SpA

Sintassi	**i32 co_rnmt (i32 nWs)**

Bit	Valore Hex	Significato
0, 1, 2, 3, 4, 5, 6	0x0000007f	PREOPERATIONAL

Inoltre vengono mappate le seguenti altre informazioni:

Bit	Valore Hex	Significato
8	0x00000100	=1 se stazione assente (vale sempre 0 se presenza forzata da cfg).
9	0x00000200	=1 se stazione assente realmente (anche se ha presenza forzata).
10	0x00000400	=1 se l'ultima richiesta di un PDO tramite Remote Transmission Request (RTR) è fallita.
11	0x00000800	=1 se stazione presente perché forzata da configuratore.
12	0x00001000	=1 se la stazione è in fase di configurazione.
13	0x00002000	=1 se la stazione ha abortito la fase di configurazione per cause varie.
14	0x00004000	=1 se la stazione ha inviato un RPDO più corto di quello configurato.

© 2025 Robox SpA

Sintassi	**i32 co_rnmt (i32 nWs)**

Bit	Valore Hex	Significato
15	0x00008000	=1 se la stazione ha inviato più di un PDO sincrono tra due messaggi di sincronismo.

Validità	Rule, Task.
Note	Rende le stesse informazioni del comando D_NMT da shell RDE.

Funzione co_robj()

ATTENZIONE: Questa funzione non può essere usata in un task se nello stesso sono presenti anche le funzioni co_send_sdo 266()/co_asw_sdo 239().

Questa funzione serve per eseguire la lettura di oggetti fieldbus. La Lettura può avvenire in modo passante o in modo bloccante (utilizzabile solo nei task 1-:-8).

Per modo passante si intende che la funzione deve essere continuamente chiamata e contemporaneamente testato lo stato attuale dell'operazione.

Per modo bloccante si intende che la funzione viene invocata e ritorna solo quando ha completato l'operazione di lettura.

La funzione rende un esito sullo stato dell'esecuzione.

Sintassi	**i32 co_robj (stru_co_obj stru)**
stru	Nome della struttura di tipo stru_co_obj 832.
Risultato	Variabile di tipo i32 contenente il risultato. Vedi gestione passante e/o gestione bloccante.
Validità	Task, Rule (solo passante).
Note	

Gestione passante

- Imporre ***stru*.phase = 1** per inizializzare la lettura.
- Il comando di lettura viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.
  La funzione rende 0.
- Attesa della risposta.
  La funzione continua a rendere 0.

© 2025 Robox SpA

- Ricezione risposta con esito **positivo**. Il valore letto viene scritto in *stru*.data. RTE cambia *stru*.phase = 4.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -1, parametri illegali (allarme 9900)
  - -2, gestore comunicazione occupato
  - -3, crash gestore comunicazione (riavvio)
  - -4, gestione bloccante non permessa nel task corrente (*stru*.phase = 1) (rule, ladder). Comando non accettato
  - -5, comando a un dispositivo non presente (*stru*.rid)
  - -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata
  - 2, comando di cui si è già ricevuto risposta positiva, valore disponibile in *data*
  - >2, codice di errore SDO, vedi tabella sotto

Gestione bloccante

- Imporre ***stru*.phase = -1** per inizializzare la lettura.
- Il comando di lettura viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.
- Attesa della risposta. Se la risposta non arriva entro un massimo di 3 secondi la funzione esce con un errore (timeout).
- Ricezione risposta con esito **positivo**. Il valore letto viene scritto in *stru*.data. RTE cambia *stru*.phase = 4.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -1, parametri illegali (allarme 9900)
  - -2, gestore comunicazione occupato
  - -3, crash gestore comunicazione (riavvio)
  - -4, gestione bloccante non permessa nel task corrente (*stru*.phase = 1) (rule, ladder). Comando non accettato
  - -5, comando a un dispositivo non presente (*stru*.rid)
  - -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata
  - 2, comando di cui si è già ricevuto risposta positiva, valore disponibile in *stru*.data
  - >2, codice di errore SDO, vedi tabella sotto

Codici di errore SDO SoE/CoE

SDO error code	Descrizione
0x0503 0000	Toggle bit non alternato.

© 2025 Robox SpA

SDO error code	Descrizione
0x0504 0000	Timeout del protocollo SDO.
0x0504 0001	Identificatore del comando client/server non valido o sconosciuto.
0x0504 0002	Dimensione del blocco non valida (solo gestione bloccante).
0x0504 0003	Numero di sequenza non valido (solo gestione bloccante).
0x0504 0004	Errore CRC (solo gestione bloccante).
0x0504 0005	Eccesso di memoria.
0x0601 0000	Accesso non supportato a un oggetto.
0x0601 0001	Tentativo di leggere un oggetto a sola scrittura.
0x0601 0002	Tentativo di scrivere un oggetto a sola lettura.
0x0602 0000	L'oggetto non esiste nel dizionario oggetti.
0x0604 0041	Impossibile mappare l'oggetto sul PDO.
0x0604 0042	Il numero e la lunghezza degli oggetti da mappare supererebbero la lunghezza del PDO.
0x0604 0043	Incompatibilità dei parametri.
0x0604 0047	Incompatibilità interna al dispositivo.
0x0606 0000	Accesso non riuscito a causa di un errore hardware.
0x0607 0010	Il tipo di dati non corrisponde, la lunghezza del parametro di servizio non corrisponde.
0x0607 0012	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo alto.
0x0607 0013	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo basso.
0x0609 0011	Il sottoindice non esiste.
0x0609 0030	Superamento del campo di valori del parametro (solo per accesso in scrittura).
0x0609 0031	Valore del parametro scritto troppo alto.
0x0609 0032	Valore del parametro scritto troppo basso.
0x0609 0036	Il valore massimo è inferiore al valore minimo.

© 2025 Robox SpA

SDO error code	Descrizione
0x0800 0000	Errore generale.
0x0800 0020	I dati non possono essere trasferiti o archiviati nell'applicazione.
0x0800 0021	I dati non possono essere trasferiti o archiviati nell'applicazione a causa del controllo locale.
0x0800 0022	I dati non possono essere trasferiti o archiviati nell'applicazione a causa dello stato attuale del dispositivo.
0x0800 0023	La generazione dinamica del dizionario oggetti non riesce o non è presente alcun dizionario oggetti (ad esempio, il dizionario oggetti viene generato da un file e la generazione non riesce a causa di un errore di file).

Esempi

Esempio di utilizzo:

```
; SAMPLE CODE
; Global Storage definition
;--------------------------

real CpuDrive_temp[32+1]
real Heater_temp[32+1]
real ControlBoard_temp[32+1]
real updateTime[32+1]
; storage used only by ReadImdTempPT
int whatTemp[32+1]
stru_co_obj sobj[32+1]
lit PER_UPD_TEMP 1.0
..
..
..
; Before the main loop
call InitReadImdTemp()
...
...
; inside the main loop
if(IWantPassThroughReadTempRoutine) ; test a flag to check if i have t
                                    ; or blocking routines
    result = ReadImdTempPT(k) ; k is the RID of the IMD that I want to
else
    result = ReadImdTempBL(k) ; k is the RID of the IMD that I want to
endif
...
```

© 2025 Robox SpA

```
...
...

;<0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0>
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
FUNCTION InitReadImdTemp()
    real tAct
    int k
    tAct = tfb
    ; initialize the tome to start reading after 5 seconds
    ; in sequence way
    for(k=1, k <=32 , INC(k))
        updateTime[k] = tAct + PER_UPD_TEMP + ((k-1)*PER_UPD_TEMP/32.0
        whatTemp[k] = 0
    end_for
ENDFUN

;<0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0>
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
FUNCTION int ReadImdTempPT(int rid)
    ; routine 'PASS_THROUGH' version
    ; function to update the IMD temperature value each PER_UPD_TEMP s
    ; the information will be set in global storage.
    ; the function want the RID of the IMD
    ; give back:
    ; <0 error
    ; >2 error (SDO diagnostic)
    ; 0 no operation
    ; 1 operation in progress
    ; 2 all temperature updated
    int ero
    ; check RID value
    if(not range(rid, 1, 32))
        return -1 ; return error if illegal value
    endif
    ; check if time to update
    if(updateTime[rid] <= tfb)
        inc(updateTime[rid], PER_UPD_TEMP)
        if(whatTemp[rid] = 0)
            whatTemp[rid] = 1
        endif
    endif
    ; check what we have to do
    select(whatTemp[rid])
        case 0
            return 0 ; nothing to do
        case 1
            sObj[rid].rid = rid
            sObj[rid].objCode = 0x2101 ; Cpu drive Temp
```

© 2025 Robox SpA

```
            sObj[rid].subIndex = 2
            sObj[rid].type = 5 ; float data
            sObj[rid].phase = 1 ; set PASS_THROUGH modality
            whatTemp[rid] = 2 ; next step
            ; break intentionally omitted
        case 2
            ero = co_robj 258(sobj[rid])
            select(ero)
                default
                    whatTemp[rid] = 0 ; go back
                    return ero ; return error code
                case 0
                    return 1 ; we have to wait
                case -2 ; busy, restart again next loop
                    sObj[rid].phase = 1 ; set PASS_THROUGH modality fo
                    return 1
                case 1 ; we got the answer and we save it
                    CpuDrive_temp[rid] = sObj[rid].data
                    break
            end_select
            ; prepare the read next temperature
            sObj[rid].objCode = 0x2101 ; Heater Temp
            sObj[rid].subIndex = 1
            sObj[rid].type = 5 ; float data
            sObj[rid].phase = 1 ; set PASS_THROUGH modality
            whatTemp[rid] = 3 ; next step
            ; break intentionally omitted
        case 3
            ero = co_robj(sobj[rid])
            select(ero)
                default
                    whatTemp[rid] = 0 ; go back
                    return ero ; return error code
                case 0
                    return 1 ; we have to wait
                case -2 ; busy, restart again next loop
                    sObj[rid].phase = 1 ; set PASS_THROUGH modality fo
                    return 1
                case 1 ; we got the answer and we save it
                    Heater_temp[rid] = sObj[rid].data
                    break
            end_select
            sObj[rid].objCode = 0x2101 ; Control Board Temp
            sObj[rid].subIndex = 0xC
            sObj[rid].type = 5 ; float data
            sObj[rid].phase = 1 ; set PASS_THROUGH modality
            whatTemp[rid] = 4 ; next step
            ; break intentionally omitted
        case 4
```

© 2025 Robox SpA

```
            ero = co_robj(sobj[rid])
            select(ero)
                default
                    whatTemp[rid] = 0 ; go back
                    return ero ; return error code
                case 0
                    return 1 ; we have to wait
                case -2 ; busy, restart again next loop
                    sObj[rid].phase = 1 ; set PASS_THROUGH modality fo
                    return 1
                case 1 ; we got the answer and we save it
                    ControlBoard_temp[rid] = sObj[rid].data
                    break
            end_select
            whatTemp[rid] = 0 ; end of the job ....
            break
    end_select
    return 2 ; all temperature updated
ENDFUN

;<0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0>
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
FUNCTION int ReadImdTempBL(int rid)
    ; routine 'BLOCKING' version
    ; function to update the IMD temperature value each PER_UPD_TEMP s
    ; the information will be set in global storage.
    ; the function want the RID of the IMD
    ; give back:
    ; <0 error
    ; >2 error (SDO diagnostic)
    ; 0 no operation
    ; 2 all temperature updated
    int ero, timeToRead
    stru_co_obj sobj
    ; check RID value
    if(not range(rid, 1, 32))
        return -1 ; return error if illegal value
    endif
    timeToRead = FALSE
    ; check if time to update
    if(updateTime[rid] <= tfb)
        inc(updateTime[rid], PER_UPD_TEMP)
        timeToRead = TRUE
    endif
    ; check what we have to do
    if(not timeToRead)
        return 0 ; nothing to do
    endif
    ; we can read
```

© 2025 Robox SpA

```
    sObj.rid = rid
    sObj.objCode = 0x2101 ; Cpu drive Temp
    sObj.subIndex = 2
    sObj.type = 5 ; float data
    sObj.phase = -1 ; set Blocking modality
    ero = co_robj(sobj)
    if(ero <> 1)
        return ero
    endif
    CpuDrive_temp[rid] = sObj.data
    ; prepare the read next temperature
    sObj.subIndex = 1 ; Heater Temp
    sObj.phase = -1 ; set PASS_THROUGH modality
    ero = co_robj(sobj)
    if(ero <> 1)
        return ero
    endif
    Heater_temp[rid] = sObj.data
    ; prepare the read next temperature
    sObj.subIndex = 0xC ; Control Board Temp
    sObj.phase = -1 ; set PASS_THROUGH modality
    ero = co_robj(sobj)
    if(ero <> 1)
        return ero
    endif
    ControlBoard_temp[rid] = sObj.data
    return 2 ; all temperatures updated
ENDFUN
```

Funzione co_rpdo()

Questa funzione serve per leggere i dati nel buffer (PDO - Process Data Object) ricevuti dalla workstation indicata. Serve inoltre a inviare la richiesta di aggiornamento dei pdo (RTR).

Sintassi	**i32 co_rpdo (i32 nWs, i32 nPdo, i32 nEl, real where)**
nWs	ID stazione Robox (workstation logica 1-:-64). **NOTA**: Se si specifica 0 come ws allora si intende per tutte.
nPdo	Numero del PDO in ricezione dall'azionamento (1-:-4).
nEl	Numero dell'elemento in questione che si vuole leggere: • 0, comando di richiesta refresh dei dati rPDO (RTR per PDO asincrono)

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 co_rpdo (i32 nWs, i32 nPdo, i32 nEl, real where)</td></tr>
<tr><td></td><td>• 1-:-8, numero dell'elemento logico da leggere

NOTA: Se nEl vale 0 il valore non viene reso.</td></tr>
<tr><td>where</td><td>Parametro in cui si scarica il valore letto da PDO. La conversione dal giusto formato (definito nel file di configurazione) a real è eseguita dal sw di base.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Condizione</th><th>Valore</th><th>Descrizione</th></tr>
<tr><td>-</td><td>0</td><td>Parametri illegali.</td></tr>
<tr><td>nEl = 1</td><td>1</td><td>Dato letto per la prima volta.

NOTA: Informazione significativa solo se questa istruzione viene eseguita nelle rule a tutte le scansioni.</td></tr>
<tr><td>nEl = 1</td><td>2</td><td>Dato già letto.

NOTA: Informazione significativa solo se questa istruzione viene eseguita nelle rule a tutte le scansioni.</td></tr>
<tr><td>nEl <> 1</td><td>3</td><td>Dato valido.</td></tr>
</table></td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td></td></tr>
</table>

Funzione co_send_sdo()

ATTENZIONE: Questa funzione non può essere usata in un task se nello stesso sono presenti anche le funzioni co_robj 258()/co_wobj 273()/coe_robj 279()/coe_wobj 284().

Questa funzione serve per inviare un messaggio SDO (Service Data Object) verso la workstation specificata al fine di scrivere/leggere un oggetto del dizionario oggetti.

© 2025 Robox SpA

- In caso di scrittura -> co_send_sdo(*nWs*, *nObj*, *nSub*): viene poi utilizzata la funzione co_asw_sdo() per controllare se la scrittura è andata a buon fine
- In caso di lettura -> co_send_sdo(*nWs*, *nObj*, *nSub*, *type*, *data*): viene poi utilizzata la funzione co_asw_sdo() per controllare se la richiesta di lettura è andata a buon fine e per scaricare i dati

Sintassi	**i32 co_send_sdo (i32 nWs, i32 nObj, i32 nSub)** **i32 co_send_sdo (i32 nWs, i32 nObj, i32 nSub, i32 type, real data)**
nWs	ID stazione Robox (workstation logica 1-:-64).
nObj	Codice oggetto del dizionario (vedi dictionary - file eds).
nSub	Codice sub index (vedi dictionary - file eds). In caso di EtherCAT (SoE) vedi sotto.
type	Permette di selezionare come trasformare i dati da scrivere contenuti in *data* nel tipo richiesto dal dispositivo collegato.

Valore	Tipo
1	Signed/unsigned 8 bit.
2	Signed/unsigned 16 bit.
3	Signed/unsigned 24 bit.
4	Signed/unsigned 32 bit.
5	Real 32 bit.

data	Valore da trasformare (da formato real a formato FIX di dimensione specificata in *type*).
Risultato	Rende:

Valore	Descrizione
>= 0	Numero di handle del messaggio. (da utilizzare poi come parametro entrante nell'istruzione co_asw_sdo()).
-1	Parametri illegali.
-2	Gestore messaggi RTE occupato.

Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**i32 co_send_sdo (i32 nWs, i32 nObj, i32 nSub)** **i32 co_send_sdo (i32 nWs, i32 nObj, i32 nSub, i32 type, real data)**
Note	Evitare, se l'istruzione è utilizzata all'interno di una funzione, di dichiarare handle locale alla funzione stessa. Da RTE 34.13.5 sono disponibili le funzioni co_robj 258() e co_wobj 273().

Ethercat SoE parametro sub index

Bit	Valore Hex	Significato
0	0x00000001	Codificati a valore: • 0 se monodispositivo • indice di quello interessato in caso di multidispositivo
1	0x00000002	
2	0x00000004	
3	0x00000008	Riservato (= 0).
4	0x00000010	Riservato (= 0).
Su operazione di scrittura:		
5	0x00000020	In 1 richiesta di abort del comando associato ad IDN specificato. **NOTA**: viene inviato comunque codice di fine comando.
6	0x00000040	In 1 abilitazione gestione notifica stato del comando associato ad IDN specificato (se associato al codice di start) (valore 3).
7	0x00000080	In 1 abilitazione gestione automatica dei comandi (con invio di start comando e su notifica di fine esecuzione comando invio codice di fine). Se il comando dura più del tempo di timeout di qualche secondo, la routine di attesa risposta uscirà di timeout. Il comando SoE rimarrà comunque attivo.
Su operazione di lettura:		
5	0x00000020	Codificati a valore:

© 2025 Robox SpA

Bit	Valore Hex	Significato
6	0x00000040	• -2 comando abortito o mai startato • 0 comando in esecuzione • 3 comando eseguito con successo • 15 errore durante l'esecuzione del comando
7	0x00000080	

Esempi

Esempio di utilizzo:

```
; if there is a reading command
if(rise(command_read))
    command_read = false
    faseRead = SEND
endif
select(faseRead)
    case IDLE
    break
    case SEND
        handle = co_send_sdo(rNws, rObj, rSub)
        if(handle < 0)
            faseRead = ERROR
            break
        endif
        faseRead = GET_DATA
    break
    case GET_DATA
       result = co_asw_sdo(rNws, handle, rType, Data)
        ; waiting answer
        _if(result =0)
            break
        ; test answer
        if(result = 1)
            faseRead = DONE
        else
            faseRead = ERROR
        endif
    break
    case DONE
        ; insert in the MREPORT diagnosys from SHELL
        str_format(srInfo, "SDO Read ws:%d, obj: %d, sub:%d, returne
        inf_report(0x1f, srInfo)
        faseRead = IDLE
    break
    case ERROR
        ; insert in the MREPORT diagnosys from SHELL
```

© 2025 Robox SpA

```
        str_format(srInfo, "Error SDO read value:%d", result)
        inf_report(0x1f,srInfo)
        faseRead = IDLE
    break
end_select

; SCRITTURA
INT handle, reso
; scrittura oggetto 2000 sub 0x92 della workstation 1
; (es. su HID corrisponde al pr 145 -Kp anello di posizione-) al valor
handle = co_send_sdo(1, 0x2000, 0x92, 4, 0x64)

reso = co_asw_sdo(1, handle)
_if(reso = 1)
    sr(1) = "scrittura avvenuta con successo"
_if(reso = 0)
    sr(1) = "Messaggio in attesa di risposta"
_if(reso < 0)
    sr(1) = "errori"

; LETTURA
INT handle, reso
REAL data
; lettura oggetto 2000 sub 0x92 della workstation 1
handle = co_send_sdo(1, 0x2000, 0x92)
_if(co_asw_sdo(1, handle, 2, data) = 1)
    ; test di arrivo risposta e scarico del dato
    ; ...
```

Funzione co_statusword()

Funzione che rende il contenuto della status word della workstation indicata.

NOTA: Da RTE 34.03.06 sono disponibili le keyword fb_stsw e fb_ctrlw.

Sintassi	**i32 co_statusword (i32 nWs [, i32 flgCw=FALSE])**
nWs	ID stazione Robox (workstation logica 1-:-64). **NOTA**: Se si specifica 0 come ws allora si intende per tutte.
flgCw	Parametro per abilitare anche la lettura anche della control word. (opzionale, default FALSE).
Risultato	Rende: Condizione | Valore | Descrizione *flgCw = 0* | Valore della status word. | Valore reso in w0[671] (parte bassa, prima word).

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 co_statusword (i32 nWs [, i32 flgCw=FALSE])</td></tr>
<tr><td></td><td>
<table>
<tr><th>Condizione</th><th>Valore</th><th>Descrizione</th></tr>
<tr><td></td><td>0</td><td>Valore reso in w1 (parte alta, seconda word).</td></tr>
<tr><td rowspan="2">flgCw <> 0</td><td>Valore della status word.</td><td>Valore reso in w0.</td></tr>
<tr><td>Valore della control word.</td><td>Valore reso in w1.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td></td></tr>
</table>

Status word bit

Bit	Valore esadecimale	Descrizione
0	0x00000001	Ready to switch ON.
1	0x00000002	Switched ON.
2	0x00000004	Operation enabled.
3	0x00000008	Fault.
4	0x00000010	Voltage enabled.
5	0x00000020	Quick stop.
6	0x00000040	Switch on disabled.
7	0x00000080	Warning.
8	0x00000100	Manufacturer specific.
9	0x00000200	Remote.
10	0x00000400	Target reached.
11	0x00000800	Internal limit active.
12	0x00001000	Operation mode specific.
13	0x00002000	Operation mode specific.
14	0x00004000	Manufacturer specific.
15	0x00008000	Manufacturer specific.

La figura è stata estrapolata da CiA Draft Standard Proposal 402.

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
int stws[33] ; stsws[0] non viene usato
for(i= 1 , i<=32 , i++)
    stws[i] = co_statusword 270 (i)
end_for
```

Esempio di utilizzo:

```
reso = co_statusword(ws,true)
r(1000) = w0 671 (reso)    ; status word
r(1001) = w1 671 (reso)    ; control word
```

Funzione co_sync()

ATTENZIONE: Obsoleta.

Questa funzione serve per abilitare/disabilitare la trasmissione del messaggio di SYNC su tutte le interfacce CANopen DS402 (SW CODMnnvv.xxx) presenti nel sistema.

Sintassi	**i32 co_sync (i32 flag)**
flag	Parametro per abilitare/disabilitare l'invio del messaggio di SYNC: • <> 0, abilita invio (default all'accensione per il gestore RTE) • 0, disabilita invio
Risultato	Rende: Valore: 0 — CANopen non presente. Valore: 1 — Abilitazione riuscita.
Validità	Rule, Task.
Note	L'abilitazione viene gestita da RTE.

Funzione co_wnmt()

Funzione che impone il modo operativo NMT della workstation indicata.

Sintassi	**i32 co_wnmt(i32 nWs, i32 opeMode)**
nWs	ID stazione Robox (workstation logica 1-:-64). **NOTA**: Se si specifica 0 come ws allora si intende per tutte.
opeMode	Modo operativo imponibile:

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 co_wnmt(i32 nWs, i32 opeMode)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Code</th><th>Modo</th></tr>
<tr><td>1</td><td>0x01</td><td>ENTER OPERATIONAL</td></tr>
<tr><td>2</td><td>0x02</td><td>ENTER STOP</td></tr>
<tr><td>128</td><td>0x80</td><td>ENTER PREOPERATIONAL</td></tr>
<tr><td>129</td><td>0x81</td><td>RESET NODE</td></tr>
<tr><td>130</td><td>0x82</td><td>RESET COMMUNICATION</td></tr>
</table>
</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td><> 0</td><td>Scrittura avvenuta con successo.</td></tr>
<tr><td>0</td><td>Riscontrati parametri illegali.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td>L'azione si può ottenere anche da comando shell.</td></tr>
</table>

Funzione co_wobj()

ATTENZIONE: Questa funzione non può essere usata in un task se nello stesso sono presenti anche le funzioni co_send_sdo 266()/co_asw_sdo 239().

Questa funzione serve per eseguire la scrittura di oggetti fieldbus. La scrittura può avvenire in modo passante o in modo bloccante (il modo bloccante è utilizzabile solo nei task 1-:-8).

Per modo passante si intende che la funzione deve essere continuamente chiamata e questa rende lo stato attuale dell'operazione.

Per modo bloccante si intende che la funzione viene invocata e ritorna solo quando ha completato l'operazione di scrittura.

La funzione rende un esito sullo stato dell'esecuzione.

Sintassi	**i32 co_wobj (stru_co_obj stru)**
stru	Nome della struttura di tipo stru_co_obj 832.
Risultato	Variabile di tipo i32 contenente il risultato. Vedi gestione passante e/o gestione bloccante.

© 2025 Robox SpA

Sintassi	**i32 co_wobj (stru_co_obj stru)**
Validità	Rule, Task.
Note	

Gestione passante

- Imporre ***stru*.phase = 1** per inizializzare la scrittura.
- Il comando di scrittura viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.
  La funzione rende 0.
- Attesa della risposta.
  La funzione continua a rendere 0.
- Ricezione risposta con esito **positivo**. Il valore letto viene scritto in *stru*.data. RTE cambia *stru*.phase = 4.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -1, parametri illegali (allarme 9900)
  - -2, gestore comunicazione occupato
  - -3, crash gestore comunicazione (riavvio)
  - -5, comando a un dispositivo non presente (*stru*.rid)
  - -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata
  - 2, comando di cui si è già ricevuto risposta positiva, valore disponibile in *stru*.data
  - >2, codice di errore SDO, vedi tabella sotto

Gestione bloccante

- Imporre ***stru*.phase = -1** per eseguire la scrittura.
- Il comando di scrittura viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.
- Attesa della risposta. Se la risposta non arriva entro un massimo di 3 secondi la funzione esce con un errore (timeout, rende -2).
- Ricezione risposta con esito **positivo**. Il valore letto viene scritto in *stru*.data. RTE cambia *stru*.phase = 4.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -1, parametri illegali (allarme 9900)
  - -2, gestore comunicazione occupato
  - -3, crash gestore comunicazione (riavvio)
  - -4, gestione bloccante non permessa nel task corrente (*stru*.phase = 1) (rule, ladder). Comando non accettato
  - -5, comando a un dispositivo non presente (*stru*.rid)

© 2025 Robox SpA

- -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata
- 2, comando di cui si è già ricevuto risposta positiva, valore disponibile in *stru*.data
- >2, codice di errore SDO, vedi tabella sotto

Codici di errore SDO SoE/CoE

SDO error code	Descrizione
0x0503 0000	Toggle bit non alternato.
0x0504 0000	Timeout del protocollo SDO.
0x0504 0001	Identificatore del comando client/server non valido o sconosciuto.
0x0504 0002	Dimensione del blocco non valida (solo gestione bloccante).
0x0504 0003	Numero di sequenza non valido (solo gestione bloccante).
0x0504 0004	Errore CRC (solo gestione bloccante).
0x0504 0005	Eccesso di memoria.
0x0601 0000	Accesso non supportato a un oggetto.
0x0601 0001	Tentativo di leggere un oggetto a sola scrittura.
0x0601 0002	Tentativo di scrivere un oggetto a sola lettura.
0x0602 0000	L'oggetto non esiste nel dizionario oggetti.
0x0604 0041	Impossibile mappare l'oggetto sul PDO.
0x0604 0042	Il numero e la lunghezza degli oggetti da mappare supererebbero la lunghezza del PDO.
0x0604 0043	Incompatibilità dei parametri.
0x0604 0047	Incompatibilità interna al dispositivo.
0x0606 0000	Accesso non riuscito a causa di un errore hardware.
0x0607 0010	Il tipo di dati non corrisponde, la lunghezza del parametro di servizio non corrisponde.
0x0607 0012	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo alto.
0x0607 0013	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo basso.
0x0609 0011	Il sottoindice non esiste.

© 2025 Robox SpA

SDO error code	Descrizione
0x0609 0030	Superamento del campo di valori del parametro (solo per accesso in scrittura).
0x0609 0031	Valore del parametro scritto troppo alto.
0x0609 0032	Valore del parametro scritto troppo basso.
0x0609 0036	Il valore massimo è inferiore al valore minimo.
0x0800 0000	Errore generale.
0x0800 0020	I dati non possono essere trasferiti o archiviati nell'applicazione.
0x0800 0021	I dati non possono essere trasferiti o archiviati nell'applicazione a causa del controllo locale.
0x0800 0022	I dati non possono essere trasferiti o archiviati nell'applicazione a causa dello stato attuale del dispositivo.
0x0800 0023	La generazione dinamica del dizionario oggetti non riesce o non è presente alcun dizionario oggetti (ad esempio, il dizionario oggetti viene generato da un file e la generazione non riesce a causa di un errore di file).

Esempi

```
; SAMPLE CODE
; Global Storage definition
;---------------------------

stru_co_obj 832 swobj[32+1]
int resW
....
....
....
; version using pass-through routine
if(nm <> 0)
    do
        resW = SetImdTorqueMaxPT(2, nm)
        ...
        ...
    end_do_while(resW = 1)
    nm = 0
endif
....
....
....
```

© 2025 Robox SpA

```
; version using blocking routine
if(nm <> 0)
    resW = SetImdTorqueMaxBL (2, nm)
    nm = 0
endif
....
....
....

;<0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0>
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
FUNCTION int SetImdTorqueMaxPT(int rid, real nm)

    ; routine 'PASS_THROUGH' version
    ; function to update the MAX_TORQUE on Imd.
    ; the function want the RID of the IMD, and the value in NM to set
    ; it gives back:
    ; <0 error
    ; >2 error (SDO diagnostic)
    ; 0 no operation
    ; 1 operation in progress
    ; 2 torque has been written
    int ero
    ; check RID value
    if(not range 557(rid, 1, 32))
        return -1 ; return error if illegal value
    endif
    if((swObj[rid].phase = 0 ) or (swobj[rid].phase = 4) )
        swobj[rid].rid = rid
        swObj[rid].objCode = 0x2050 ; Absolute Torque Limit
        swObj[rid].subIndex = 9
        swObj[rid].type = 5 ; float data
        swObj[rid].data = nm ; value to write
        swObj[rid].phase = 1 ; set PASS_THROUGH modality
    endif
    ; check what we have to do
    ero = co_wobj 273(swobj[rid])
    select(ero)
        default
            whatTemp[rid] = 0 ; go back
            return ero ; return error code
        case 0
            return 1 ; we have to wait
        case -2 ; busy, restart again next loop
            sObj[rid].phase = 1 ; set PASS_THROUGH modality for the ne
            return 1
        case 1 ; we got the answer that the parameters has been writte
            break
    end_select
    return 2 ; the parameters has been written
```

© 2025 Robox SpA

```
endfun

;<0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0><0>
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
FUNCTION int SetImdTorqueMaxBl(int rid, real nm)

    ; routine 'BLOCKING' version
    ; function to update the MAX_TORQUE on Imd.
    ; the function want the RID of the IMD, and the value in NM to set
    ; it gives back:
    ; <0 error
    ; >2 error (SDO diagnostic)
    ; 0 no operation
    ; 1 operation in progress
    ; 2 torque has been written
    int ero
    stru_co_obj swobj
    swobj.rid = rid
    swObj.objCode = 0x2050 ; Absolute Torque Limit
    swObj.subIndex = 9
    swObj.type = 5 ; float data
    swObj.data = nm ; value to write
    swObj.phase = -1 ; set BLOCKING modality
    ; check what we have to do
    ero = co_wobj(swobj)
    if(ero <> 1)
        return ero ; error
    endif
    return 2 ; the parameters has been written
ENDFUN
```

Funzione co_wpdo()

Questa funzione serve per scrivere i dati nel buffer (PDO - Process Data Object) di trasmissione verso la workstation indicata.

Sintassi	**i32 co_wpdo (i32 nWs, i32 nPdo, i32 nEl, real value)**
nWs	ID stazione Robox (workstation logica 1-:-64). **NOTA**: Se si specifica 0 come ws allora si intende per tutte.
nPdo	Numero del PDO in trasmissione verso l'azionamento (1-:-4).
nEl	Numero dell'elemento in questione che si vuole scrivere: • 0, comando di trasmissione (per PDO asincrono)

© 2025 Robox SpA

Sintassi	**i32 co_wpdo (i32 nWs, i32 nPdo, i32 nEl, real value)**
	• 1-:-8, numero dell'elemento logico all'interno del PDO **NOTA**: Se *nEl* vale 0 il valore non conta.
value	Valore da immettere nel PDO. La conversione nel giusto formato (definito nel file di configurazione) è eseguita dal sw di base.
Risultato	Rende: Valore / Descrizione <> 0 / Scrittura avvenuta con successo. 0 / Riscontrati parametri illegali.
Validità	Rule, Task.
Note	

Funzione coe_robj()

ATTENZIONE: Questa funzione non può essere usata in un task se nello stesso sono presenti anche le funzioni co_send_sdo 266()/co_asw_sdo 239().

Questa funzione serve per eseguire la lettura di oggetti fieldbus CoE (viene letto l'oggetto definito da '*stru*.objCode + *stru*.subIndex' sul dispositivo identificato da *stru*.rid).

Può leggere più tipi di dati rispetto alla funzione analoga co_robj 258().

La lettura può avvenire in modo passante o in modo bloccante (utilizzabile solo nei task 1-:-8).

Per modo passante si intende che la funzione deve essere continuamente chiamata e contemporaneamente testato lo stato attuale dell'operazione.

Per modo bloccante si intende che la funzione viene invocata e ritorna solo quando ha completato l'operazione di lettura.

La funzione rende un esito sullo stato dell'esecuzione.

Sintassi	**i32 co_robj (stru_coe_obj stru, destination)**
stru	Nome della struttura di tipo stru_coe_obj 834.
destination	Contiene le informazioni ricevute. *destination* deve essere un oggetto compatibile con i dati che si stanno leggendo (se si legge una VISIBLE STRING, *destination* sarà di tipo string, se si legge un OCTET_STRING sarà un array di byte o una struttura).

© 2025 Robox SpA

Sintassi	**i32 co_robj (stru_coe_obj stru, destination)**
Risultato	Variabile di tipo i32 contenente il risultato. Vedi gestione passante e/o gestione bloccante.
Validità	Rule, Task.
Note	

Gestione passante

- Imporre ***stru*.phase = 1** per inizializzare la lettura.
- Il comando di lettura viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.
  La funzione rende 0.
- Attesa della risposta.
  La funzione continua a rendere 0.
- Ricezione risposta con esito **positivo**. Il valore letto viene scritto in *destination*. RTE cambia *stru*.phase = 4.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -2, gestore comunicazione occupato
  - -3, crash gestore comunicazione (riavvio)
  - -5, comando a un dispositivo non presente (*stru*.rid)
  - -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata
  - -12, valore illegale di objcode (allarme 9900 arg 12)
  - -13, valore illegale di subindex (allarme 9900 arg 13)
  - -14, valore illegale di datatype (allarme 9900 arg 4/arg 14)
  - -15, valore illegale di datalen (allarme 9900 arg 5/arg 15)
  - -16, valore illegale di phase (allarme 9900 arg 6)
  - -21, destinazione non specificata (allarme 9900 arg 16)
  - -22, destinazione cambiata (allarme 9900 arg 32)
  - 2, comando di cui si è già ricevuto risposta positiva, valore disponibile in *destination*
  - >2, codice di errore SDO, vedi tabella sotto

Gestione bloccante

- Imporre ***stru*.phase = -1** per eseguire la lettura.
- Il comando di lettura viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.
- Attesa della risposta. Se la risposta non arriva entro un massimo di 3 secondi la funzione esce con un errore (timeout, rende -2).

© 2025 Robox SpA

- Ricezione risposta con esito **positivo**. Il valore letto viene scritto in *destination*. RTE cambia *stru*.phase = 4.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -2, gestore comunicazione occupato
  - -3, crash gestore comunicazione (riavvio)
  - -4, gestione bloccante non permessa nel task corrente (*stru*.phase = 1) (rule, ladder). Comando non accettato
  - -5, comando a un dispositivo non presente (*stru*.rid)
  - -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata
  - -12, valore illegale di objcode (allarme 9900 arg 12)
  - -13, valore illegale di subindex (allarme 9900 arg 13)
  - -14, valore illegale di datatype (allarme 9900 arg 4/arg 14)
  - -15, valore illegale di datalen (allarme 9900 arg 5/arg 15)
  - -16, valore illegale di phase (allarme 9900 arg 6)
  - -21, destinazione non specificata (allarme 9900 arg 16)
  - -22, destinazione cambiata (allarme 9900 arg 32)
  - 2, comando di cui si è già ricevuto risposta positiva, valore disponibile in *destination*
  - >2, codice di errore SDO, vedi tabella sotto

Codici di errore SDO SoE/CoE

SDO error code	Descrizione
0x0503 0000	Toggle bit non alternato.
0x0504 0000	Timeout del protocollo SDO.
0x0504 0001	Identificatore del comando client/server non valido o sconosciuto.
0x0504 0002	Dimensione del blocco non valida (solo gestione bloccante).
0x0504 0003	Numero di sequenza non valido (solo gestione bloccante).
0x0504 0004	Errore CRC (solo gestione bloccante).
0x0504 0005	Eccesso di memoria.
0x0601 0000	Accesso non supportato a un oggetto.
0x0601 0001	Tentativo di leggere un oggetto a sola scrittura.
0x0601 0002	Tentativo di scrivere un oggetto a sola lettura.

© 2025 Robox SpA

SDO error code	Descrizione
0x0602 0000	L'oggetto non esiste nel dizionario oggetti.
0x0604 0041	Impossibile mappare l'oggetto sul PDO.
0x0604 0042	Il numero e la lunghezza degli oggetti da mappare supererebbero la lunghezza del PDO.
0x0604 0043	Incompatibilità dei parametri.
0x0604 0047	Incompatibilità interna al dispositivo.
0x0606 0000	Accesso non riuscito a causa di un errore hardware.
0x0607 0010	Il tipo di dati non corrisponde, la lunghezza del parametro di servizio non corrisponde.
0x0607 0012	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo alto.
0x0607 0013	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo basso.
0x0609 0011	Il sottoindice non esiste.
0x0609 0030	Superamento del campo di valori del parametro (solo per accesso in scrittura).
0x0609 0031	Valore del parametro scritto troppo alto.
0x0609 0032	Valore del parametro scritto troppo basso.
0x0609 0036	Il valore massimo è inferiore al valore minimo.
0x0800 0000	Errore generale.
0x0800 0020	I dati non possono essere trasferiti o archiviati nell'applicazione.
0x0800 0021	I dati non possono essere trasferiti o archiviati nell'applicazione a causa del controllo locale.
0x0800 0022	I dati non possono essere trasferiti o archiviati nell'applicazione a causa dello stato attuale del dispositivo.
0x0800 0023	La generazione dinamica del dizionario oggetti non riesce o non è presente alcun dizionario oggetti (ad esempio, il dizionario oggetti viene generato da un file e la generazione non riesce a causa di un errore di file).

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
; SAMPLE CODE
; =======
; in task
; =======

stru_coe_obj 834 rCoe
int          esitoFB
string       rCoeS
u8           rCoeO[256]
...
...
    rCoe.rid      = 2       ; axioline
    rCoe.objCode  = 0x9002  ; module specific information
    rCoe.subIndex = 1       ; product name
    rCoe.datatype = 9       ; visible string
    rCoe.datalen  = 128     ; dimension max string
    rCoe.phase    = -1
    esitoFB = coe_robj 279 (rCoe, rCoeS)
    if(esitoFB < 0)
        ; errore
        ...
    else
       ; now the string can be used
        ....
    endif
    ...
    ...
    ...
    rCoe.rid      = 2       ; axioline
    rCoe.objCode  = 0x9002  ; module specific information
    rCoe.subIndex = 0xb     ; Device Type
    rCoe.datatype = 10      ; octet
    rCoe.datalen  = 256
    rCoe.phase    = -1
    esitoFB = coe_robj(rCoe, rCoeO)
    if(esitoFB < 0)
        ; errore
        ...
    else
       ; now the data of the array can be used
        ....
    endif
```

© 2025 Robox SpA

Funzione coe_wobj()

ATTENZIONE: Questa funzione non può essere usata in un task se nello stesso sono presenti anche le funzioni co_send_sdo266()/co_asw_sdo239().

Questa funzione serve per eseguire la scrittura di oggetti fieldbus CoE (vengono scritte le informazioni presenti in *source*).

Può scrivere più tipi di dati rispetto alla funzione analoga co_wobj273().

La scrittura può avvenire in modo passante o in modo bloccante (utilizzabile solo nei task 1-:-8).

Per modo passante si intende che la funzione deve essere continuamente chiamata e contemporaneamente testato lo stato attuale dell'operazione.

Per modo bloccante si intende che la funzione viene invocata e ritorna solo quando ha completato l'operazione di scrittura.

La funzione rende un esito sullo stato dell'esecuzione.

Sintassi	**i32 coe_wobj (stru_coe_obj stru, source)**
stru	Nome della struttura di tipo stru_coe_obj834.
source	I dati contenuti in *source* verranno assunti in base al tipo di dato specificato (se si scrive una VISIBLE STRING, *source* sarà di tipo string, se si scrive un OCTET_STRING, *source* sarà un array di byte o una struttura).
Risultato	Variabile di tipo i32 contenente il risultato. Vedi gestione passante e/o gestione bloccante.
Validità	Rule, Task.
Note	

Gestione passante

- Imporre ***stru*.phase = 1** per inizializzare la scrittura.
- Il comando di scrittura del valore *source* viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.
  La funzione rende 0.
- Attesa della risposta.
  La funzione continua a rendere 0.
- Ricezione risposta con esito **positivo**. Il valore letto viene scritto in *source*. RTE cambia *stru*.phase = 4.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -2, gestore comunicazione occupato
  - -3, crash gestore comunicazione (riavvio)
  - -5, comando a un dispositivo non presente (*stru*.rid)
  - -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata

© 2025 Robox SpA

- -12, valore illegale di objcode (allarme 9900 arg 12)
- -13, valore illegale di subindex (allarme 9900 arg 13)
- -14, valore illegale di datatype (allarme 9900 arg 4/arg 14)
- -15, valore illegale di datalen (allarme 9900 arg 5/arg 15)
- -16, valore illegale di phase (allarme 9900 arg 6)
- -21, *source* non specificato (NULL) (allarme 9900 arg 16)
- -22, *source* cambiato (esecuzione a fasi) (allarme 9900 arg 32)
- 2, comando di cui si è già ricevuto risposta positiva, valore disponibile in *source*
- >2, codice di errore SDO, vedi tabella sotto

Gestione bloccante

- Imporre ***stru*.phase = -1** per eseguire la scrittura.
- Il comando di scrittura del valore *source* viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.
- Attesa della risposta. Se la risposta non arriva entro un massimo di 3 secondi la funzione esce con un errore (timeout, rende -2).
- Ricezione risposta con esito **positivo**. Il valore letto viene scritto in *source*. RTE cambia *stru*.phase = 4.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -2 gestore comunicazione occupato
  - -3 crash gestore comunicazione (riavvio)
  - -4 gestione bloccante non permessa nel task corrente (*stru*.phase = 1) (rule, ladder). Comando non accettato
  - -5 comando a un dispositivo non presente (*stru*.rid)
  - -6 nessuna operazione in esecuzione. *stru*.phase non inizializzata
  - -12 valore illegale di objcode (allarme 9900 arg 12)
  - -13 valore illegale di subindex (allarme 9900 arg 13)
  - -14 valore illegale di datatype (allarme 9900 arg 4/arg 14)
  - -15 valore illegale di datalen (allarme 9900 arg 5/arg 15)
  - -16 valore illegale di phase (allarme 9900 arg 6)
  - -21 *source* non specificato (NULL) (allarme 9900 arg 16)
  - -22 *source* cambiato (esecuzione a fasi) (allarme 9900 arg 32)
  - 2 comando di cui si è già ricevuto risposta positiva, valore disponibile in *source*
  - >2 codice di errore SDO, vedi tabella sotto

Codici di errore SDO SoE/CoE

© 2025 Robox SpA

SDO error code	Descrizione
0x0503 0000	Toggle bit non alternato.
0x0504 0000	Timeout del protocollo SDO.
0x0504 0001	Identificatore del comando client/server non valido o sconosciuto.
0x0504 0002	Dimensione del blocco non valida (solo gestione bloccante).
0x0504 0003	Numero di sequenza non valido (solo gestione bloccante).
0x0504 0004	Errore CRC (solo gestione bloccante).
0x0504 0005	Eccesso di memoria.
0x0601 0000	Accesso non supportato a un oggetto.
0x0601 0001	Tentativo di leggere un oggetto a sola scrittura.
0x0601 0002	Tentativo di scrivere un oggetto a sola lettura.
0x0602 0000	L'oggetto non esiste nel dizionario oggetti.
0x0604 0041	Impossibile mappare l'oggetto sul PDO.
0x0604 0042	Il numero e la lunghezza degli oggetti da mappare supererebbero la lunghezza del PDO.
0x0604 0043	Incompatibilità dei parametri.
0x0604 0047	Incompatibilità interna al dispositivo.
0x0606 0000	Accesso non riuscito a causa di un errore hardware.
0x0607 0010	Il tipo di dati non corrisponde, la lunghezza del parametro di servizio non corrisponde.
0x0607 0012	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo alto.
0x0607 0013	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo basso.
0x0609 0011	Il sottoindice non esiste.
0x0609 0030	Superamento del campo di valori del parametro (solo per accesso in scrittura).
0x0609 0031	Valore del parametro scritto troppo alto.
0x0609 0032	Valore del parametro scritto troppo basso.

© 2025 Robox SpA

SDO error code	Descrizione
0x0609 0036	Il valore massimo è inferiore al valore minimo.
0x0800 0000	Errore generale.
0x0800 0020	I dati non possono essere trasferiti o archiviati nell'applicazione.
0x0800 0021	I dati non possono essere trasferiti o archiviati nell'applicazione a causa del controllo locale.
0x0800 0022	I dati non possono essere trasferiti o archiviati nell'applicazione a causa dello stato attuale del dispositivo.
0x0800 0023	La generazione dinamica del dizionario oggetti non riesce o non è presente alcun dizionario oggetti (ad esempio, il dizionario oggetti viene generato da un file e la generazione non riesce a causa di un errore di file).

Esempi

Esempio di utilizzo:

```
; SAMPLE CODE
; =======
; in task
; =======

struct_p tunnelCommnad
  u8 u
  u8 pdiH
  u8 pdiL
  u8 sidx
end_struct_p
stru_coe_obj 834    wCoe, rCoe
int              esitoFB
TunnelCommnad    tnRead
u8               rCoeO[256]
...
...
    wCoe.rid      = 2        ; axioline
    wCoe.objCode  = 0x3000   ; Axioline F module nn PDI read tunnel
    wCoe.subIndex = 1        ; command
    wCoe.datatype = 10
    wCoe.datalen  = sizeof(tnRead)   ; dimensione dato da scrivere
    wCoe.phase    = -1

    tnRead.u      = 0
```

© 2025 Robox SpA

```
tnRead.pdiH    = 0
tnRead.pdiL    = 0x18
tnRead.sidx    = 0
rCoe.rid       = 2        ; axioline
rCoe.objCode   = 0x3000   ; Axioline F module nn PDI read tunnel
rCoe.subIndex  = 0x3      ; Response
rCoe.datatype  = 10       ; octet
rCoe.datalen   = 256
rCoe.phase     = -1
esitoFB = coe_wobj 284(wCoe, tnRead)
if(esitofb < 0)
   ; errore su scrittura
    ....
endif

; comando di scrittura andato a buon fine., ora per
; completare il tunneling del prodotto PHOENIX- AXIOLINE
; andiamo a leggere la risposta..
esitoFB = coe_robj(rCoe, rCoe0)
if(esitofb < 0)
   ; errore
    ....
endif

; ora ho la risposta in rCoe0 delle info del pdi 18
...
...
```

Funzione command()

Funzione che specifica una direttiva ascii che deve essere eseguita dal controllo assi.

Sintassi	**string command (string strCmd)**
strCmd	Parametro che contiene la direttiva da far eseguire al controllo assi.
Risultato	Rende: Valore: Risposta alla direttiva — Descrizione: La direttiva è stata riconosciuta ed è prevista una risposta. Valore: "unknown command" — Descrizione: La direttiva non è stata riconosciuta. Valore: "OK" — Descrizione: La direttiva è stata eseguita e non è prevista una risposta.
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**string command (string strCmd)**
Note	

Nel capitolo 'connettività-->comunicazione seriale protocollo libero' viene illustrata la possibilità, via collegamento seriale, di inviare dei comandi seriali opportunamente formattati e ottenere da RTE delle risposte.

Questa funzione serve appunto per "simulare" la richiesta fatta da seriale.

Esempi

Esempio di utilizzo:

```
sr(4) = command(288)("D R 1")
; sr(4) = "    0 0x00000000")
;------------------------------
sr(4) = command("D AL 1")
; sr(4) = " 0 no alarm"
;------------------------------
sr(4) = command("sysinfo")
; sr(4) = "unknown command"
;------------------------------
sr(4) = command("f_inp 171")
; sr(4) = "ok"
```

Funzione cond_val()

Funzione che permette di imporre una grandezza in funzione dello stato logico di una condizione.

Sintassi	**real cond_val (cond, real valTrue, real valFalse)**
cond	Condizione da valutare. Può essere anche un'espressione generica.
valTrue	Valore del risultato in caso di *cond* TRUE.
valFalse	Valore del risultato in caso di *cond* FALSE.
Risultato	Rende *valTrue* o *valFalse* in base allo stato logico della condizione.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
rr(1) = cond_val(289)(r(100) = 4, 24.18, -24.18)
; è equivalente a
if(r(100) = 4))
```

© 2025 Robox SpA

```
    rr(1)= 24.18
else
    rr(1)= -24.18
endif


;--------
int x = 5
int y = 8
int result = cond_val(x > y, inc(x), inc(y))
; A questo punto x = 5 e y = 9
```

Funzione cos()

Funzione che esegue il coseno dell'angolo in radianti specificato.

Sintassi	**real cos (real value)**
value	Argomento del coseno. **NOTA**: *value* deve essere in radianti.
Risultato	Rende il coseno di *value*.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
rr(5) = 0

rr(10) = cos(rr(5))              ; = 1
rr(10) = cos(K_PI/4)             ; = 0.7071...
rr(101) = cos(torad(rr(100)) ; rr (100) --> angolo in gradi
                                 ; rr (101) --> coseno dell'angolo contenu
```

Funzione crc16_be()

Dato un buffer di dati (sequenza di byte), questa funzione esegue il calcolo del CRC16 in base al polinomio e al valore CRC iniziale specificati.

Sintassi	**u16 crc16_be (string strData, i32 nCount [, u16 initVal=0xFFFF] [, u16 polin=0xA001])**
strData	Stringa oppure array, dei quali si vuole calcolare il CRC16. Il calcolo del CRC avviene comunque a byte, indipendentemente dal tipo specificato per la variabile data.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>u16 crc16_be (string strData, i32 nCount [, u16 initVal=0xFFFF] [, u16 polin=0xA001])</td></tr>
<tr><td>nCount</td><td>È il numero di byte di strData su cui eseguire il calcolo.</td></tr>
<tr><td>initVal</td><td>Valore iniziale del CRC. (opzionale, default 0xFFFF)</td></tr>
<tr><td>polin</td><td>Valore del polinomio usato nel calcolo del CRC. (opzionale, default 0xA001)
Principali polinomi utilizzati dai diversi CRC:
<table>
<tr><th>Polinomio</th><th>Valore polin</th></tr>
<tr><td>Normal original</td><td>0x8005</td></tr>
<tr><td>Normal reversed</td><td>0xA001</td></tr>
<tr><td>Koopman original</td><td>0xC002</td></tr>
<tr><td>Normal reciprocal</td><td>0x4003</td></tr>
<tr><td>Reversed reciprocal</td><td>0xC002</td></tr>
<tr><td>Koopman reciprocal</td><td>0xA001</td></tr>
</table>
Casi frequenti:
<table>
<tr><th>CRC più usati</th><th>Rappresentazione</th><th>NORMALE</th><th>REVERSE</th><th>RECIPROCO</th><th>Usato da</th></tr>
<tr><td>CRC-16-CCITT</td><td>x16 + x12 + x5 + 1</td><td>0x1021</td><td>0x8408</td><td>0x0811</td><td>Modem X25, Bluetooth, PPP</td></tr>
<tr><td>CRC-16-IBM</td><td>x16 + x15 + x2 + 1</td><td>0x8005</td><td>0xa001</td><td>0x4003</td><td>USB, MODBUS, altri</td></tr>
</table></td></tr>
<tr><td>Risultato</td><td>Rende il valore del CRC calcolato.</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td>Chiamando ripetutamente la funzione con valore iniziale uguale al valore del CRC calcolato, si può eseguire il calcolo incrementale del CRC di un buffer di lunghezza qualsiasi.
NOTA: Le precedenti funzioni crc16 e crc16_init sono deprecate, ma rimangono per compatibilità.</td></tr>
</table>

Il calcolo viene effettuato secondo l'algoritmo Big Endian.

Differisce dall'algoritmo crc16_le() perchè si effettua uno shift del CRC in elaborazione a sinistra anzichè a destra e perchè il byte di dati viene inserito nella parte alta anzichè bassa della word su cui si esegue il calcolo.

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

```
crcData = crc16_be(data, numBytes, crcInit, pol)
```

Valore crcData	data *(strData)*	numBytes *(nCount)*	crcInit *(initVal)*	pol *(polin)*
50244	"d ve"	4	0xFFFF	0xA001
16302	"d ve"	4	0	0xA001
35609	"d ve"	4	0	0x1021
1978	"13564"	5	0xFFFF	0xA001
11532	"13564"	5	0	0xA001

Funzione crc16_le()

Questa funzione, dato un buffer di dati (sequenza di byte), esegue il calcolo del CRC16 in base al polinomio e al valore CRC iniziale specificati.

Sintassi	**u16 crc16_le (string strData, i32 nCount [, u16 initVal=0xFFFF] [, u16 polin=0xA001])**
strData	Stringa oppure array, dei quali si vuole calcolare il CRC16. Il calcolo del CRC avviene comunque a byte, indipendentemente dal tipo specificato per la variabile data.
nCount	È il numero di byte di *strData* su cui eseguire il calcolo.
initVal	Valore iniziale del CRC. (opzionale, default 0xFFFF)
polin	Valore del polinomio usato nel calcolo del CRC. (opzionale, default 0xA001) Principali polinomi utilizzati dai diversi CRC:

Polinomio	Valore *polin*
Normal original	0x8005
Normal reversed	0xA001
Koopman original	0xC002
Normal reciprocal	0x4003
Reversed reciprocal	0xC002

© 2025 Robox SpA

Sintassi	**u16 crc16_le (string strData, i32 nCount [, u16 initVal=0xFFFF] [, u16 polin=0xA001])**
	Polinomio / Valore *polin*: Koopman reciprocal / 0xA001 Casi frequenti: CRC più usati / Rappresentazione / NORMALE / REVERSE / RECIPROCO / Usato da: CRC-16-CCITT / x16 + x12 + x5 + 1 / 0x1021 / 0x8408 / 0x0811 / Modem X25, Bluetooth, PPP CRC-16-IBM / x16 + x15 + x2 + 1 / 0x8005 / 0xa001 / 0x4003 / USB, MODBUS, altri
Risultato	Rende il valore del CRC calcolato.
Validità	Rule, Task.
Note	Chiamando ripetutamente la funzione con valore iniziale uguale al valore del CRC calcolato, si può eseguire il calcolo incrementale del CRC di un buffer di lunghezza qualsiasi. **NOTA**: La precedente funzione crc16_ero (Little Endian con polinomio iniziale non modificabile 0xA001) è deprecata, ma rimane per compatibilità.

Il calcolo viene effettuato secondo l'algoritmo Little Endian.

Esempi

Esempio di utilizzo:

```
crcData = crc16_le(data, numBytes, crcInit, pol)
```

Valore crcData	data *(strData)*	numBytes *(nCount)*	crcInit *(initVal)*	pol *(polin)*
38392	"d ve"	4	0xFFFF	0xA001
45560	"d ve"	4	0	0xA001
2126	"d ve"	4	0	0x1021
63824	"13564"	5	0xFFFF	0xA001
63860	"13564"	5	0	0xA001

© 2025 Robox SpA

Funzione ct_cv()

Funzione per leggere il valore corrente del counter.

NOTA: L'analoga istruzione ct_cv() serve invece per impostare il valore corrente del counter.

Sintassi	**real ct_cv (counter cnt)**
cnt	Nome del counter (definito come tipo di variabile).
Risultato	Rende il valore del conteggio.
Validità	Rule, Task.
Note	Utile per creare counter ritentivi.

Esempi

Esempio di utilizzo:

```
ct_cv(C1, rr(1000))   ; scrivo il valore contenuto nel registro reale
currVal = ct_cv(C1)   ; leggo il valore corrente del counter e lo scriv
```

Funzione ct_pv()

Istruzione per leggere il valore di preset del counter.

NOTA: L'analoga istruzione ct_pv() serve invece per scivere il valore di preset del counter.

Sintassi	**real ct_pv (counter cnt)**
cnt	Nome del counter (definito come tipo di variabile).
Risultato	Rende il valore del preset.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
ct_pv(C1, rr(1000))     ; scrivo il valore contenuto nel registro rea
presetVal = ct_pv(C1)   ; leggo il valore del preset e lo scrivo nella
```

© 2025 Robox SpA

Funzione ct_qd()

Funzione per leggere il valore di stato dell'uscita del counter (rispetto al valore 0).

Sintassi	**i32 ct_qd (counter cnt)**
cnt	Nome del counter 921 (definito come tipo di variabile).
Risultato	Rende lo stato d'uscita del counter.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
out(257) = ct_qd 295(C1)            ; out 257 on se il valore corrente ct
out(257) = ct_qd(C1) and AUTO   ; out 257 on se il valore corrente ct_q
```

Funzione ct_qu()

Funzione per leggere il valore di stato dell'uscita del counter (rispetto al preset (ct_pv 294 ())).

Sintassi	**i32 ct_qu (counter cnt)**
cnt	Nome del counter 921 (definito come tipo di variabile).
Risultato	Rende lo stato d'uscita del counter.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
out(257) = ct_qu 295(C1)            ; out 257 on se il valore di preset c
out(257) = ct_qu(C1) and AUTO   ; out 257 on se il valore di preset ct_
```

Funzione dec()

Decrementa una variabile della grandezza specificata e rende inoltre il suo nuovo contenuto.

Sintassi	**real dec (real var [, real value=1])**
var	Variabile da decrementare.
value	Valore di decremento. (opzionale, default 1)

© 2025 Robox SpA

Sintassi	**real dec (real var [, real value=1])**
Risultato	Rende il nuovo contenuto di *var* (risultato della sottrazione).
Validità	Rule, Task.
Note	Il risultato può anche essere omesso.

Esempi

Esempio di utilizzo:

```
pluto = 150
rr(1) = 5
rr(1) = dec(rr(1))
rr(2) = dec(pluto, 100)
; rr(1) sarà uguale a 4
; rr(2) sarà uguale a 50

for(i=8, i > 0, dec(i))
; corrisponde a for(i=8, i>0, i=i-1)

appo = dec(pluto, 10)
; corrisponde a:
; pluto = pluto - 10
; appo  = pluto
```

Funzione delta_enc16()

Questa funzione serve ad eseguire il calcolo della differenza fra due grandezze intere, gestendo correttamente lo "scaravolto" dei dati ogni volta che il valore di un parametro entrante ha superato il range della grandezza utilizzata.

Sintassi	**real delta_enc16 (real actEnc, real oldEnc)**
actEnc	Contiene la prima grandezza (valore attuale). (formato int a 16 bit: range -32768-:-32767)
oldEnc	Contiene la grandezza da sottrarre alla prima (valore precedente). (formato int a 16 bit: range -32768-:-32767)
Risultato	Rende la differenza *actEnc* - *oldEnc*, modulata come se fosse una variabile di tipo intero a 16 bit.
Validità	Rule, Task.
Note	

Esempio tipico di utilizzo della funzione è quello della costruzione di una quota reale in base alla lettura di un encoder incrementale.

© 2025 Robox SpA

Nel caso di encoder, se *actEnc* è la lettura attuale dell'encoder e *oldEnc* è la lettura effettuata nella precedente battuta di calcolo, la funzione restituisce il numero di bit cui l'encoder si è spostato nel periodo compreso fra le due letture.

Esempi

Esempio di utilizzo:

```
real quotaAsse
real actEnc, oldEnc
; lettura encoder (da rule)
oldEnc = actEnc
actEnc = inp_w(xxx)    ; vedi file RHW.cfg
quotaAsse = quotaAsse + delta_enc16(actEnc, oldEnc)

; var real 32bitHex I32
argAct = 1000.0 0x000003e8
argOld = 900.0  0x00000384
reso   = 100    0x00000064

argAct = 900.0  0x00000384
argOld = 1000.0 0x000003e8
reso   = -100   0xffffff9c

argAct = 10.0   0x0000000a
argOld = -10.0  0xfffffff6
reso   = 20     0x00000014

argAct = -10.0  0xfffffff6
argOld = 10.0   0x0000000a
reso   = -20    0xffffFFEC

argAct = -2147483638 0x8000000A
argOld = 2147483638  0x7ffffff6
reso   = 20          0x00000014

argAct = 2147483638  0x7ffffff6
argOld = -2147483638 0x8000000A
reso   = -20         0xFFFFFFEC
```

Funzione delta_enc32()

Questa funzione serve ad eseguire il calcolo della differenza tra i due argomenti, dopo averli ridotti a entità a 32 bit di tipo fix.

Sintassi	**real delta_enc32 (real actEnc, real oldEnc)**
actEnc	Contiene la prima grandezza (valore attuale). (formato int a 32 bit: range -2147483648-:-2147483647)
oldEnc	Contiene la grandezza da sottrarre alla prima (valore precedente). (formato int a 32

© 2025 Robox SpA

Sintassi	**real delta_enc32 (real actEnc, real oldEnc)**
	bit: range -2147483648-:-2147483647)
Risultato	Rende la differenza *actEnc* - *oldEnc*, modulata come se fosse una variabile di tipo intero a 32 bit.
Validità	Rule, Task.
Note	

Esempio tipico di utilizzo della funzione è quello della costruzione di una quota reale in base alla lettura di un encoder incrementale.

Nel caso di encoder, se *actEnc* è la lettura attuale dell'encoder e *oldEnc* è la lettura effettuata nella precedente battuta di calcolo, la funzione restituisce il numero di bit cui l'encoder si è spostato nel periodo compreso fra le due letture.

Esempi

Esempio di utilizzo:

```
real quotaAsse
real actEnc, oldEnc
; lettura encoder (da rule)
oldEnc = actEnc
actEnc = inp_w(xxx)      ; vedi file RHW.cfg
quotaAsse = quotaAsse + delta_enc32(actEnc, oldEnc)

; var real 32bitHex I32
argAct = 1000.0 0x000003e8
argOld = 900.0  0x00000384
reso   = 100    0x00000064

argAct = 900.0  0x00000384
argOld = 1000.0 0x000003e8
reso   = -100   0xffffff9c

argAct = 10.0   0x0000000a
argOld = -10.0  0xfffffff6
reso   = 20     0x00000014

argAct = -10.0  0xfffffff6
argOld = 10.0   0x0000000a
reso   = -20    0xffffFFEC

argAct = -2147483638 0x8000000A
argOld = 2147483638  0x7ffffff6
reso   = 20          0x00000014

argAct = 2147483638  0x7ffffff6
argOld = -2147483638 0x8000000A
reso   = -20         0xFFFFFFEC
```

© 2025 Robox SpA

Funzione diff()

Controlla se una variabile o una espressione ha cambiato il suo valore rispetto all'analisi precedente.

Sintassi	**i32 diff (var)**
var	Parametro da controllare. Deve essere di tipo numerico (i8, i16, i32, u8, u16, u32, float, real) o booleano (espressione logica).
Risultato	Rende: Valore: <> 0 — La variabile ha cambiato valore rispetto all'analisi precedente. Valore: 0 — Altrimenti.
Validità	Rule, Task.
Note	La prima volta che viene eseguita, la funzione rende FALSE e memorizza lo stato attuale della condizione. Nel caso si utilizzi diff() in una funzione R3 utente, va evitato che i parametri della funzione stessa siano coinvolti nell'espressione di "condizione". Il risultato non sarebbe quello sperato a causa della sovrapposizione in memoria dello stato di più variabili.

Ogni istruzione di diff() ha la propria memoria dello stato della "condizione".

La valutazione del fronte è fatta tra lo stato attuale della condizione e la memoria relativa alla singola chiamata della funzione.

Per esempio, se abbiamo allo step 1000 `if(diff(inp(1)))` e allo step 1100 `if(diff(inp(1)))`, benchè facciano riferimento alla stessa condizione, le valutazioni sono fatte in modo distinto, ognuna in base allo stato memorizzato, eseguendo step 1000 e poi step 1100.

Esempi

Esempio di utilizzo:

```
_if(diff(r(1)))
    sr(1) = "È cambiato lo stato"

if(diff(rModoOperativo))
    call aggiornaValori
endif

; equivale a:
```

© 2025 Robox SpA

```
int appo
appo = rmodoOperativo
loop:
    if(rmodoOperativo <> appo)
        call aggiornaValori
    endif
    appo = rmodoOperativo
    ...
    ...
    ...
goto loop
```

Funzione dw0()

Questa funzione si usa per leggere o scrivere (istruzione) la prima dword (a partire da quella meno significativa) dell'argomento.

Lettura:

Sintassi	**u32 dw0 (i32|u32|real arg)**
arg	Argomento della prima dword. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende la dword di *arg*.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una dword non esistente ritorna un errore.

Scrittura (istruzione):

Sintassi	**dw0 (i32|u32|real arg) = u32 val**
arg	Argomento della prima dword.
value	Valore da imporre in *arg*.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una dword non esistente ritorna un errore.

Esempio

Esempio di utilizzo:

```
int pippo1, pippo2, pippo3, pippo4
int prova[4]
prova[0] = 0x1122
prova[1] = 0x3344
prova[2] = 0x5566
prova[3] = 0x7788
```

© 2025 Robox SpA

```
pippo1 = dw0(prova[0]) ; pippo1 = 0x33441122
pippo2 = dw1(prova[0]) ; pippo2 = 0x77885566
pippo3 = dw0(prova[2]) ; pippo3 = 0x77885566
pippo4 = dw1(prova[2]) ; pippo4 = 0x00000000

STRING pippo[26]
STRING pluto[26]
pippo = "abcdefghilmnopqrstuv12345"

dw0(pluto[0]) = dw1(pippo[0])
dw1(pluto[0]) = dw0(pippo[0])
; in pluto avrò "efghabcd"

; con RTE 33.9.0:
i16 pippo = 123
dw1(pippo) = 9  ; errore!
```

Funzione dw1()

Questa funzione si usa per leggere o scrivere (istruzione) la seconda dword (a partire da quella meno significativa) dell'argomento.

Lettura:

Sintassi	**u32 dw1 (real arg)**
arg	Argomento della seconda dword. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende la dword di *arg*.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una dword non esistente ritorna un errore.

Scrittura (istruzione):

Sintassi	**dw1 (real arg) = u32 val**
arg	Argomento della seconda dword.
value	Valore da imporre in *arg*.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una dword non esistente ritorna un errore.

© 2025 Robox SpA

Funzione dwbe2le()

Converte big endian dword (x32) in little endian dword (x32).

Sintassi	**i32|u32 dwbe2le (i32|u32 data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
u32 input
u32 result

input = 0x01020304
result = dwbe2le(input)
; result = 0x04030201
```

Funzione dwbe2lef()

Converte big endian dword (x32) in little endian float (4 byte).

Sintassi	**float dwbe2lef (i32|u32 data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
u32 input
float result

input = 0xdb0f4940     ; data memory: 40 49 0f db hex
result = dwbe2lef(input)
; result = 3.1415927 -> data memory: db 0f 49 40 hex
```

© 2025 Robox SpA

Funzione dwbe2mbe()

Converte big endian dword (x32) in middle big endian dword (x32).

Sintassi	**i32|u32 dwbe2mbe (i32|u32 data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
u32 input
u32 result

input = 0x01020304
result = dwbe2mbe(input)
; result = 0x03040102
```

Funzione dwbe2mle()

Converte big endian dword (x32) in middle little endian dword (x32).

Sintassi	**i32|u32 dwbe2mle (i32|u32 data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
u32 input
u32 result

input = 0x01020304
result = dwbe2mle(input)
; result = 0x02010403
```

© 2025 Robox SpA

Funzione dwle2be()

Converte little endian dword (x32) in big endian dword (x32).

Sintassi	**i32|u32 dwle2be (i32|u32 data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
u32 input
u32 result

input = 0x04030201
result = dwle2be 304 (input)
; result = 0x01020304
```

Funzione dwle2bef()

Converte little endian float (4 byte) in big endian dword (u32).

Sintassi	**u32 dwle2bef (float data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
float input
u32 result

input = 3.1415927      ; data memory: db 0f 49 40 hex
result = dwle2bef 304 (input)
; result = 0xdb0f4940 -> data memory: 40 49 0f db hex
```

© 2025 Robox SpA

Funzione dwle2mbe()

Converte little endian dword (x32) in middle big endian dword (x32).

Sintassi	**i32|u32 dwle2mbe (i32|u32 data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
u32 input
u32 result

input = 0x04030201
result = dwle2mbe(input)
; result = 0x03040102
```

Funzione dwle2mbef()

Converte little endian float (4 byte) in middle big endian dword (u32).

Sintassi	**u32 dwle2mbef (float data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
float input
u32 result

input = 3.1415927        ; data memory: db 0f 49 40 hex
result = dwle2mbef(input)
; result = 0x4940db0f -> data memory: 0f db 40 49 hex
```

© 2025 Robox SpA

Funzione dwle2mle()

Converte little endian dword (x32) in middle little endian dword (x32).

Sintassi	**i32|u32 dwle2mle (i32|u32 data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
u32 input
u32 result

input = 0x04030201
result = dwle2mle 306(input)
; result = 0x02010403
```

Funzione dwle2mlef()

Converte little endian float (4 byte) in middle little endian dword (u32).

Sintassi	**u32 dwle2mlef (float data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
float input
u32 result

input = 3.1415927        ; data memory: db 0f 49 40 hex
result = dwle2mlef 306(input)
; result = 0x0fdb4049 -> data memory: 49 40 db 0f hex
```

© 2025 Robox SpA

Funzione dwmbe2be()

Converte middle big endian dword (x32) in big endian dword (x32).

Sintassi	**i32|u32 dwmbe2be (i32|u32 data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
u32 input
u32 result

input = 0x03040102
result = dwmbe2be 307(input)
; result = 0x01020304
```

Funzione dwmbe2le()

Converte middle big endian dword (x32) in little endian dword (x32).

Sintassi	**i32|u32 dwmbe2le (i32|u32 data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
u32 input
u32 result

input = 0x04030201
result = dwmbe2le 307(input)
; result = 0x03040102
```

© 2025 Robox SpA

Funzione dwmbe2lef()

Converte middle big endian dword (u32) in little endian float (4 byte).

Sintassi	**float dwmbe2lef (u32 data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
u32 input
float result

input = 0x4940db0f       ; data memory: 0f db 40 49 hex
result = dwmbe2lef 308(input)
; result = 3.1415927 -> data memory: db 0f 49 40 hex
```

Funzione dwmle2be()

Converte middle little endian dword (x32) in big endian dword (x32).

Sintassi	**i32|u32 dwmle2be (i32|u32 data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
u32 input
u32 result

input = 0x02010403
result = dwmle2be 308(input)
; result = 0x01020304
```

© 2025 Robox SpA

Funzione ecat_dev_on_off()

Questa funzione abilita/disabilita un dispositivo sulla linea EtherCAT.

NOTA: Per poter abilitare/disabilitare il dispositivo, la linea deve essere in stato INIT (ecat_set_state 314(*nEth*, 1)).

Sintassi	**i32 ecat_dev_on_off (i32 nEth, i32 cfgAddr , bool sts)**
nEth	Numero canale Ethernet.
cfgAddr	Indirizzo del dispositivo.
sts	Stato desiderato:

Valore	Descrizione
TRUE	Dispositivo abilitato.
FALSE	Dispositivo non abilitato.

Risultato	Rende:

Valore	Descrizione
0	Operazione completata.
-1	Numero canale Ethernet illegale.
-2	Canale Ethernet non è EtherCAT.
-3	EtherCAT non in stato di INIT.
-4	Nessun dispositivo nella posizione selezionata.
-5	Linea EtherCAT non configurata correttamente.
-6	Errore sw interno. (contattare Robox)

Validità	Task.
Note	

Esempi

Esempio di utilizzo:

```
;
```

© 2025 Robox SpA

```
FUNCTION int Ecat_Change_Config(int nEth, int addr, bool sts, bool res

; it performs the following operations :
; 1 set the ethercat in INIT state
; 2 change the device status
; 3 wait for the restart Ecat flag
; 4 set the ethercat in OPERATIONAL state
; 5 wait for OPERATIONAL state is reached, if not an error is set
; 6 when OPERATIONAL state is reached, the alarm due to this operation
;
; parameters
; nEth  number of ethernet (1 -:-...)
; addr  device config_addr
; sts   1 --> device present  0 --> device absent
; restartEcat  flag to command the restart of the ethercat line (to be
;               device has been physically removed or restored to ethe
; phase  working fase to be set to 0 to restart the opreation
; return
;   0   operation in execution
;   1   operation to change the status done, waiting to 'restartEcat'
;   2   operation to restart ethercat line  is done
;  -1   no such device
;  -2   illegal phase
;  -3   unable to  restart ecat , check device
    select(phase)
        default
            phase = -2
        case -2
            return -2       ; illegal phase

        case -3
            return -3       ; unable to restart ecat , check device

        case -1
            return -1       ; error .. no such device

        case 0
            ; set ethercat line in init mode
            _if(nEth = 0)       ; 0 means alla ethercat lines
                nEth = -1       ; not acceptable here
            esitoECC = ecat_set_state(314)(nEth, 1)
            if (esitoECC = 0)
                phase = 1
            endif
            return 0
        case 1  ; wait the init state
            if(ecat_get_state(312)(nEth) <> 1)
                return 0
            endif
            ; now the modification are allowed
```

© 2025 Robox SpA

```
            esitoECC = ecat_dev_on_off(nEth, addr, sts)
; $lin_step esitoecc = ECATSlaveOnOff(neth, addr, sts);
            if(esitoECC < 0)
                phase = -1
                return -1
            endif

            phase = 2

        case 2
            if(not restartEcat)
                return 1
            endif
            ; set ethercat line in operational mode
            esitoECC = ecat_set_state(nEth, 8)
            if(esitoECC = 0)
                phase = 3
                ecatTimeOut = tfb + 10
            endif
            return 0
        case 3
            ecatState = ecat_get_state(nEth)
            if(ecatState = 1)
                if(tfb < ecatTimeOut)
                    return 0
                endif
                ; too much time to change from init state to preopreat
                ; something wrong
                phase = -3
                return -3
            endif
            ; wait the operational state
            if(ecatState <> 8)
                return 0
            endif
            ; ok the ethercat line is in operational....
            ; clean some alarms due to the operation
            alarm_reset(18, 0)
            alarm_reset(82)
            alarm_reset(83)
            phase = 4

        case 4
            return 2
    end_select
    return -2
END_FUN
```

© 2025 Robox SpA

Funzione ecat_get_state()

Funzione che rende il modo operativo NMT della linea EtherCAT indicata.

Sintassi	**i32 ecat_get_state (i32 nEth)**
nEth	Numero di linea EtherCAT.
Risultato	Rende:

Valore	Descrizione
1	INIT
2	PREOPERATIONAL
4	SAFEOPERATIONAL
8	OPERATIONAL
-1	Indice linea EtherCAT illegale.
-2	Porta Ethernet non configurata EtherCAT.

Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
....
; Restart the Ethercat line  on ETH 2
; force the line to INIT STATE
result = ecat_set_state(2, 1)       ; set line 2 in init
if (result < 0)
    alarm_set(UNABLE_TO_SET_ECAT, 0, 1)
    return
endif
mt_wait(ecat_get_state(2) = 1 )     ; wait for line in init state
; ok now restart
result = ecat_set_state(2, 8)       ; set line 2 in operational
if (result < 0)
    alarm_set(UNABLE_TO_SET_ECAT, 0, 2)
    return
endif
mt_wait(ecat_get_state(2) = 8 )     ; wait for line in operational stat
...
```

NOTA: Questo esempio può essere utilizzato solo nel task, in quanto viene usata l'istruzione mt_wait().

© 2025 Robox SpA

Funzione ecat_probe()

Funzione per leggere le informazioni del dispositivo connesso in linea EtherCAT

L'istruzione richiede molto tempo. Deve essere chiamata continuamente fino a che non viene conclusa.

Sintassi	**i32 ecat_probe (i32 nEth, stru_ecat_probe stru [, string devName=""])**
nEth	Numero di canale EtherCAT.
stru	Struttura di tipo stru_ecat_probe 841 che contiene le informazioni.
devName	Stringa dove viene salvato il nome del dispositivo. (opzionale, default "")
Risultato	Rende:

Valore	Descrizione
1	Operazione completata.
0	Operazione in esecuzione.
-1	Indice linea EtherCAT illegale.
-2	Il canale Ethernet non è EtherCAT.
-3	Argomento illegale.
-4	Nessun dispositivo nella posizione selezionata.
-5	Linea EtherCAT non configurata correttamente.
-6	Cavo EtherCAT disconnesso.
-7	Errore sw interno. (contattare Robox)

Validità	Task.
Requisiti di sistema	RC3E 33.7.1-beta0
Note	

Esempi

Esempio di utilizzo:

```
; storage for all the ethercat device in a line (up to N_MAX_ECAT_DEV)
```

© 2025 Robox SpA

```
$LIT N_MAX_ECAT_DEV     100
stru_ecat_probe ecatDev[1+N_MAX_ECAT_DEV]        ; structure to store
string devName[1+N_MAX_ECAT_DEV][64]             ; device name
int nDev                                         ; number of devices fo
int esitoEP                                      ; result of ECAT_PROBE
string appoName[64]                              ; temporary device nam
...
...
;_____________________________________________
FUNCTION void hndEcatProbe(int nEth)
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; function to probe all the devices of an ethercat channel
    int howMany = 0
    nDev = 1
    do
        ecatDev[ndev].phase =  0
        ecatDev[ndev].n_Pos  = nDev
        esitoEp = 0
        while(esitoEp = 0)
            esitoEp = ecat_probe(nEth, ecatDev[nDev], apponame)
    ;$lin_step esitoep = ECATProbe(neth, &ecatdev[ndev], apponame, siz
        end_while
        if(esitoEp = 1)
            howMany = ecatDev[ndev].how_many
            devName[nDev] = apponame
            nDev = nDev+1
        endif
    end_do_while((esitoEp =1) and (nDev <= N_MAX_ECAT_DEV))
    if(nDev = 1)
        nDev = 0
    else
        if(nDev > howMany)
            nDev = howMany
        endif
    endif
END_FUN
```

Funzione ecat_set_state()

Funzione che imposta il modo operativo NMT della linea EtherCAT indicata.

Sintassi	**i32 ecat_set_state (i32 nEth, i32 desState [, bool doBopex=FALSE])**
nEth	Numero di linea EtherCAT. Se 0 si intende tutte le linee EtherCAT presenti.
desState	Stato della rete che si desidera impostare:

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 ecat_set_state (i32 nEth, i32 desState [, bool doBopex=FALSE])</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td>1</td><td>INIT</td></tr>
<tr><td>2</td><td>PREOPERATIONAL</td></tr>
<tr><td>4</td><td>SAFEOPERATIONAL</td></tr>
<tr><td>8</td><td>OPERATIONAL</td></tr>
</table>
</td></tr>
<tr><td>doBopex</td><td>Se uguale a TRUE, attende di essere in SAFEOPERATIONAL prima di eseguire il file comandi bopex.stp. Altrimenti non verrà eseguito il file. (opzionale, default FALSE)</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td>2</td><td>OK:
SAFEOPERATIONAL raggiunto, bopex.stp eseguito.</td></tr>
<tr><td>1</td><td>OK:
SAFEOPERATIONAL raggiunto, bopex.stp non presente.</td></tr>
<tr><td>0</td><td>OK:
stato desiderato imposto.</td></tr>
<tr><td>-1</td><td>Indice linea EtherCAT illegale.</td></tr>
<tr><td>-2</td><td>Porta Ethernet non configurata EtherCAT.</td></tr>
<tr><td>-3</td><td>Codice di stato illegale.</td></tr>
<tr><td>-4</td><td>Rete non partita a causa di un errore di configurazione. (usare mreport -a)</td></tr>
<tr><td>-5</td><td>Funzione usata non in un task di timesharing con flag di doBopex TRUE.</td></tr>
<tr><td>-6</td><td>Timeout per attesa stato di SAFEOPERATIONAL</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 ecat_set_state (i32 nEth, i32 desState [, bool doBopex=FALSE])**
	Valore: — / Descrizione: per poi eseguire bopex.stp: operazione non eseguita.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
....
; Restart the Ethercat line  on ETH 2
; force the line to INIT STATE
result = ecat_set_state(2, 1)       ; set line 2 in init
if (result < 0)
    alarm_set(UNABLE_TO_SET_ECAT, 0, 1)
    return
endif
mt_wait(ecat_get_state(2) = 1 )     ; wait for line in init state
; ok now restart
result = ecat_set_state(2, 8)       ; set line 2 in operational
if (result < 0)
    alarm_set(UNABLE_TO_SET_ECAT, 0, 2)
    return
endif
mt_wait(ecat_get_state(2) = 8 )     ; wait for line in operational stat
...
```

NOTA: Questo esempio può essere utilizzato solo nel task, in quanto viene usata l'istruzione mt_wait().

Funzione excl_lam()

Funzione per ottenere la maschera degli assi logici esclusi/inclusi del PowerSet.

Sintassi	**i32 psName.excl_lam**
psName	Nome del PowerSet
Risultato	Ritorna la maschera degli assi logici esclusi/inclusi del PowerSet. (1 esclusi, 0 inclusi)
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**i32 psName.excl_lam**
Note	

Esempi

Esempio di utilizzo:

```
; in cfgdefs.i3
...
; {7:0} Power set variables
...
power_set sys
power_set ps_qua         ; powerset definition
...

; in task o rule
int maskAx               ; axes mask excluded/reincluded
...
...
maskAx = ps_qua.excl_lam
...
```

Funzione excl_pam()

Funzione per ottenere la maschera degli assi fisici esclusi/inclusi del PowerSet.

Sintassi	**i32 psName.excl_pam**
psName	Nome del PowerSet.
Risultato	Ritorna la maschera degli assi fisici esclusi/inclusi del PowerSet. (1 esclusi, 0 inclusi)
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; in cfgdefs.i3
...
; {7:0} Power set variables
...
power_set sys
power_set ps_qua         ; powerset definition
...

; in task o rule
```

© 2025 Robox SpA

```
i32 maskAx                ; axes mask excluded/reincluded
...
...
maskAx = ps_qua.excl_pam
...
```

Funzione f_avail()

Questa funzione è utilizzata per visualizzare la memoria della flash card disponibile sui vari dispositivi logici.

Sintassi	**real f_avail (string path)**
path	Stinga contenente il nome di una directory.
Risultato	Rende un valore contenente il numero di byte liberi sul *path* indicato.
Validità	Task.
Requisiti di sistema	**ATTENZIONE**: Questa funzione non è valida per versioni di RDE >= 33.
Note	Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

Esempi

Esempio di utilizzo:

```
esito = f_avail(/FA/)
```

Funzione f_close()

Questa funzione permette di chiudere un file su flash eprom precedentemente aperto tramite l'istruzione f_open().

Sintassi	**i32 f_close (stru_filehnd stru)**
stru	Variabile di tipo stru_filehnd che contiene l'handle ottenuto tramite la funzione f_open().
Risultato	Rende:

Valore	Descrizione
0	Esito positivo dell'operazione.
EOF (codice 0x1a)	Esito negativo dell'operazione.

© 2025 Robox SpA

Sintassi	**i32 f_close (stru_filehnd stru)**
Validità	Task.
Note	Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

Esempi

Esempio di utilizzo:

```
int result_close
int result_open
string result
stru_filehnd 842 cf

result_open = f_open 327 ("/fa/ipaddr2.def", "wb", 0, cf)
...
result_close = f_close 318 (cf)
if(result_close <> 0)
    result = "function f_close: negative result"
endif
```

Funzione f_copy()

Questa funzione è utilizzata per copiare il contenuto di un file, presente sulla CF del controllo assi collegato, in un altro file specificato.

Sintassi	**i32 f_copy (string srcPathName, string destPathName)**
srcPathName	Nome attuale del file compreso di percorso.
destPathName	Nome desiderato del file compreso di percorso. Il percorso di destinazione può essere anche diverso.
Risultato	Rende: **Valore** / **Descrizione** 0 — Esito positivo dell'operazione. -21 — Il file di destinazione esiste già o è una directory. -17 — Il file di destinazione esiste già.

© 2025 Robox SpA

Sintassi	**i32 f_copy (string srcPathName, string destPathName)**
	Valore / Descrizione -2 — Il file sorgente non esiste. -nn — Codice ERRNO di errore. (vedi codici di errore 59)
Validità	Task.
Note	Questa funzione blocca l'evoluzione del task fino a che l'operazione non è terminata. Si tenga presente che la copia di un file di 120K impiega circa 1.35 sec su uRmc2 e circa 0.8 sec su uRmc3. Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

Esempi

Esempio di utilizzo:

```
int result_copy
string result

result_copy = f_copy 319("/fa/test.def", "/fa/test1.def")
if(result_copy = 0)
   result = "function f_copy: positive result"
endif
```

Funzione f_del()

Questa funzione è utilizzata per la cancellazione di uno o più file presenti in una partizione.

Sintassi	**i32 f_del (string fileName)**
fileName	Percorso completo della directory/file che si vuole eliminare. /fx/filename dove: • x è il carattere identificatore del dispositivo logico (@, A, B, C, D, E) • filename è il nome del file con wildcard (* e ?)
Risultato	Rende:

© 2025 Robox SpA

Sintassi	**i32 f_del (string fileName)**
	Valore / Descrizione
	> 0 — Numero di file che sono stati cancellati.
	< 0 — Codice di errore (vedi codici di errore 591)
Validità	Task.
Note	Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

Esempi

Esempio di utilizzo:

```
int result_del
string result

result_del = f_del("/fa/test.def")
if(result_del<=0)
   result = "function f_del: error"
endif
```

Funzione f_dir()

Questa funzione permette di ottenere determinate informazioni sui file presenti in flash.

Sintassi	**i32 f_dir (string path, i32 ndef=0 [, stru_fileinfo stru[]=NULL])**
path	Percorso della directory contenente il file voluto (o più di uno) o del file stesso. Se si vuole fare riferimento a uno specifico file bisogna mettere il percorso completo (ad es. "/fa/override.cfg"). Se si vuole riferirsi a uno (non direttamente) o più file si deve utilizzare il carattere *, dove * sostituisce un qualsiasi carattere o stringa (ad es. "/fa/*.cfg" richiedo informazioni su tutti i file, a prescindere dal nome, con estensione cfg contenuti nella cartella fa). I caratteri identificatori del folder possono essere @, A, B, C, D, E, ... (ad es. /f@/, /fb/ , ...). **NOTA1**: Se inserisco il nome completo di un file presente nel folder, ottengo informazioni

© 2025 Robox SpA

Sintassi	**i32 f_dir (string path, i32 ndef=0 [, stru_fileinfo stru[]=NULL])**
	circa la dimensione del file stesso (ad es "/fa/stat.txt" avrò la dimensione del file stat.txt). **NOTA2**: Se si sta utilizzando una vecchia flash-card con una CPU Pentium, allora il campo *path* sarà solamente /fx/ e non /fx/*.*.
nDef	Parametro contenente il numero di informazioni desiderate (5 = 5 files) (opzionale, default 0)
stru	Array di stru_fileinfo 843 che conterrà le informazioni sui file trovati. (opzionale, default NULL--> le informazioni desiderate non verranno salvate) Verranno salvati nome e dimensione dei file trovati nella cartella definita da *path.* Deve essere di almeno *nDef* elementi.
Risultato	Rende il numero di file presenti nella partizione indicata.
Validità	Task.
Note	Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

Esempi

Esempio di utilizzo:

```
; Supponiamo di avere la cartella /fb/ con dentro due file: test1.def

int result
stru_fileinfo 843 infoFile[2]

result = f_dir 321("/FB/*.*")
; result = 2

result = f_dir("/FB/test1.def")
; result = 1

result = f_dir("/FB/*.*", 0, infofile)
; result = 2
; infofile[0].name = ""
; infofile[0].len = 0
; infofile[1].name = ""
; infofile[1].len = 0
```

© 2025 Robox SpA

```
result = f_dir("/FB/*.*", 2, infofile)
; result = 2
; infofile[0].name = "test1.def"
; infofile[0].len = 161
; infofile[1].name = "test2.def"
; infofile[1].len = 30

result = f_dir("/FB/test1.def", 2, infofile)
; result = 1
; infofile[0].name = "test1.def"
; infofile[0].len = 161
; infofile[1].name = ""
; infofile[1].len = 0
```

Funzione f_format()

ATTENZIONE: Questa funzione non è valida per versioni di RDE >= 33.

Questa funzione serve per eseguire la formattazione di un dispositivo logico.

Sintassi	**i32 f_format (string path [, string result=NULL] [, real timeOut=0])**
path	Parametro contenente il nome della cartella. Può assumere i seguenti caratteri: • /FB/ • /FC/ Usare direttiva fmem da shell per visionare le partizioni presenti sulla flash.
result	Parametro contenente l'esito dell'operazione. (opzionale, default NULL) Può essere:

Valore	Descrizione
"Ok"	Operazione terminata con successo.
"timeout system not available"	Gestore occupato o scaduto il timeout.
"illegal or not allowed device"	*path* sbagliato, inesistente o non ammesso (F@, FA).

timeOut	Parametro contenente il tempo di attesa massimo per l'esecuzione dell'istruzione. (opzionale, default 0)
Risultato	Rende:

© 2025 Robox SpA

Sintassi	**i32 f_format (string path [, string result=NULL] [, real timeOut=0])**
	<table><tr><th>Valore</th><th>Descrizione</th></tr><tr><td><> 0</td><td>Operazione eseguita con successo.</td></tr><tr><td>0</td><td>Operazione fallita.</td></tr></table> Utilizzare i parametri opzionali *result* e *timeOut* per avere maggiore diagnostica.
Validità	Task.
Requisiti di sistema	
Note	Tutto il contenuto del dispositivo in questione viene cancellato. RTE impedisce la formattazione di F@ e FA. Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

Diagnostica

- Genera allarme 9900 illegal arg A:1 T:x ST:xxx se parametro *path* illegale (dove T = numero di task e ST numero di step).

Esempi

Esempio di utilizzo:

```
string esitoOper
reso = f_format("/fb/")
reso = f_format("/fb/", esitoOper)
reso = f_format("/fb/", esitoOper , 5)
f_format("/FC/")
```

Funzione f_gets()

Questa funzione è utilizzata per leggere un blocco di dati da un file che sia stato aperto in modo lettura testo.

Sintassi	**i32 f_gets (string buff, i32 maxLen, stru_filehnd stru)**
buff	In *buff* viene copiata la stringa letta dal file. La lettura si conclude quando viene incontrato il carattere terminatore di riga LF (Line Feed, codice 0X0A). Il carattere LF è messo come ultimo carattere della stringa.

© 2025 Robox SpA

Sintassi	**i32 f_gets (string buff, i32 maxLen, stru_filehnd stru)**
maxLen	Specifica la dimensione massima di ciascuna stringa letta dal file (considerando il carattere di terminazione). **NOTA**: Il buffer deve essere almeno un carattere più grande di *maxLen*.
stru	Variabile di tipo stru_filehnd 842 che contiene l'handle ottenuto tramite la funzione f_open 327().
Risultato	Rende: Valore / Descrizione 0 / Operazione fallita. 1 / Operazione eseguita con successo.
Validità	Task.
Note	Se nel file è presente una stringa più lunga di *maxLen*, questa verrà troncata alla lunghezza *maxLen*. La parte troncata verrà letta alla prossima chiamata di f_gets 324(). Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

Esempi

Esempio di utilizzo:

```
; Supponiamo di avere in /fa/ il file autoexec.stp, composto da queste
; sr(1) = "test.def"
; sr(2) = "rt"

int result_gets
int result_open
string gets_buffer
stru_filehnd 842 cf

result_open = f_open 327("/fa/autoexec.stp", "rt", 0, cf)
result_gets = f_gets 324(gets_buffer, sizeof(gets_buffer)-1, cf)
; ottengo gets_buffer = sr(1) \t= "test.def"\n
```

© 2025 Robox SpA

Funzione f_mkdir()

Funzione per creare una nuova directory in flash.

Tutte le directory utilizzate nel percorso scelto, eccetto l'ultima, devono esistere nella flash.

Sintassi	**i32 f_mkdir (string dirName)**
dirName	Nome della nuova directory. Può contenere lettere, spazi e caratteri speciali. Non è case sensitive.
Risultato	Rende: Valore: Descrizione 0: Operazione eseguita con successo. -1: Operazione fallita.
Validità	Task.
Note	

Esempi

Esempio di utilizzo:

```
; making the directory structure:
; machine3
; ... par
; ... ... ax1
; ... ... ax2
; ... ... ax3
; ... ric
; ... doc
esitoF = f_mkdir("/machine3")
if(esitoF = 0)
    esitoF = f_mkdir("/machine3/par")
    if(esitoF = 0)
        esitoF = f_mkdir("/machine3/par/ax1")
        if(esitoF <> 0)
            ; set diagnostic "/machine3/par/ax1"
            ....
        endif
        esitoF = f_mkdir("/machine3/par/ax2")
        if(esitoF <> 0)
            ; set diagnostic on "/machine3/par/ax2"
            ....
        endif
        esitoF = f_mkdir("/machine3/par/ax2")
        if(esitoF <> 0)
            ; set diagnostic on "/machine3/par/ax3"
            ....
        endif
```

© 2025 Robox SpA

```
    else
        ; set diagnostic on "/machine3/par"
        ....
    endif
    esitoF = f_mkdir("/machine3/ric")
    if(esitoF <> 0)
        ; set diagnostic "/machine3/ric"
            ....
    endif
    esitoF = f_mkdir("/machine3/doc")
    if(esitof <> 0)
        ; set diagnostic "/machine3/doc"
            ....
    endif
else
        ; set diagnostic on "/machine3"
        ....
endif
```

Funzione f_open()

Questa funzione permette di aprire un file su CF per lettura o scrittura e di ottenere un handle, che dovrà essere utilizzato come parametro per le successive operazioni sullo stesso file (f_puts 329(), f_close 318(), ..).

Sintassi	**i32 f_open (string fileName, string mode, i32 maxChar, stru_filehnd stru)**
fileName	È il nome (completo di estensione) del file che si intende aprire. **NOTA**: Il nome deve essere preceduto dall'indicazione del dispositivo logico su cui si intende lavorare, che può essere FA, FB, ecc. Può essere una variabile stringa locale o globale, oppure una stringa costante.
mode	È un parametro che specifica la modalità di apertura del file e può assumere uno dei seguenti valori:

Valore	Significato
"rb"	Apertura in lettura, modo binario.
"rt"	Apertura in lettura, modo testo.
"wb"	Apertura in scrittura, modo binario.
"wt"	Apertura in scrittura, modo testo.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 f_open (string fileName, string mode, i32 maxChar, stru_filehnd stru)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>"ab"</td><td>Apertura in append, modo binario.</td></tr>
<tr><td>"at"</td><td>Apertura in append, modo testo.</td></tr>
</table>
</td></tr>
<tr><td>maxChar</td><td>ATTENZIONE: Da RTE >= 33 deve essere impostato uguale a zero.

È un parametro che specifica il numero massimo di caratteri che costituiscono il file.

Tale valore deve essere indicato quando si apre un file in scrittura, mentre deve essere impostato a 0 quando il file viene aperto in lettura.

Questo parametro viene usato dal sistema operativo per riservare in flash una porzione fissa di memoria in cui verrà scritto successivamente il file.</td></tr>
<tr><td>stru</td><td>Variabile di tipo stru_filehnd 842 dove verrà memorizzato l'handle che caratterizza il file aperto e che rimane associato ad esso durante tutte le successive operazioni, fino a quando il file verrà chiuso tramite l'istruzione f_close 318().

Questa variabile dovrà essere utilizzata come parametro in tutte le altre istruzioni di gestione dei file.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td>0</td><td>Esito negativo dell'operazione.</td></tr>
<tr><td><> 0</td><td>Esito positivo dell'operazione.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Task.</td></tr>
<tr><td>Note</td><td>Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.</td></tr>
</table>

Esempi

Esempio di utilizzo:

© 2025 Robox SpA

```
int result_open
string result
stru_filehnd 842 cf

result_open = f_open 327 ("/fa/autoexec.stp", "rb", 0, cf)
if(result_open = 0)
    result = "Not existing file"
endif
```

Funzione f_puts()

Questa funzione è utilizzata per scrivere una stringa in un file che sia stato aperto in modo scrittura tipo testo.

Sintassi	**i32 f_puts (string buff, stru_filehnd stru)**
buff	Stringa da inserire nel file.
stru	Variabile di tipo stru_filehnd 842 che contiene l'handle ottenuto tramite la funzione f_open 327 ().
Risultato	Rende: Valore: 0 — File inesistente. Valore: EOF (codice 0x1a) — Esito negativo. Valore: > 0 — Esito positivo. Il valore reso è quello dell'ultimo byte inserito.
Validità	Task.
Note	Se utilizzo un file non vuoto, il contenuto del file precedente all'utilizzo della funzione viene cancellato. Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

Esempi

Esempio di utilizzo:

```
int result_open
int result_puts
string result
stru_filehnd 842 cf
```

© 2025 Robox SpA

```
result_open = f_open("/fa/test.stp", "wt", 0, cf)
result_puts = f_puts("blablabla", cf)
if(result_puts > 0)
   result = "function f_puts: positive result"
endif
```

Funzione f_read()

Questa istruzione è utilizzata per leggere un blocco di dati da un file che sia stato aperto in modo lettura binaria.

Sintassi	**i32 f_read (buff, i32 blockLen, i32 nBlock, stru_filehnd stru)**
buff	Variabile locale o globale che può essere di qualsiasi tipo. In buffer viene scaricato il blocco di dati letti sequenzialmente dal file.
blockLen	Dimensione in byte di ciascun blocco che deve essere scritto.
nBlock	Numero di blocchi, di dimensione *blockLen*, da scrivere.
stru	Variabile di tipo stru_filehnd che contiene l'handle ottenuto tramite la funzione f_open().
Risultato	Rende: Valore: 0 — Operazione fallita. Valore: > 0 — L'operazione ha avuto successo. Rende il numero degli elementi effettivamente letti. Tale numero coincide di regola con il numero dei blocchi, a meno che non venga incontrata la fine del file prima di avere letto tutti gli elementi specificati.
Validità	Task.
Note	Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg

© 2025 Robox SpA

Sintassi	**i32 f_read (buff, i32 blockLen, i32 nBlock, stru_filehnd stru)**
	(0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

Esempi

Esempio di utilizzo:

```
int result_open
int result_read
string read_buffer
string result
stru_filehnd 842 cf

result_open = f_open 327("/fa/test.stp", "rb", 0, cf)
result_read = f_read 330(read_buffer, 16, 1, cf)
if(result_read = 0)
    result = "function f_read: negative result"
endif
```

Funzione f_read_table()

NOTA: Deprecata. Si consiglia di utilizzare la funzione f_read_table_s 333 (da RTE 33.21).

Legge, da un file presente in flash, una serie di dati che vengono memorizzati in registri reali consecutivi o in un array reale specificato.

Sintassi	**i32 f_read_table (string fileName, i32 nEntry, i32 nData, buff)**
fileName	Nome del file da cui leggere i dati. Esso deve contenere anche la partizione della flash.
nEntry	Numero di entry della tabella.
nData	Numero di dati per entry (max 16), cioè il numero di dati per ogni riga del file.
buff	Deve essere una variabile di uno dei seguenti tipi: • array di variabili di tipo stru_cam 831 • registro reale rr o nvrr, a partire dal quale devono essere memorizzati i dati della camma letti dal file • array locale di variabili reali in cui devono essere memorizzati i dati della camma letti dal file • array di strutture che contengono solo variabili real

© 2025 Robox SpA

Sintassi	**i32 f_read_table (string fileName, i32 nEntry, i32 nData, buff)**
Risultato	Rende:

Valore	Descrizione
> 0	Mancano uno o più elementi. Restituisce il numero della riga a cui manca almeno un elemento.
0	Lettura OK.
-1	Lettura OK. Altri dati nel file.
-2	Lettura OK. Meno dati nel file di quelli previsti. Tabella non corretta.
-10	Il file non esiste.
-11	Numero di dati per linea illegale.
-12	Numero di entry illegale.
-14	File vuoto.

Validità	Task.
Note	Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

I dati ricavati saranno poi utilizzati in una rule per l'esecuzione delle funzioni mv_table 481() e mv_cam 451().

I dati contenuti all'interno del file devono sottostare a quanto segue:

- dimensione massima della linea: 128 caratteri
- dimensione massima della singola voce: 18

Esempi

Esempio di utilizzo:

```
public stru_cam 831 cerchio[201]  ; definisco cerchio come variabile di
; viene definita in un task come pubblica e come extern nelle regole
reso = f_read_table 331("FA:cerchio.tab", 201, 2, cerchio)
;dove il file caricato sulla partizione FA della flash-card è composto
; file cerchio.tab
-1 0
```

© 2025 Robox SpA

```
-0.99 0.14106736
-0.9800 0.198997487


int reso
string nomefile
reso = f_read_table(nomefile, 201, 2, rr(100))
; rr(100) 1 riga 1 colonna
; rr(101) 1 riga 2 colonna
; rr(102) 2 riga 1 colonna
; rr(103) 2 riga 2 colonna
```

Funzione f_read_table_s()

Legge da un file presente in flash una serie di dati, che vengono memorizzati in registri reali consecutivi o in un array reale specificato.

È come la f_read_table 331(), ma viene controllato che non ci sia sbordamento dal buffer *buff* specificato.

Un'ulteriore differenza è che non è possibile iniziare da un indice qualsiasi di un array di real o strutture, ma solo dall'inizio.

La funzione deve essere utilizzata solo passando parametri corretti. Si sconsiglia di usarla normalmente.

NOTA: In caso di lettura di meno dati, la funzione riempe i rimanenti con gli ultimi letti.

Sintassi	**i32 f_read_table_s (string fileName, i32 nEntry, i32 nData, * buff [, i32 nRows=NULL])**
fileName	Nome del file da cui leggere i dati. Esso deve contenere anche la partizione della flash.
nEntry	È il numero di entry della tabella.
nData	Numero di dati per entry (max 128), cioè numero di dati per ogni riga del file. La linea comunque può essere lunga al massimo 256 caratteri.
buff	Deve essere una variabile di uno dei seguenti tipi: • array di variabili di tipo stru_cam 831 • registro reale rr o nvrr, a partire dal quale devono essere memorizzati i dati della camma letti dal file • array locale di variabili reali in cui devono essere memorizzati i dati della camma letti dal file • array di strutture che contengono solo variabili real

© 2025 Robox SpA

Sintassi	**i32 f_read_table_s (string fileName, i32 nEntry, i32 nData, * buff [, i32 nRows=NULL])**
nRows	Ritorna il numero di righe valide lette. (opzionale, default NULL) (da RTE 34.24.8)
Risultato	Rende:

Valore	Descrizione
> 0	Mancano uno o più elementi. Restituisce il numero della riga a cui manca almeno un elemento.
0	Lettura OK.
-1	Lettura OK. Altri dati nel file.
-2	Lettura OK. Meno dati nel file di quelli previsti. Tabella non corretta.
-10	Il file non esiste.
-11	Numero di dati per linea illegale.
-12	Numero di entry illegale.
-14	File vuoto.
-15	Utilizzato argomento *nRows* con RTE < 34.24.8 (viene generato allarme 9900).

Validità	Task.
Note	Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

I dati ricavati saranno poi utilizzati nelle rule per l'esecuzione delle funzioni mv_table 481() e mv_cam 451().

I dati contenuti all'interno del file devono sottostare a quanto segue:

- dimensione massima della linea: 128 caratteri
- dimensione massima della singola voce: 18

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
public stru_cam 831 cerchio[201]  ; definisco cerchio come variabile di
; viene definita in un task come pubblica e come extern nelle regole
reso = f_read_table_s 333("/FA/cerchio.tab", 201, 2, cerchio)
;dove il file caricato sulla partizione FA della flash-card è composto
; file cerchio.tab
-1 0
-0.99 0.14106736
-0.9800 0.198997487

; Prendo i dati dal file residente in /FA/ "file_read_table.txt", comp
; 1
; 2
; 3
; 4
; 5
; 6
; 7
; 8
; 9
; 0
result_table = f_read_table_s("/FA/file_read_table.txt", 10, 1, rr(100
; in questo modo ho che rr(100)=1, rr(101)=2, ... rr(109)=0
```

Funzione f_rename()

Questa funzione è utilizzata per rinominare un file presente sulla CF del controllo assi.

Sintassi	**i32 f_rename (string oldPathName, string newPathName)**
oldPathName	Nome attuale del file compreso di folder.
newPathName	Nome desiderato del file compreso di folder. Il folder di destinazione deve essere lo stesso.
Risultato	Rende: Valore: 0 — Esito positivo dell'operazione. Valore: -1 — Esito negativo.
Validità	Task.
Note	Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

```
int result_rename
string result

result_rename = f_rename(335)("/fa/old_file.txt", "/fa/new_file.txt")
if(result_rename = -1)
   result = "funzione f_rename: esito negativo, vedere report"
else
   result = "funzione f_rename: esito positivo"
endif
```

Funzione f_seek()

Questa funzione è utilizzata per spostare il puntatore di lettura/scrittura all'interno di un file.

Sintassi	**i32 f_seek (i32 offset, i32 mode, stru_filehnd stru)**
offset	Definisce la posizione del puntatore all'interno del file. Il valore fornito da offset viene interpretato dal sistema operativo in modo opportuno, a seconda del parametro *mode* specificato.
mode	È una costante che specifica la modalità di interpretazione del valore del parametro *offset*. I valori ammessi sono i seguenti:
stru	Variabile di tipo stru_filehnd(842) che contiene l'handle ottenuto tramite la funzione

Valore	Significato
0	Posizione assoluta dall'inizio del file (se *offset*=1 si punta il primo carattere del file).
1	Posizione relativa rispetto alla posizione corrente del puntatore.
2	Posizione assoluta contando all'indietro dalla fine del file (se *offset*=1 si punta l'ultimo carattere del file).

© 2025 Robox SpA

Sintassi	**i32 f_seek (i32 offset, i32 mode, stru_filehnd stru)**
	f_open 327().
Risultato	Rende 1 se l'operazione ha avuto esito positivo.
Validità	Task.
Note	Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

Esempi

Esempio di utilizzo:

```
int result_open
int result_seek
stru_filehnd 842 cf

result_open = f_open 327("/fa/autoexec.stp", "wb", 0, cf)
result_seek = f_seek 336(1, 2, cf)
; il puntatore si trova sull'ultimo carattere del file autoexec.stp
```

Funzione f_write()

Questa funzione è utilizzata per scrivere un blocco di dati in un file che sia stato aperto in modo scrittura binaria.

Sintassi	**i32 f_write (buff, i32 blockLen, i32 nBlock, stru_filehnd stru)**
buff	È una variabile locale o globale. Può essere di tipo string o un qualsiasi altro tipo purchè array. In *buff* è contenuto il blocco di dati da scrivere sequenzialmente nel file.
blockLen	Dimensione in byte di ciascun blocco che deve essere scritto.
nBlock	Numero di blocchi di dimensioni *blockLen* da scrivere.
stru	Variabile di tipo stru_filehnd 842 che contiene l'handle ottenuto tramite la funzione f_open 327().
Risultato	Rende: Valore: 0 — Esito negativo

© 2025 Robox SpA

Sintassi	**i32 f_write (buff, i32 blockLen, i32 nBlock, stru_filehnd stru)**
	Valore / Descrizione dell'operazione. <> 0 — Esito positivo.
Validità	Task.
Note	Se si utilizzano funzioni riguardanti la gestione della compact flash, si consiglia di impostare a 1 il bit 14 della variabile sys_cfg (0x00004000), per abilitare l'uso delle routine safety per la scrittura/lettura flash.

NOTA: Se utilizzo un file non vuoto, il contenuto del file precedente all'utilizzo della funzione viene cancellato.

Esempi

Esempio di utilizzo:

```
; modifico il file test.txt nella cartella /fa/, scrivendoci dentro 12
int result_open
int result_close
int result_write
stru_filehnd 842 cf

result_open = f_open 327("/fa/test.txt", "wb", 0, cf)
result_write = f_write 337("12345678", 8, 1, cf)
result_close = f_close 318(cf)
```

Funzione fall()

Funzione che permette di cogliere il fronte di discesa di una condizione.

Sintassi	**bool fall (bool cond)**
cond	Condizione di cui si vuole cogliere il fronte di discesa.
Risultato	Rende TRUE solo quando la condizione è FALSE ed era TRUE all'esecuzione precedente.
Validità	Rule, Task.
Note	La prima volta che viene eseguita, la funzione rende FALSE e memorizza lo stato attuale della condizione. Nel caso si utilizzi fall 338() in una funzione R3 utente, va evitato che i parametri della funzione stessa siano coinvolti nell'espressione di condizione. Il risultato non

© 2025 Robox SpA

Sintassi	**bool fall (bool cond)**
	sarebbe quello sperato a causa della sovrapposizione in memoria dello stato di più variabili.

Ogni funzione di fall() ha una propria memoria dello stato della condizione.

La valutazione del fronte è fatta tra lo stato attuale della condizione e la memoria relativa alla singola chiamata della funzione (es. se abbiamo allo step 1000 `if(fall(inp(1)))` e allo step 1100 `if(fall(inp(1)))`, benché facciano riferimento alla stessa condizione, la valutazione è fatta in modo distinto, ognuna in base allo stato memorizzato eseguendo step 1000 e poi step 1100).

Esempi

Esempio di utilizzo:

```
if(fall(inp(449)))
    sr(1) = "fronte di discesa"
endif

if(fall(r(1)=3))
    sr(1) = "il registro r(1) non ha più valore 3"
endif
```

Funzione fall_filt()

Funzione che permette di cogliere il fronte di discesa di una condizione impostando un filtro in spazio o tempo.

Funzionamento

Al rilevamento del fronte di discesa della condizione viene memorizzato il valore della variabile di filtro corrispondente.

Finché la differenza tra il valore attuale della variabile di filtro e quello memorizzato è inferiore alla soglia specificata la funzione rende 0, mentre quando la differenza diventa maggiore della soglia specificata la funzione rende il valore memorizzato sul fronte di discesa. Quando la condizione è TRUE, la funzione rende 0.

Sintassi	**real fall_filt (bool cond, real varFilt, real thrFilt)**
cond	Condizione da valutare per la ricerca del fronte.
varFilt	Grandezza evolvente (tipicamente tempo (tfb) o spazio ip(n)).
thrFilt	Quantità di *varFilt* in cui si deve vedere stabilmente la condizione falsa.
Risultato	Rende:

© 2025 Robox SpA

Sintassi	**real fall_filt (bool cond, real varFilt, real thrFilt)**

Valore	Condizione
0	*cond* = TRUE
	cond = FALSE ma non si è superato la soglia *thrFilt*
<> 0 (viene reso il valore di *varFilt* registrato sul fronte di discesa di *cond*)	*cond* = FALSE e si è superato *thrFilt* (valore della variabile di riferimento memorizzato su transizione).
	cond = FALSE e delta filtro è soddisfatto (valore della quota memorizzata sul fronte di discesa).

Validità	Rule, Task.
Note	Dalla prima volta che viene eseguita la funzione fino a che la condizione si mantiene bassa e non si supera la soglia il valore reso è 0. Superata tale soglia il valore reso è il primo valore fornito di *varFilt*. Nel caso si utilizzi fall_filt() in una funzione R3 utente, va evitato che i parametri della funzione stessa siano coinvolti nell' espressione di "condizione". Il risultato non sarebbe quello sperato a causa della sovrapposizione in memoria dello stato di diversi variabili.

Fornendo come variabile di filtro ad esempio ip(n) si avrà un filtro in spazio, mentre fornendo come variabile di filtro tfb si avrà un filtro in tempo (unità secondi).

Ogni funzione di fall_filt() ha la propria memoria dello stato della "condizione". La valutazione del fronte è fatta tra lo stato attuale della condizione e la memoria relativa alla singola chiamata della funzione. (es. se abbiamo allo step 1000 `if(fall_filt(inp(1), tfb, 0.5))` e allo step 1100 `if(fall(inp(1), tfb, 0.5))`, benché facciano riferimento alla stessa condizione, la valutazione è fatta in modo distinto, ognuna in base allo stato memorizzato eseguendo step 1000 e poi step 1100).

Esempi

Esempio di utilizzo:

```
reso = fall_filt(inp(1), tfb, 0.5)
```

© 2025 Robox SpA

```
; filtro temporale su inp(1)
inp1f = not(fall_filt(rise_filt563(inp(1), tfb, TimeFilter), tfb, TimeF

; calcolo della distanza tra la discesa dell'input e la posizione attu
; per gestire una uscita tra due soglie
posDw = fall_filt(inp(1), ip(1), spaceFilter)
if(posDw)
    ; siamo in basso con input. vediamo lo spazio percorso
    deltaPos = abs156(ip(1) - posDW)
    ; apri la botola per eliminare pezzo
    out(oApriBotola) = range557(deltaPos,  SogliaApriBotola, SogliaChiu
else
    out(oApriBotola) = false
endif
```

Funzione filter_lp()

Funzione di filtro digitale passa-basso. Ogni chiamata di funzione avrà il suo storage.

Sintassi	**real filter_lp (real val, real omega, real valQ)**
val	È il valore (oppure una generica espressione matematica) da filtrare.

© 2025 Robox SpA

Sintassi	**real filter_lp (real val, real omega, real valQ)**
omega	Valore della omega di taglio del filtro in radianti al secondo (omega = 2 * K_PI * frequenza di taglio (Hz)). Può essere una generica espressione matematica.
valQ	Valore del coefficiente Q del filtro (Q deve essere >= 0.75). Può essere una generica espressione matematica.
Risultato	Rende il valore filtrato.
Validità	Rule.
Note	Limitazione automatica di omega: (omega_max = 0.15 / si * 2 * K_PI) -> ossia non più del 15% del valore omega corrispondente al tempo di battuta delle rule. (es. se Freq = 200Hz -> frequenza di taglio filtro <= 30 Hz)

Esempi

Esempio di utilizzo:

```
function filtraCV()
    ; Filtriamo la velocità reale letta dall'encoder master
    cVf = filter_lp(cv(AX_C), 2.0 * K_PI * rrFreqFilCv, rrQfilt)
end_fun
```

Funzione filter_lp_st()

Funzione di filtro digitale passa-basso con storage specifico esterno, in modo da poter essere utilizzato in chiamate multiple.

Sintassi	**real filter_lp_st (real val, real omega, real valQ, real storage[2], real period, i32 flagInit)**
val	È il valore (oppure una generica espressione matematica) da filtrare.
omega	Valore della omega di taglio del filtro in radianti al secondo (omega = 2 * K_PI * frequenza di taglio (Hz)). Può essere una generica espressione matematica.
valQ	Valore del coefficiente Q del filtro (Q deve essere >= 0.75). Può essere una generica espressione matematica.
storage	Variabile storage per l'elaborazione matematica della funzione filtro.

© 2025 Robox SpA

Sintassi	**real filter_lp_st (real val, real omega, real valQ, real storage[2], real period, i32 flagInit)**
period	Intervallo di campionamento, se <= 0 si assume si.
flagInit	Variabile per la richiesta di inizializzazione del filtro; imponendo un valore diverso da 0 il filtro verrà resettato e la variabile *flagInit* verrà scritta a 0. La prima volta che viene chiamata la funzione questa variabile deve avere un valore diverso da 0.
Risultato	Rende il valore filtrato.
Validità	Rule.
Note	Limitazione automatica di omega: (omega_max = 0.15 / si * 2 * K_PI) -> ossia non più del 15% del valore omega corrispondente al tempo di battuta delle rule. (es. se Freq = 200Hz -> frequenza di taglio filtro <= 30 Hz)

I parametri sono come per la filter_lp(), ma in più consente di specificare una variabile storage adibita all'elaborazione matematica della funzione filtro.

In questo modo è possibile utilizzare i filtri anche con chiamate multiple (es. entro dei cicli for).

Esempi

Esempio di utilizzo:

```
real cv_filt[MAX_ASSI]
real store[MAX_ASSI][2]
int  flgInit[MAX_ASSI]

for(i=0, i<MAX_ASSI, i=i+1)
    cv_filt[i] = filter_lp_st(cv(i+1), omega, Q, store[i], 0, flgIn
end_for
```

Funzione filter_notch()

Funzione di filtro digitale notch. Ogni chiamata di funzione avrà il suo storage.

Sintassi	**real filter_notch (real val, real omega, real valQ)**
val	È il valore (oppure una generica espressione matematica) da filtrare.
omega	Valore della omega di taglio del filtro in radianti al secondo (omega = 2 * K_PI * frequenza di taglio (Hz)). Può essere una generica espressione matematica.

© 2025 Robox SpA

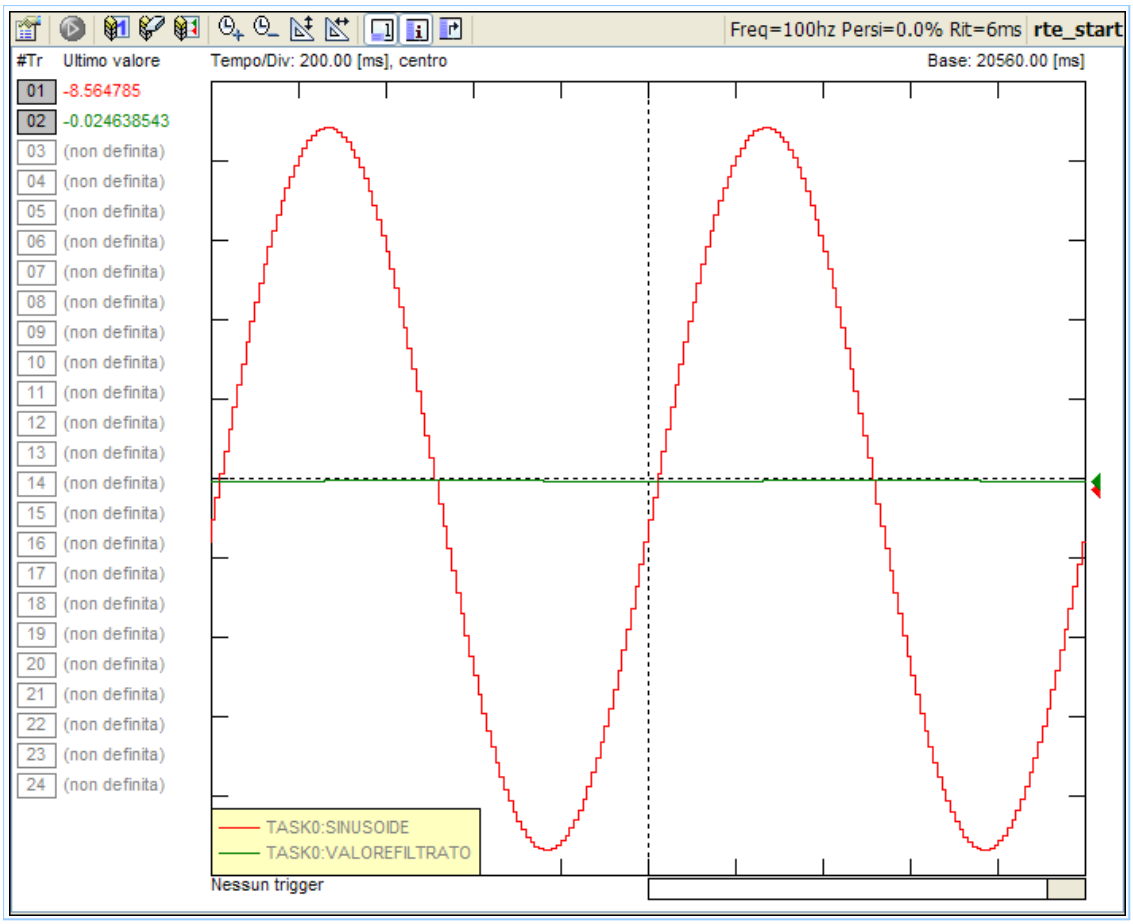
Sintassi	**real filter_notch (real val, real omega, real valQ)**
valQ	Valore del coefficiente Q del filtro (Q deve essere >= 0.75). Può essere una generica espressione matematica.
Risultato	Rende il valore filtrato.
Validità	Rule.
Note	Limitazione automatica di omega: (omega_max = 0.15 / si * 2 * K_PI) -> ossia non più del 15% del valore omega corrispondente al tempo di battuta delle rule. (es. se Freq = 200Hz -> frequenza di taglio filtro <= 30 Hz)

Esempi

Esempio di utilizzo:

```
Rule 2
Axes 1
motion
    ; Costruisco una sinusoide di ampiezza 100 e frequenza rrFreqSin
    sinusoide = 100.0 * sin(2 * K_PI * rrFreqSin * tfb)
    ; Filtriamo
    valoreFiltrato = filter_notch(sinusoide, 2 * k_pi * rrFreqNot
end_motion
; (se rrFreqSin = 1 ; rrFreqNotch = 1 ; rrQnotch = 0.8 )
```

© 2025 Robox SpA



Funzione filter_notch_st()

Funzione di filtro digitale notch con storage specifico esterno, in modo da poter essere utilizzato in chiamate multiple.

Sintassi	**real filter_notch_st (real val, real omega, real valQ, real storage[2], real period, i32 flagInit)**
val	È il valore (oppure una generica espressione matematica) da filtrare.
omega	Valore della omega di taglio del filtro in radianti al secondo (omega = 2 * K_PI[17] * frequenza di taglio (Hz)). Può essere una generica espressione matematica.
valQ	Valore del coefficiente Q del filtro (Q deve essere >= 0.75). Può essere una generica espressione matematica.
storage	Variabile storage per l'elaborazione matematica della funzione filtro.

© 2025 Robox SpA

Sintassi	**real filter_notch_st (real val, real omega, real valQ, real storage[2], real period, i32 flagInit)**
period	Intervallo di campionamento, se <= 0 si assume si.
flagInit	Variabile per la richiesta di inizializzazione del filtro; imponendo un valore diverso da 0 il filtro verrà resettato e la variabile *flagInit* verrà scritta a 0. La prima volta che viene chiamata la funzione questa variabile deve avere un valore diverso da 0.
Risultato	Rende il valore filtrato.
Validità	Rule.
Note	Limitazione automatica di omega: (omega_max = 0.15 / si * 2 * K_PI) -> ossia non più del 15% del valore omega corrispondente al tempo di battuta delle rule. (es. se Freq = 200Hz -> frequenza di taglio filtro <= 30 Hz)

I parametri sono come per la filter_notch(), ma in più consente di specificare una variabile storage adibita all'elaborazione matematica della funzione filtro.

In questo modo è possibile utilizzare i filtri anche con chiamate multiple (es. entro dei cicli for).

Esempi

Esempio di utilizzo:

```
real cv_filt[MAX_ASSI]
real store[MAX_ASSI][2]
int  flgInit[MAX_ASSI]
....
for(i=0, i<MAX_ASSI, i=i+1)
    cv_filt[i] = filter_notch_st(cv(i+1), omega, Q, store[i], 0, flg
end_for
```

Funzione first_time()

Questa funzione rende TRUE solo durante la prima esecuzione della rule in questione, dopo che è stata posta in esecuzione mediante un'istruzione group o rc(n). Ovvero, se dalla battuta precedente rc(n), con n esecutore attualmente in uso, è cambiato.

Sintassi	**i32 first_time ()**
Risultato	Rende: Valore: <> 0 — Descrizione: Se prima esecuzione della rule.

© 2025 Robox SpA

Sintassi	**i32 first_time ()**
	Valore: 0 — Descrizione: Altrimenti.
Validità	Rule.
Note	È molto utile per effettuare delle inizializzazioni.

Esempi

Esempio di utilizzo:

```
rule 1
axes (1)
motion
    if(first_time346())
        fase = 0
        rr(100) = 0
    endif
...
end_motion
end_rule
```

Funzione force()

Funzione per forzare lo stato degli IO

Sintassi	**i32 force (what, value)**
what	Oggetto da forzare. Può essere:

Tipo	Descrizione
inp(nCh)	Dove *nCh* è il canale input da forzare. Può essere un'espressione qualsiasi.
inp_w(nW)	Dove *nW* è l'indice della input word da forzare. Può essere un'espressione qualsiasi.
inp_w(nW).(iB)	Dove *nW* è l'indice della input word e *iB* è il bit della input word da forzare. Può essere un'espressione qualsiasi.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 force (what, value)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Tipo</th><th>Descrizione</th></tr>
<tr><td>out(nCh)</td><td>Dove nCh è il canale output da forzare. Può essere un'espressione qualsiasi.</td></tr>
<tr><td>out_w(nW)</td><td>Dove nW è l'indice della output word da forzare. Può essere un'espressione qualsiasi.</td></tr>
<tr><td>out_w(nW).(iB)</td><td>Dove nW è l'indice della output word e iB è il bit della output word da forzare. Può essere un'espressione qualsiasi.</td></tr>
</table>
</td></tr>
<tr><td>value</td><td>Valore da forzare.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td>0</td><td>Operazione eseguita con successo.</td></tr>
<tr><td>-1</td><td>Versione di RTE non compatibile.</td></tr>
<tr><td>-2</td><td>Indice non valido.</td></tr>
<tr><td>-3</td><td>Bit non valido.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td>Si ricorda che:
Forzare un input vuol dire che il sw vedrà lo stato dell'input forzato, indipendentemente dallo stato reale dell'input fisico
Forzare un output vuol dire che l'uscita fisica assumerà il valore forzato indipendentemente dal valore desiderato impostato da sw</td></tr>
</table>

Esempi

Esempio di utilizzo:

```
$LIT iw_rem  401                    ; indice input word remota
...
```

© 2025 Robox SpA

```
...
$LIT pbStart  inp_w(iw_rem).(4) ; start pushbutton
...
...
; simula il fronte di salita sull'input di start pushbutton
esitoF = force(pbStart, 0)
if(esitoF < 0)
    ...                 ; diagnostica per errore..
endif
dwell(0.02)
esitoF = force(pbStart, 1)
if(esitoF < 0)
    ...                 ; diagnostica per errore..
endif
dwell(0.2)
```

Esempio di utilizzo:

```
$LIT iw_base_rem  401                   ; indice input word remota
...
...
$LIT pbStop  inp((iw_base_rem-1)*16+5)  ; stop pushbutton
...
...
; simula una richiesta di stop
esitoF = force(pbStop, 0)
if(esitoF < 0)
    ...                 ; diagnostica per errore..
endif
dwell(0.1)
esitoF = release(pbStop)
if(esitoF < 0)
    ...                 ; diagnostica per errore..
endif
...
...
```

Esempio di utilizzo:

```
; forza un output
; r(1000) contiene il canale output
; r(1001) contiene il valore da forzare
esitoF = force(out(r(1000)), r(1001))
if(esitoF < 0)
    ...                 ; diagnostica per errore..
endif
...
...
```

© 2025 Robox SpA

Funzione fract()

Funzione che rende la parte frazionaria del suo argomento.

Sintassi	**real fract (real value)**
value	Valore su cui eseguire l'operazione.
Risultato	Rende la parte frazionaria dell'argomento.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
pippo = fract(12.876)
; dopo l'esecuzione pippo contiene il valore: 0.876
```

Funzione get_float()

Questa funzione restituisce un float letto da un buffer all'offset specificato.

Sintassi	**float get_float (buff, i32 offset)**
buff	Può essere un array, una struttura o un registro.
offset	Offset in numero di byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	Non viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer.

NOTA: Se l'utente desidera estrarre un float dal buffer di una struttura di tipo bbc_msguser allora deve utilizzare la funzione bcc_get_float().

Esempi

Esempio di utilizzo:

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della get_float(). Per fare ciò, impostate un valore `((0-:-127) - sizeof(my_data))` alla variabile `offset`, poi impostate un valore diverso da zero alla variabile `set_val` e verificare che la variabile `my_data` sia uguale a `set_val`.

```
u8 buffer[128]
i32 offset
float set_val
```

© 2025 Robox SpA

```
float my_data

; -------- loop begin --------
__MAIN_LOOP__

    ; range check
    if(offset >= 0 and (offset + sizeof(my_data)) < 128)
        set_float(buffer, offset, set_val)
        my_data = get_float(buffer, offset)
    endif

END_MAIN_LOOP
```

Funzione get_hw_info()

Questa istruzione permette di avere informazioni relative alle schede direttamente connesse alla CPU.

Sintassi	**i32 get_hw_info (i32 nSlot , stru_hw_info stru [, i32 info=0])**
nSlot	Numero di slot presente nel file rhw.cfg **Nota**: *nSlot* non corrisponde al numero di slot che viene assegnato ai moduli Axioline.
stru	Struttura di tipo stru_hw_info dove vengono salvate le informazioni quando viene chiamata la funzione.
info	Parametro per richiedere i valori desiderati o attuali. (opzionale, default 0) Valore: 1 — Vengono restituiti i valori attuali. Valore: 0 — Vengono restituiti i valori presenti nel file rhw.cfg. **NOTA**: I valori differiscono solo in caso di allarme 988.
Risultato	Rende: Valore: <> 0 — Operazione andata a buon fine. Valore: 0 — *nSlot* non rientra nel range previsto. Viene

© 2025 Robox SpA

Sintassi	**i32 get_hw_info (i32 nSlot , stru_hw_info stru [, i32 info=0])**
	Valore: — / Significato: generato allarme 9900.
Validità	Task.
Requisiti di sistema	RTE 34.20.3
Note	Il parametro *info* è stato implementato da RTE 34.27.2.

Esempi

Esempio di utilizzo:

```
; Parte del file rhw.cfg dove è scritto il numero dello slot di ogni m
SLOT 0 CPURBX
    PSER 1 RS232
    PSER 2 RS485
    ETH 1
    ETH 2
    CAN 1 Unused
    CAN 2 Unused
    PROFIBUS 1 Unused
SLOT 1 AUX1_URMC
SLOT 2 AXL_DRIVER
SLOT 3 AXL_F_SSDO8_3_1F
SLOT 4 AXL_F_SSDI8_4_1F
SLOT 5 AXL_F_LPSDO8_3_1F
SLOT 6 AXL_F_CNT2_INC2

; Task1
$task 1
$include "litreg"

; variables definition
STRU_HW_INFO hw
int result
int nSlot
int inpWoutW
U32 board_code_low
U32 board_code_high
string boardName[64]

nSlot = 1
; start main loop
__MAIN_LOOP__
if(rise(inp(1))) ; ADV button pressed, call function
    result = get_hw_info(nSlot, hw)
```

© 2025 Robox SpA

```
    if(result = 1) ; operation OK
        inpWoutW = hw.iwow
        board_code_low = hw.board_code_l
        board_code_high = hw.board_code_h
        boardName = hw.name
    endif
endif
END_MAIN_LOOP
```

Funzione get_i16()

Questa funzione restituisce una word i16 letta da un buffer all'offset specificato.

Sintassi	**i16 get_i16 (buff, i32 offset)**
buff	Può essere un array, una struttura o un registro.
offset	Offset in byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	Non viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer.

NOTA: Se l'utente desidera estrarre una word dal buffer di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_get_i16 220().

Esempi

Esempio di utilizzo:

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della get_i16 353(). Per fare ciò, impostate un valore `((0-:-127) - sizeof(my_data))` alla variabile `offset`, poi impostate un valore diverso da zero alla variabile `set_val` e verificare che la variabile `my_data` sia uguale a `set_val`.

```
u8 buffer[128]
i32 offset
float set_val
i16 my_data

; -------- loop begin --------
__MAIN_LOOP__

    ; range check
    if(offset >= 0 and (offset + sizeof(my_data)) < 128)
        set_i16 768(buffer, offset, set_val)
        my_data = get_i16 353(buffer, offset)
```

© 2025 Robox SpA

```
    endif

END_MAIN_LOOP
```

Funzione get_i32()

Questa funzione restituisce un i32 letto da un buffer all'offset specificato.

Sintassi	**i32 get_i32 (buff, i32 offset)**
buff	Può essere un array, una struttura o un registro.
offset	Offset in byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	Non viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer.

NOTA: Se l'utente desidera estrarre una dword dal buffer di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_get_i32 221().

Esempi

Esempio di utilizzo:

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della get_i32 354(). Per fare ciò, impostate un valore `((0-:-127) - sizeof(my_data))` alla variabile `offset`, poi impostate un valore diverso da zero alla variabile `set_val` e verificare che la variabile `my_data` sia uguale a `set_val`.

```
u8 buffer[128]
i32 offset
float set_val
i32 my_data

; -------- loop begin --------
__MAIN_LOOP__

    ; range check
    if(offset >= 0 and (offset + sizeof(my_data)) < 128)
        set_i32 769(buffer, offset, set_val)
        my_data = get_i32 354(buffer, offset)
    endif

END_MAIN_LOOP
```

© 2025 Robox SpA

Funzione get_i8()

Questa funzione restituisce una variabile i8 letta da un buffer all'offset specificato.

Sintassi	**i8 get_i8 (buff, i32 offset)**
buff	Può essere un array, una struttura o un registro.
offset	Offset in byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	Non viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer.

NOTA: Se l'utente desidera estrarre un byte dal buffer di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_get_i8 221().

Esempi

Esempio di utilizzo:

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della get_i8 355(). Per fare ciò, impostate un valore `((0-:-127) - sizeof(my_data))` alla variabile `offset`, poi impostate un valore diverso da zero alla variabile `set_val` e verificare che la variabile `my_data` sia uguale a `set_val`.

```
u8 buffer[128]
i32 offset
float set_val
i8 my_data

; -------- loop begin --------
__MAIN_LOOP__

    ; range check
    if(offset >= 0 and (offset + sizeof(my_data)) < 128)
        set_i8 771(buffer, offset, set_val)
        my_data = get_i8 355(buffer, offset)
    endif

END_MAIN_LOOP
```

Funzione get_real()

Questa funzione restituisce un real letto da un buffer all'offset specificato.

Sintassi	**real get_real (buff, i32 offset)**
buff	Può essere un array, una struttura o un registro.

© 2025 Robox SpA

Sintassi	**real get_real (buff, i32 offset)**
offset	Offset in byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	Non viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer.

NOTA: Se l'utente desidera estrarre un real dal buffer di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_get_double 219().

Esempi

Esempio di utilizzo:

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della get_real 355(). Per fare ciò, impostate un valore `((0-:-127) - sizeof(my_data))` alla variabile `offset`, poi impostate un valore diverso da zero al registro `rr(1)` e verificare che la variabile `my_data` sia uguale a `rr(1)`.

```
u8 buffer[128]
i32 offset
i8 my_data

; -------- loop begin --------
__MAIN_LOOP__

    ; range check
    if(offset >= 0 and (offset + sizeof(my_data)) < 128)
        set_real 772(buffer, offset, rr(1))
        my_data = get_real 355(buffer, offset)
    endif

END_MAIN_LOOP
```

Funzione get_u16()

Questa funzione restituisce una word u16 letta da un buffer all'offset specificato.

Sintassi	**u16 get_u16 (buff, i32 offset)**
buff	Può essere un array, una struttura o un registro.
offset	Offset in byte a cui si vuole leggere.
Risultato	Rende la word letta.
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**u16 get_u16 (buff, i32 offset)**
Note	Non viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer.

NOTA: Se l'utente desidera estrarre una word dal buffer di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_get_u16 223().

Esempi

Esempio di utilizzo:

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della get_u16 356(). Per fare ciò, impostate un valore `((0-:-127) - sizeof(my_data))` alla variabile `offset`, poi impostate un valore diverso da zero alla `inp_w 200` e verificare che la variabile `my_data` sia uguale a `inp_w 200`.

```
u8 buffer[128]
i32 offset
u16 my_data

; -------- loop begin --------
__MAIN_LOOP__

    ; range check
    if(offset >= 0 and (offset + sizeof(my_data)) < 128)
        set_u16 772(buffer, offset, inp_w(200))
        my_data = get_u16 356(buffer, offset)
    endif

END_MAIN_LOOP
```

Funzione get_u32()

Questa funzione restituisce una variabile u32 letta da un buffer all'offset specificato.

Sintassi	**u32 get_u32 (buff, i32 offset)**
buff	Può essere un array, una struttura o un registro.
offset	Offset in byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	Non viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer.

NOTA: Se l'utente desidera estrarre un u32 dal buffer di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_get_u32 224().

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della get_u16 357(). Per fare ciò, impostate un valore `((0-:-127) - sizeof(my_data))` alla variabile `offset`, poi impostate un valore diverso da zero alla `set_val` e verificare che la variabile `my_data` sia uguale a `set_val`.

```
u8 buffer[128]
i32 offset
u32 set_val
u32 my_data

; -------- loop begin --------
__MAIN_LOOP__

    ; range check
    if(offset >= 0 and (offset + sizeof(my_data)) < 128)
        set_u32 773(buffer, offset, set_val)
        my_data = get_u32 357(buffer, offset)
    endif

END_MAIN_LOOP
```

Funzione get_u8()

Questa funzione restituisce una variabile u8 letta da un buffer all'offset specificato.

Sintassi	**u8 get_u8 (buff, i32 offset)**
buff	Può essere un array, una struttura o un registro.
offset	Offset in byte a cui si vuole leggere.
Risultato	Rende il valore letto.
Validità	Rule, Task.
Note	Non viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer.

NOTA: Se l'utente desidera estrarre un byte dal buffer di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_get_u8 224().

Esempi

Esempio di utilizzo:

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della get_u8 358(). Per fare ciò, impostate un valore `((0-:-`

© 2025 Robox SpA

127) - sizeof(my_data)) alla variabile offset, poi impostate un valore diverso da zero alla set_val e verificare che la variabile my_data sia uguale a set_val.

```
u8 buffer[128]
i32 offset
u8 set_val
u8 my_data

; -------- loop begin --------
__MAIN_LOOP__

    ; range check
    if(offset >= 0 and (offset + sizeof(my_data)) < 128)
        set_u8 775(buffer, offset, set_val)
        my_data = get_u8 358(buffer, offset)
    endif

END_MAIN_LOOP
```

Funzione get_virt_ow()

Questa funzione permette di leggere lo stato reale di una output word virtuale. Con stato reale si intende lo stato programmato da un'applicazione software più un eventuale forzamento.

Sintassi	**i32 get_virt_ow (i32 outW)**
outW	Numero di out_w della quale si vuole leggere lo stato reale.
Risultato	Rende il valore dello stato reale della out_w selezionata.
Validità	Rule, Task.
Note	In caso di codice illegale viene generato l'allarme 9900.

Funzione gv_create

Funzione per creare una nuova variabile globale.

Crea le informazioni in memoria e, se necessario, anche nella memoria ritentiva.

Se richiesto, crea il file di definizione. Può creare anche un'interfaccia alla variabile globale.

Sintassi	**i32 gv_create (string var, string dataType, i32 len, string wrtPsw, i32 flag, i32 gvID, buff)**
var	Nome della variabile. Può contenere lettere, numeri, spazi e caratteri speciali. Non è case sensitive.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 gv_create (string var, string dataType, i32 len, string wrtPsw, i32 flag, i32 gvID, buff)</td></tr>
<tr><td>dataType</td><td>Descrizione del tipo di dato.
Può contenere lettere, numeri, spazi e caratteri speciali. Se omesso (""), il tipo di dato sarà inserito automaticamente.</td></tr>
<tr><td>len</td><td>Dimensione in byte della variabile globale.</td></tr>
<tr><td>wrtPsw</td><td>Password per l'accesso in scrittura. Se omessa (""), la password non viene utilizzata.</td></tr>
<tr><td>flag</td><td>
<table>
<tr><th>Valore HEX</th><th>Descrizione</th></tr>
<tr><td>0x0001</td><td>Variabile persistente.</td></tr>
<tr><td>0x0002</td><td>(Solo se definita come variabile persistente)
Sovrascrive il valore di una variabile persistente non linkata con lo stesso nome e dimensione di quella nuova.</td></tr>
<tr><td>0x0004</td><td>(Solo se definita come variabile non persistente)
Disabilita la creazione del file di configurazione.</td></tr>
<tr><td>0x0008</td><td>(Solo se definita come variabile non persistente)
Disabilita i dati nel file di configurazione.</td></tr>
<tr><td>0x0010</td><td>(Solo se viene utilizzata una password)
Abilita l'accesso in scrittura solo alla prima interfaccia linkata con password corretta.</td></tr>
</table>
</td></tr>
<tr><td>gvID</td><td>Variabile locale nella quale la funzione inserisce l'ID relativo all'interfaccia alla variabile globale. In caso di errore la funzione inserisce 0.</td></tr>
<tr><td>buff</td><td>Buffer locale usato come cache connessa all'ID (se presente).</td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 gv_create (string var, string dataType, i32 len, string wrtPsw, i32 flag, i32 gvID, buff)</td></tr>
<tr><td></td><td>Viene usato per aggiornare le informazioni della variabile globale durante le operazioni di lettura e scrittura. Alla chiamata della funzione, il suo valore indica il valore di inizializzazione della variabile globale. Deve necessariamente avere la stessa dimensione della variabile globale.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0</td><td>Operazione eseguita con successo.</td></tr>
<tr><td>-1</td><td>Nome o dimensione NULL.</td></tr>
<tr><td>-2</td><td>Gestore della variabile globale non abilitato.</td></tr>
<tr><td>-3</td><td>Variabile già esistente (o stessa key).</td></tr>
<tr><td>-4</td><td>Variabile persistente già esistente (o stessa key) o memoria persistente piena o tabella di hash piena.</td></tr>
<tr><td>-5</td><td>Problema durante la creazione del file di definizione.</td></tr>
<tr><td>-6</td><td>La dimensione della cache è minore della dimensione della variabile globale.
Viene generato l'allarme 9900.</td></tr>
</table></td></tr>
<tr><td>Validità</td><td>Task.</td></tr>
<tr><td>Note</td><td></td></tr>
</table>

Esempi

Esempio di utilizzo:

Creo 2 GV:

- global_1, attributo i32, dimensione 4 byte, no password, persistente, nessun valore inizializzato

© 2025 Robox SpA

- global_2, attributo non definito, dimensione 2 byte, password 123, solo la prima interfaccia connessa ha accesso in scrittura, valore di inizializzazione 5

```
loc_var_1 = 0
loc_var_2 = 5
result = gv_create(359)("global_1", "INT", 4, "", 0x1, gvID_1, loc_var_1)
result = gv_create("GLOBAL_2", "", 2, "123", 0x10, gvID_2, loc_var_2)
```

Usando `gv_info` da shell:

```
gv_info
Generic device command [GV_INFO]
  1   NAME<GLOBAL_2>  DATATYPE<BYTEARRAY_2>  SIZE<2>  ATTR.<Volatile - PSW on Write - Only one can write>
  2   NAME<global_1>  DATATYPE<INT>  SIZE<4>  ATTR.<Persistent>
2 GLOBAL VARIABLEs found
```

Funzione gv_datatype

Legge il tipo della variabile globale passato l'ID.

Sintassi	**string gv_datatype (i32 gvID)**
gvID	ID della variabile globale (ottenuta da gv_link(368) o gv_create(359)).
Risultato	Rende: Valore / Significato "Unhandled" / Gestore non abilitato. "Unconnected" / Interfaccia non valida. "*dataType*" / Il tipo di dato della variabile globale.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; usando i dati dell' Esempio utilizzo variabili globali
...
sr(2) = gv_datatype(362)(GVInterf_sum)    ; ottiene il tipo di dato della
```

Funzione gv_datatype_byname

Legge il tipo di dato della variabile globale.

Sintassi	**string gv_datatype_byname (string var)**
var	Nome della variabile.

© 2025 Robox SpA

Sintassi	**string gv_datatype_byname (string var)**
	Può contenere lettere, numeri, spazi e caratteri speciali. Non è case sensitive.
Risultato	Rende:

Valore	Significato
""	La variabile non esiste.
"*dataType*"	Il tipo di dato della variabile globale.

Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; usando i dati dell' Esempio utilizzo variabili globali
 ...
sr(2) = gv_datatype_byname("Global structValori raise")    ; ottiene
```

Funzione gv_destroy

Funzione che rimuove la variabile globale e cancella il file di configurazione.

Sintassi	**bool gv_destroy (i32 gvID)**
gvID	ID della variabile globale (ottenuta da gv_link o gv_create).
Risultato	Rende:

Valore	Significato
TRUE	Operazione completata con successo.
FALSE	Problemi durante l'operazione.

Validità	Task.
Note	

Esempi

Esempio di utilizzo:

© 2025 Robox SpA

```
; usando i dati dell' Esempio utilizzo variabili globali
...
esitoGV = gv_destroy(GVInterf_sum)    ; rimuove la connessione. Acces
if(esitoGV = false)
    alarm_set(5000, 0, 11, esitoGV)
endif
```

Funzione gv_exist

Questa funzione controlla che la stringa specificata combaci con il nome di una variabile globale definita.

Sintassi	**bool gv_exist (string var)**
var	Nome della variabile. Può contenere lettere, numeri, spazi e caratteri speciali. Non è case sensitive.
Risultato	Rende: Valore: TRUE — Significato: Variabile trovata. Valore: FALSE — Significato: Variabile non trovata.
Validità	Task.
Note	

Esempi

Esempio di utilizzo:

```
; usando i dati dell' Esempio utilizzo variabili globali
 ...
esitoGV = gv_exist("global_1")    ; verifica se esiste una GV che si
```

Funzione gv_flags

Questa funzione ritorna il flag della variabile globale.

Sintassi	**i32 gv_flags (i32 gvID)**
gvID	ID della variabile globale (ottenuta da gv_link o gv_create).
Risultato	Rende:

© 2025 Robox SpA

Sintassi	**i32 gv_flags (i32 gvID)**

Bit	Valore Hex	Significato
	Se = 0	La variabile non esiste.
0	0x00000001	Variabile persistente (R/W).
1	0x00000002	Sovrascrive il valore delle GV persistenti quando si crea la GV da istruzione (R/W)
2	0x00000004	Disabilita la creazione del file di configurazione (se non persistente) (R/W).
3	0x00000008	Disabilita i dati nel file di configurazione (se non persistente) (R/W).
4	0x00000010	Abilita l'accesso in scrittura solo alla prima interfaccia linkata con password corretta (R/W).
16	0x00010000	GV linkata correttamente alla componente ritentiva.
17	0x00020000	GV da linkare alla componente ritentiva.
18	0x00040000	Accesso in scrittura.
24	0x01000000	Uso interno - name

© 2025 Robox SpA

Sintassi	**i32 gv_flags (i32 gvID)**

Bit	Valore Hex	Significato
25	0x02000000	Uso interno - data
26	0x04000000	Uso interno - dataType
27	0x08000000	Uso interno - wrtPsw

Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; usando i dati dell' Esempio utilizzo variabili globali
...
r(2) = gv_flags(GVInterf_sum)    ; ritorna il flag di configurazione
```

Funzione gv_flags_byname

Questa funzione ritorna il flag della variabile globale.

Sintassi	**i32 gv_flags_byname (string var)**
var	Nome della variabile. Può contenere lettere, numeri, spazi e caratteri speciali. Non è case sensitive.
Risultato	Rende:

Bit	Valore Hex	Significato
	Se = 0	La variabile non esiste.
0	0x00000001	Variabile persistente (R/W).
1	0x00000002	Sovrascrive il valore delle GV persistenti quando si crea la GV da istruzione (R/W)

© 2025 Robox SpA

Sintassi	**i32 gv_flags_byname (string var)**

Bit	Valore Hex	Significato
2	0x00000004	Disabilita la creazione del file di configurazione (se non persistente) (R/W).
3	0x00000008	Disabilita i dati nel file di configurazione (se non persistente) (R/W).
4	0x00000010	Abilita l'accesso in scrittura solo alla prima interfaccia linkata con password corretta (R/W).
16	0x00010000	GV linkata correttamente alla componente ritentiva.
17	0x00020000	GV da linkare alla componente ritentiva.
18	0x00040000	Accesso in scrittura.
24	0x01000000	Uso interno - name
25	0x02000000	Uso interno - data
26	0x04000000	Uso interno - dataType
27	0x08000000	Uso interno - wrtPsw

Validità	Rule, Task.
Note	

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
; usando i dati dell' Esempio utilizzo variabili globali
...
r(2) = gv_flags_byname("Global Real 1")    ; ritorna il flag di confi
```

Funzione gv_link

Funzione per linkare una variabile globale.

Crea un'interfaccia alla variabile. La variabile locale specificata è strettamente associata all'interfaccia.

Sintassi	**i32 gv_link (string var, i32 len, string wrtPsw, i32 gvID, buff)**
var	Nome della variabile. Può contenere lettere, numeri, spazi e caratteri speciali. Non è case sensitive.
len	Dimensione in byte della variabile globale.
wrtPsw	Password per l'accesso in scrittura. Se omessa (""), la password non viene utilizzata.
gvID	Variabile locale nella quale la funzione inserisce l'ID relativo all'interfaccia alla variabile globale. In caso di errore la funzione inserisce 0.
buff	Buffer locale usato come cache connessa all'ID (se presente). Viene usato per aggiornare le informazioni della variabile globale durante le operazioni di lettura e scrittura. Alla chiamata della funzione, il suo valore indica il valore di inizializzazione della variabile globale. Deve necessariamente avere la stessa dimensione della variabile globale.
Risultato	Rende: Valore / Significato 0 – Operazione conclusa con successo. 1 – Link creato, ma la GV è in lock, per cui la 'locVar' non è aggiornata. 2 – Link creato, ma non si ha accesso in scrittura (password sbagliata o non è il primo link).

© 2025 Robox SpA

Sintassi	**i32 gv_link (string var, i32 len, string wrtPsw, i32 gvID, buff)**

Valore	Significato
3	Link creato, ma la GV è in lock, per cui la 'locVar' non è aggiornata e non si ha accesso in scrittura (password sbagliata o non è il primo link).
-1	La variabile non esiste.
-2	Dimensione non valida.
-3	Password non valida.
-10	Gestore non abilitato.
-11	nome NULL.
-12	locVar NULL.
-13	gvIDrule NULL.
-14	dimensione NULL.
-15	La dimensione della cache è minore della dimensione della variabile. Viene generato l'allarme 9900.

Validità	Rule, Task.
Note	

Funzione gv_lock

Funzione che blocca l'accesso alla variabile globale.

L'accesso alla GV non è permesso da nessun'altra interfaccia.

Sintassi	**bool gv_lock (i32 gvID)**
gvID	ID della variabile globale (ottenuta da gv_link 368 o gv_create 359).
Risultato	Rende:

Valore	Significato
TRUE	Variabile bloccata.

© 2025 Robox SpA

Sintassi	**bool gv_lock (i32 gvID)**
	Valore: FALSE — Significato: Variabile già bloccata (da un'altra interfaccia o da se stessa).
Validità	Rule, Task.
Note	

Funzione gv_name

Funzione che ritorna il nome della variabile globale.

Sintassi	**string gv_name (i32 gvID)**
gvID	ID della variabile globale (ottenuta da gv_link 368 o gv_create 359).
Risultato	Rende: "Unhandled" — Gestore non abilitato. "Unconnected" — Interfaccia non valida. "*vName*" — Il nome della variabile globale.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; usando i dati dell' Esempio utilizzo variabili globali
...
sr(1) = gv_name 370(GVInterf_sum)    ; ottiene il nome della GV... in SR
```

Funzione gv_read

Questa funzione aggiorna il dato della variabile locale (fornita dalla gv_link 368 o gv_create 359) associata tramite l'interfaccia, leggendo dalla GV.

Sintassi	**i32 gv_read (i32 gvID)**
gvID	ID della variabile globale (ottenuta da gv_link 368 o gv_create 359).

© 2025 Robox SpA

Sintassi	**i32 gv_read (i32 gvID)**
Risultato	Rende: Valore: Significato 0: Operazione conclusa con successo. -1: Gestore non abilitato. -2: *gvID* non valido (interfaccia inesistente). -3: La GV è bloccata.
Validità	Rule, Task.
Note	

Funzione gv_read_partial

Questa funzione aggiorna parzialmente il dato della variabile locale (fornita dalla GV_LINK 368 o GV_CREATE 359) associata tramite l'interfaccia, leggendo dalla GV

Sintassi	**i32 gv_read_partial (i32 gvID, u32 offset, u32 dataLen)**
gvID	ID della variabile globale (ottenuta da gv_link 368 o gv_create 359).
offset	Offset in byte nella variabile (da 1 alla dimensione di GV-1).
dataLen	Lunghezza del dato da aggiornare (da 1 alla dimensione di GV).
Risultato	Rende: Valore: Significato 0: Operazione conclusa con successo. -1: Gestore non abilitato. -2: *gvID* non valido (interfaccia inesistente). -3: La GV è bloccata.
Validità	Rule, Task.
Note	

© 2025 Robox SpA

Funzione gv_size

Questa funzione ritorna la dimensione della variabile globale.

Sintassi	**i32 gv_size (i32 gvID)**
gvID	ID della variabile globale (ottenuta da gv_link 368 o gv_create 359).
Risultato	Rende: Valore: 0 — Gestore variabili globali non abilitato o interfaccia non valida. Valore: *size* — Dimensione della GV.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; usando i dati dell' Esempio utilizzo variabili globali
...
r(2) = gv_size 372(GVInterf_real)    ; ottiene la dimensione della GV, i
```

Funzione gv_size_byname

Questa funzione ritorna la dimensione della variabile globale.

Sintassi	**i32 gv_size_byname (string var)**
var	Nome della variabile. Può contenere lettere, numeri, spazi e caratteri speciali. Non è case sensitive.
Risultato	Rende: Valore: 0 — La variabile non esiste. Valore: *size* — Dimensione della GV.
Validità	Rule, Task.
Note	

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
; usando i dati dell' Esempio utilizzo variabili globali
...
r(2) = gv_size_byname[372]("Global Real 1")    ; ottiene la dimensione de
```

Funzione gv_unlink

Questa funzione rimuove la connessione dell'interfaccia con la variabile globale.

Sintassi	**bool gv_unlink (i32 gvID)**
gvID	ID della variabile globale (ottenuta da gv_link[368] o gv_create[359]).
Risultato	Rende:

Valore	Significato
TRUE	Connessione rimossa.
FALSE	Operazione fallita.

Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; usando i dati dell' Esempio utilizzo variabili globali
...
esitoGV = gv_unlink[373](GVInterf_sum)   ; rimuove la connessione. Le succ
```

Funzione gv_unlock

Funzione che sblocca l'accesso alla variabile globale.

L'accesso alla GV ora è permesso a tutte le altre interfacce.

Sintassi	**bool gv_unlock (i32 gvID)**
gvID	ID della variabile globale (ottenuta da gv_link[368] o gv_create[359]).
Risultato	Rende:

Valore	Significato
TRUE	Variabile sbloccata.
FALSE	Variabile già sbloccata (o bloccata da un'altra interfaccia).

© 2025 Robox SpA

Sintassi	**bool gv_unlock (i32 gvID)**
Validità	Rule, Task.
Note	

Funzione gv_update_file

Questa funzione aggiorna il file di configurazione delle variabili globali.

Sintassi	**bool gv_update_file (i32 gvID)**
gvID	ID della variabile globale (ottenuta da gv_link 368 o gv_create 359).
Risultato	Rende: Valore / Significato TRUE: Operazione eseguita con successo. FALSE: L'operazione ha avuto problemi.
Validità	Task.
Note	

Esempi

Esempio di utilizzo:

```
; usando i dati dell' Esempio utilizzo variabili globali
...
esitoGV = gv_update_file 374(GVInterf_raise)    ; aggiorna il file di co
if(esitoGV = false)
    alarm_set 175(5000, 0, 12, esitoGV)
endif
```

Funzione gv_write

Questa funzione aggiorna il valore della variabile globale con il valore della variabile locale associata tramite l'interfaccia ('*locVar*' fornita dalla gv_link 368 o gv_create 359).

Sintassi	**i32 gv_write (i32 gvID)**
gvID	ID della variabile globale (ottenuta da gv_link 368 o gv_create 359).
Risultato	Rende:

© 2025 Robox SpA

Sintassi	**i32 gv_write (i32 gvID)**

Valore	Significato
0	Operazione conclusa con successo.
-1	Gestore non abilitato.
-2	*gvID* non valido (interfaccia inesistente).
-3	La GV è bloccata.
-4	Accesso in scrittura bloccato da password.

Validità	Rule, Task.
Note	

Funzione gv_write_partial

Questa funzione aggiorna parzialmente il valore della variabile globale con il valore della variabile locale associata tramite l'interfaccia ('*locVar*' fornita dalla gv_link 368 o gv_create 359).

Sintassi	**i32 gv_write_partial (i32 gvID, u32 offset, u32 dataLen)**
gvID	ID della variabile globale (ottenuta da gv_link 368 o gv_create 359).
offset	Offset in byte nella variabile (da 1 alla dimensione di GV-1).
dataLen	Lunghezza del dato da aggiornare (da 1 alla dimensione di GV).
Risultato	Rende:

Valore	Significato
0	Operazione conclusa con successo.
-1	Gestore non abilitato.
-2	*gvID* non valido (interfaccia inesistente).
-3	La GV è bloccata.
-4	Accesso in scrittura bloccato da password.

© 2025 Robox SpA

Sintassi	**i32 gv_write_partial (i32 gvID, u32 offset, u32 dataLen)**
	Valore: -5 — Significato: Offset o lunghezza del dato non validi.
Validità	Rule, Task.
Note	

Funzione in_range()

Funzione che restituisce il valore entrante opportunamente limitato all' interno del range specificato.

Sintassi	**real in_range (real value, real min, real max)**
value	Grandezza reale da valutare (può essere un'espressione reale generica).
min	Valore di limite inferiore (può essere un'espressione reale generica).
max	Valore di limite superiore (può essere un'espressione reale generica).
Risultato	Rende: Condizione: *value* > *max* — Valore: *max* Condizione: *value* < *min* — Valore: *min* Condizione: Altrimenti. — Valore: *value*
Validità	Rule, Task.
Note	Con il compilatore RC3E è bene specificare la castizzazione del tipo di dati in ingresso onde evitare warning in fase di compilazione. (es. `in_range((real) rrParPuntiCammaLMot, (real)2.0, (real)max_punti_camma_lmot))`

Esempi

Esempio di utilizzo:

```
lenObj = in_range(rr(1), 10, 100)
lenObj = in_range(rr(1), rr(2), max_str(1) - ip(1))
```

© 2025 Robox SpA

Funzione inc()

Incrementa la variabile delle grandezza specificata e rende inoltre il suo nuovo contenuto.

Sintassi	**real inc (real var [, real value=1])**
var	Valore da incrementare.
value	Valore di incremento (opzionale, default 1).
Risultato	Rende il nuovo contenuto di *var* (valore incrementato: *var*+*value*).
Validità	Rule, Task.
Note	Il risultato può anche essere omesso.

Esempi

Esempio di utilizzo:

```
if(inp(1))
    inc(rr(1), 10)
else
    dec(rr(1),10)
endif

real count1, count2
if(inp(1))
    count1 = inc(rr(1),10)
else
    count2 = dec(rr(1),10)
endif

for(i=8, i > 0, inc(i)) ; corrisponde a for(i=8, i>0, i=i+1)

appo = inc(pluto, 10)
; corrisponde a:
; pluto = pluto +10
; appo  = pluto

inc(r(1))
; corrisponde a r(1) = r(1) + 1
```

Funzione index_of()

Funzione che restituisce l'indice della risorsa passata come argomento.

La funzione è in grado di distinguere associazioni a:

r(index), rr(index), sr(index), nvr(index), nvrr(index), nvsr(index), inp(index), out(index), inp_w(index), out_w(index).

Sintassi	**i32 index_of (define)**
define	È il simbolico associato alla variabile.

© 2025 Robox SpA

Sintassi	**i32 index_of (define)**
Risultato	Rende l'indice dell'argomento passato.
Validità	Rule, Task.
Note	

La funzione è di fatto una macro del compilatore la quale, avendo un argomento r/rr/sr/nvr/nvrr/nvsr/inp/out/inp_w/out_w(index), restituisce compilation-time <index> al posto della index_of(...).

Esempi

Esempio di utilizzo:

```
lit rrRicettaAppo  rr(345)
lit iAriaOk        inp(264)
...
...
idxPar = index_of(rrRicettaAppo)   ; idxPar assume il valore 345
idxInp = index_of(iAriaOk)         ; idxInp assume il valore 264
```

Funzione ipaddr_to_str()

Conversione da indirizzo IP rappresentato in un intero a stringa.

Sintassi	**i32 ipaddr_to_str (u32 addr, string sAddr)**
addr	Parametro contenente l'indirizzo da convertire.
sAddr	Parametro in cui la funzione scrive l'indirizzo convertito.
Risultato	Rende: Valore: <> 0 — Significato: Operazione conclusa con successo. Valore: 0 — Significato: *sAddr* è 0 o la dimensione della stringa è troppo piccola.
Validità	Rule, Task.
Note	

La stringa ha formato: "xxx.xxx.xxx.xxx".

L'intero è di tipo u32 con formato: u32 *addr* = 0xaabbccdd; aa, bb, cc e dd rappresentano le 4 componenti dell'indirizzo IP espresse in formato esadecimale.

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

```
int reso
string addrSr
u32 addrNum
reso = ipaddr_to_str(addrNum, addrSr)
; se addrNum= 0xc0a8010a --> addrSr = "192.168.1.10"
; se addrNum= 0x0a001110 --> addrSr = "10.0.17.16"
```

Funzione is_alarm_par_present()

Questa funzione permette di verificare se nello stack allarmi (32 posizioni) è presente un allarme specificando anche il valore di uno o più parametri.

Sintassi	**i32 is_alarm_par_present (i32 nAl, i32 nAx, u32 parMask, real par1 [, real par2=0] [, real par3=0])**
nAl	Numero allarme del quale verificare la presenza (1-:-9999).
nAx	Numero asse in allarme (1-:-32) o 0 se allarme non di asse.
parMask	Maschera per indicare quali parametri sono interessati:
par1	Valore del primo parametro dell'allarme.
par2	Valore del secondo parametro dell'allarme. (opzionale, default 0)
par3	Valore del terzo parametro dell'allarme. (opzionale, default 0)

Valore	Descrizione
0x0	Nessuno. Utilizzare is_alarm_present()
0x1	par1
0x2	par2
0x3	par1 e par2
0x4	par3
0x5	par1 e par3
0x6	par2 e par3
0x7	par1, par2 e par3

© 2025 Robox SpA

Sintassi	**i32 is_alarm_par_present (i32 nAl, i32 nAx, u32 parMask, real par1 [, real par2=0] [, real par3=0])**
Risultato	Rende: Valore: Descrizione <> 0: Allarme presente. 0: Altrimenti.
Validità	Rule, Task.
Note	Per i parametri intermedi che non sono stati specificati nella ricerca, si deve comunque inserire necessariamente un valore (valore indifferente, anche diverso dal valore reale). Da RTE 34.30.0 se viene passato un parametro che non era previsto dall'allarme, viene comunque considerata come se fosse una diversità (reso 0).

Esempi

Esempio di utilizzo:

==========

Nel file allarmi utente .txt:

```
1010 37 10 0x32000000 "Allarme grave: %d, %d, %d" -4 -4 -4  ;
ALM_MAJOR_TEST
```

==========

Nel file litconst.i3:

```
lit ALM_MAJOR_TEST      1010
```

==========

Nel task:

```
int result, resOK, resErr
...

if(rise(rGenerateAlarm))
    result = alarm_set(ALM_MAJOR_TEST, 0, 1, 2, 3) ; par1=1, par2=2,
    ; Nello stack si avrà: "Allarme grave: 1, 2, 3"
endif

if(result)
; Risultati corretti (ritorna 1)
    resOK = is_alarm_par_present(ALM_MAJOR_TEST, 0, 0x1, 1)    ; par1
    resOK = is_alarm_par_present(ALM_MAJOR_TEST, 0, 0x2, 0, 2)       ;
    resOK = is_alarm_par_present(ALM_MAJOR_TEST, 0, 0x3, 1, 2, 0)   ; p
    resOK = is_alarm_par_present(ALM_MAJOR_TEST, 0, 0x4, 0, 0, 3)   ; p
```

© 2025 Robox SpA

```
    resOK = is_alarm_par_present(ALM_MAJOR_TEST, 0, 0x5, 1, 0, 3)  ; p
    resOK = is_alarm_par_present(ALM_MAJOR_TEST, 0, 0x6, 0, 2, 3)  ; p
    resOK = is_alarm_par_present(ALM_MAJOR_TEST, 0, 0x7, 1, 2, 3)  ; p
    ; si può inserire qualsiasi valore per i parametri intermedi che n
    resOK = is_alarm_par_present(ALM_MAJOR_TEST, 0, 2, -1, 2)
    resOK = is_alarm_par_present(ALM_MAJOR_TEST, 0, 4, 123, 3, 3)
    resOK = is_alarm_par_present(ALM_MAJOR_TEST, 0, 2, 0, 2)

; Risultati errati (ritorna 0)
    resErr = is_alarm_par_present(ALM_MAJOR_TEST, 0, 0x1, -1)  ; par1
    resErr = is_alarm_par_present(ALM_MAJOR_TEST, 0, 0x2,  2)  ; error
endif
...
```

Funzione is_alarm_par_present_us()

Questa funzione permette di verificare se nello stack allarmi (32 posizioni) è presente un allarme specificando anche il valore di uno o più parametri USV.

Sintassi	**bool is_alarm_par_present_us (i32 nAl, i32 nAx, u32 parMask [, string s1=0] [, string s2=0] [, string s3=0])**
nAl	Numero allarme del quale verificare la presenza (0 o 1-:-9999).
nAx	Numero asse in allarme (1-:-32) o 0 se allarme non di asse.
parMask	Maschera per indicare quali parametri sono interessati:

Valore	Descrizione
0x0	Nessuno parametro USV.
0x1	s1
0x2	s2
0x3	s1 e s2
0x4	s3
0x5	s1 e s3
0x6	s2 e s3
0x7	s1, s2 e s3

s1	Stringa utente da usare come primo parametro dell'allarme. (opzionale, default 0)

© 2025 Robox SpA

Sintassi	**bool is_alarm_par_present_us (i32 nAl, i32 nAx, u32 parMask [, string s1=0] [, string s2=0] [, string s3=0])**
s2	Stringa utente da usare come secondo parametro dell'allarme. (opzionale, default 0)
s3	Stringa utente da usare come terzo parametro dell'allarme. (opzionale, default 0)
Risultato	Rende: Valore: TRUE — Allarme presente Valore: FALSE — • allarme non presente • numero allarme illegale • l'allarme non ha USV e parMask diversa da 0
Validità	Rule, Task.
Requisiti di sistema	Da RTE 34.30.0.
Note	

Diagnostica

- Genera 9986 wrong test_alarm presence funct. on alarm:xxx se l'allarme non ha una USV definita e parMask diversa da 0 (dove xxx numero di allarme)

Esempi

Esempio di utilizzo:

```
string s1
string s2
int result
...
; testa presenza allarme maggiore di asse specificando il parametro 1
str_format (s1, "Broken Brake")
result = is_alarm_par_presence_us(1101, 3, 1, s1)
...
...
; testa presenza warning specificando il parametro 1
str_format (s1, "Emergency Pressed")
result = is_alarm_par_presence_us(4101, 0, 1, s1)
...
...
; testa presenza warning specificando il parametro 2
; per il parametro 1 è inserito 0 (non se ne tiene conto, ma è obbliga
```

© 2025 Robox SpA

```
str_format (s2, "end of line")
result = is_alarm_par_presence_us(4101, 0, 2, 0, s2)

; definizione allarmi in file utente
1101 -1 14 0x22001000 "Fault <%s>" 17 0 0
4101 53 24 0x22001000 "Warning <%s:%s>" 17 17 0
```

Funzione is_alarm_present()

Questa funzione permette di verificare se nello stack allarmi (32 posizioni) è presente un allarme.

Sintassi	**i32 is_alarm_present ([i32 nAl=0] [, i32 nAx=0])**
nAl	Numero allarme del quale verificare la presenza (1-:-9999). (opzionale, default 0)
nAx	Numero asse in allarme (1-:-32) o 0 se allarme non di asse. (opzionale, default 0)
Risultato	Rende: Valore: <> 0 — Allarme presente. Valore: 0 — Altrimenti.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
int nAl, nAx, isAlPres
u32 parMask
real pAl1, pAl2, pAl3
...
...
...
if(is_alarm_present 383(11, 1))
    sr(1) = "axes 1 in following error"
```

© 2025 Robox SpA

Funzione is_forced()

Funzione per controllare se lo stato degli I/O è forzato.

<table>
<tr><td>Sintassi</td><td>i32 is_forced (what [, i16|i32|u16|u32 sts=NULL [, i32|u32 mask=NULL]])</td></tr>
<tr><td>what</td><td>I/O da controllare. Può essere:
<table>
<tr><th>Tipo</th><th>Descrizione</th></tr>
<tr><td>inp(nCh)</td><td>Dove nCh è il canale input. Può essere un'espressione qualsiasi.</td></tr>
<tr><td>inp_w(nW)</td><td>Dove nW è l'indice della input word. Può essere un'espressione qualsiasi.</td></tr>
<tr><td>inp_w(nW).(iB)</td><td>Dove nW è l'indice della input word e iB è il bit della input word. Può essere un'espressione qualsiasi.</td></tr>
<tr><td>out(nCh)</td><td>Dove nCh è il canale output. Può essere un'espressione qualsiasi.</td></tr>
<tr><td>out_w(nW)</td><td>Dove nW è l'indice della output word. Può essere un'espressione qualsiasi.</td></tr>
<tr><td>out_w(nW).(iB)</td><td>Dove nW è l'indice della output word e iB è il bit della output word. Può essere un'espressione qualsiasi.</td></tr>
</table></td></tr>
<tr><td>sts</td><td>La funzione scriverà in sts lo stato di forzatura dei bit della word, ovvero:
il bit X di sts è 0 se il bit X della word è forzato a FALSE o non è forzato
il bit X di sts è 1 se il bit X della word è forzato a TRUE
NOTA: Utilizzando sts come i32 per una word che viene considerata i16, verrà fatta l'estensione di segno sui 32 bit.
(opzionale, default NULL)</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 is_forced (what [, i16|i32|u16|u32 sts=NULL [, i32|u32 mask=NULL]])**
mask	La funzione scriverà in *mask* la maschera dei bit della word forzati, ovvero: • il bit X di *mask* è 0 se il bit X della word non è forzato • il bit X di *mask* è 1 se il bit X della word è forzato (sia a TRUE sia a FALSE) Deve essere u32 se *sts* è i32 o u32, oppure u16 se *sts* è i16 o u16. (opzionale, default NULL)
Risultato	Rende: Valore | Descrizione 1 | Entità forzata. 0 | Entità non forzata. -1 | Versione RTE non compatibile. -2 | Indice non valido. -3 | Bit non valido.
Validità	Rule, Task.
Note	Si ricorda che: • forzare un input vuol dire che il sw vedrà lo stato dell'input forzato, indipendentemente dallo stato reale dell'input fisico • forzare un output vuol dire che l'uscita fisica assumerà il valore forzato indipendentemente dal valore desiderato impostato da sw

Esempi

Esempio di utilizzo:

```
int result
u16 sts
u16 mask
....
result = release(560)(inp_w(401))
result = force(347)(inp_w(401).0, 1)   ; forza inp_w 401.0 a TRUE
result = force(inp_w(401).4, 0)   ; forza inp_w 401.4 a FALSE
result = force(inp_w(401).5, 1)   ; forza inp_w 401.5 a TRUE
result = force(inp_w(401).12, 0)  ; forza inp_w 401.12 a FALSE
```

© 2025 Robox SpA

```
result = is_forced(inp_w(401), sts, mask)
; ottengo:
;    result = 1,    inp_w forzata
;    sts = 0x0021,  bit 0/bit 5 forzati a TRUE
;    mask = 0x1031, bit 0/bit 4/bit 5/bit 12 forzati
```

Esempio di utilizzo:

```
$LIT iw_rem   401                    ; indice input word remota
$LIT pbStart  inp_w(iw_rem).(4)      ; start pushbutton
$LIT pbStop   inp((iw_rem-1)*16+5)   ; stop pushbutton
int result
int result2
int result3
...
...
if(is_forced(pbStart) > 0)
    result = release(pbStart)
    if(result < 0)
        ...    ; diagnostica per errore
    endif
endif

result2 = is_forced(pbStop)
if(result2 < 0)
    ...    ; diagnostica per errore
endif

result3 = is_forced(out(r(1000)))   ; r(1000) contiene il canale di out
if(result3 < 0)
    ...    ; diagnostica per errore
endif
```

Funzione kin_conv()

Funzione che calcola i valori di accelerazione e di jerk che consentono, partendo da fermo, di raggiungere la velocità *spe* nel tempo *tAcc*.

Sintassi	**i32 kin_conv (real spe, real tAcc, real pJerk, real rAcc, real rJerk [, real maxAcc=0.0])**
spe	Valore di velocità da raggiungere partendo da 0. Viene usato il suo valore assoluto [unit/sec].
tAcc	Tempo per passare da 0 a *spe* [sec].
pJerk	Valore compreso tra 0 e 1 che stabilisce quanto jerk applicare al tratto di accelerazione:

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 kin_conv (real spe, real tAcc, real pJerk, real rAcc, real rJerk [, real maxAcc=0.0])</td></tr>
<tr><td></td><td>• 0, accelerazione brusca a gradino (jerk non utilizzato)
• 1, accelerazione dolce a triangolo (tratto completamente jerkato - rampa di velocità a S)</td></tr>
<tr><td>rAcc</td><td>Variabile in cui viene scritto il valore di accelerazione necessario [unit/sec^2].</td></tr>
<tr><td>rJerk</td><td>Variabile in cui viene scritto il valore di jerk necessario [unit/sec^3].</td></tr>
<tr><td>maxAcc</td><td>Valore massimo che può assumere l'accelerazione. (opzionale, default 0.0)
Viene usato il suo valore assoluto.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td>3</td><td>tAcc e pJerk non rispettati a causa della limitazione di maxAcc,(in tal caso) il parametro pJerk viene assunto come se fosse 0.</td></tr>
<tr><td>2</td><td>tAcc non rispettato a causa della limitazione di maxAcc.</td></tr>
<tr><td>1</td><td>pJerk non rispettato a causa della limitazione di maxAcc.</td></tr>
<tr><td>0</td><td>Tutto OK.</td></tr>
<tr><td>-2</td><td>tAcc negativo o nullo (si assume 1).</td></tr>
<tr><td>-3</td><td>pJerk esterno all'intervallo 0-:-1 (si assume 0).</td></tr>
</table></td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td>Nel caso di accelerazione limitata da maxAcc, quando possibile, la funzione cerca di garantire comunque la durata del tratto di accelerazione tAcc a scapito del fattore di jerk pJerk.</td></tr>
</table>

© 2025 Robox SpA

Funzione log()

Funzione che restituisce il logaritmo naturale di *value*.

Sintassi	**real log (real value)**
value	Argomento del logaritmo.
Risultato	Rende il logaritmo naturale di *value*.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
reso = log(100)   ; reso = 4.605170185988092
reso = log(1000)  ; reso = 6.907755278982137
reso = log(1)     ; reso = 0
reso = log(0)     ; reso = -inf
```

Funzione log10()

Funzione che restituisce il logaritmo base 10 di *value*.

Sintassi	**real log10 (real value)**
value	Argomento del logaritmo.
Risultato	Rende il logaritmo base 10 di *value*.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
reso = log10(100)   ; reso = 2
reso = log10(1000)  ; reso = 3
reso = log10(1)     ; reso = 0
reso = log10(0)     ; reso = -inf
reso = log10(234)   ; reso = 2.369215857410143
```

© 2025 Robox SpA

Funzione loop_time()

Questa funzione viene utilizzata per misurare la durata di un loop software.

Sintassi	**real loop_time ()**
Risultato	Rende il tempo [ms] intercorso tra due esecuzioni successive della stessa funzione.
Validità	Rule, Task.
Note	La prima volta che viene eseguita, la funzione rende 0 e memorizza il tempo attuale. Nel caso si utilizzi loop_time() in una funzione R3 utente, si calcola il tempo che intercorre tra le chiamate della funzione.

Ogni funzione di loop_time() ha la propria memoria del tempo di esecuzione. Il calcolo della durata è la differenza tra il tempo attuale e la memoria relativa alla singola chiamata della funzione.

Esempi

Esempio di utilizzo:

```
__main_loop__
    time = loop_time() ; durata tra le varie chiamate del main
    ...
    ...
end_main_loop
```

Funzione master_filter()

Questa funzione viene utilizzata per filtrare la velocità di un master. A differenza della funzione filter_lp() essa recupera l'errore di posizione introdotto dal filtro.

Sintassi	**i32 master_filter (stru_masterF stru [, i32 nAx=0] [, real period=0])**
stru	Struttura di tipo stru_masterF contenente i parametri di input/output.
nAx	Numero asse del master se asse reale o 0 se asse virtuale. (opzionale, default 0)
period	Intervallo di tempo tra due esecuzioni consecutive di questa funzione. Se 0 si assume pari a SI. (opzionale, default 0).
Risultato	Rende: Valore: -5 [MV_ILLEGAL_ARGS] — Significato: Parametri illegali.

© 2025 Robox SpA

Sintassi	**i32 master_filter (stru_masterF stru [, i32 nAx=0] [, real period=0])**
	Valore / Significato 5 [MV_FOLLOWING] — Inseguimento in corso. 6 [MV_REACHED] — Missione OK.
Validità	Rule.
Note	

Esempi

Esempio di utilizzo:

```
stru_masterf 874 masterFiltered
; Filtriamo la velocità reale letta dall'encoder master 23
result = master_filter 389 (masterFiltered, 23, si)
; oppure
result = master_filter(masterFiltered, 23)
```

Funzione maximum()

Funzione che restituisce il maggiore fra i due valori forniti in ingresso.

Sintassi	**real maximum (real val1, real val2)**
val1	Primo valore da confrontare.
val2	Secondo valore da confrontare.
Risultato	Rende il maggiore fra *val1* e *val2*.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
velCalc = 60
velMin = 30


vel = maximum 390 (velCalc, velMin)
; vel = 60
```

© 2025 Robox SpA

Funzione mbc_connect()

Funzione utilizzata per connettere il client Modbus al server Modbus TCP specificato e configurare l'indice iniziale per i registri Modbus.

Sintassi	**i32 mbc_connect (i32 mbc, const char* ipAddr, i32 port [, i32 baseIdx=-1])**
mbc	Handle Modbus restituito dalla funzione precedente mbc_init 391 ().
ipAddr	Indirizzo IP del server a cui connettersi.
port	Porta TCP da utilizzare per la connessione. Solitamente 502.
baseIdx	Parametro per determinare come gestire l'indice iniziale delle entità. (opzionale, default -1)

Valore	Significato
< 0	Viene utilizzato sys_cfg.5 per determinare il modo di gestione dell'indice iniziale delle entità.
0	L'indice iniziale delle entità è 0 (non standard).
> 0	L'indice iniziale delle entità è 1 (standard).

Risultato — Rende:

Valore	Significato
> 0	Connessione con il server attivata.
-1	Argomenti illegali.
-2	Client già connesso a un server.

Validità	Task.
Requisiti di sistema	Da RTE 34.30.0.
Note	

Funzione mbc_init()

Funzione utilizzata per l'apertura di una comunicazione Modbus/TCP di tipo client.

La funzione crea un client Modbus che resta in vita finché il task corrente è in esecuzione. Se il task è riavviato, il client viene eliminato. Per questo motivo, per un corretto

© 2025 Robox SpA

funzionamento, si consiglia di eseguire la funzione ad ogni avvio del task (vedi variabile restarted).

Sintassi	**i32 mbc_init (string strIp, i32 nPort, real timeout, real expTime, i32 buffDim [, i32 initCode<0])**
strIp	Stringa con indirizzo IP target.
nPort	Numero di porta target.
timeout	Timeout in secondi su esecuzione comandi.
expTime	Tempo minimo garantito, in secondi, in cui la risposta rimane nel buffer prima di essere scartata per far posto a nuovi messaggi, nel caso in cui il buffer della gestione messaggi Modbus sia pieno.
buffDim	Dimensione del buffer della gestione messaggi Modbus. Rappresenta il numero massimo di comandi (funzioni lettura/scrittura Modbus) che si possono usare contemporaneamente.
initCode	Indica l'indice iniziale delle grandezze: • <0, viene usato il bit sys_cfg.5 per determinare l'indice iniziale delle grandezze • 0, l'indice iniziale delle grandezze è 0 (valore non standard) indipendentemente dallo stato di sys_cfg.5 • >0, l'indice iniziale delle grandezze è 1 (valore standard) indipendentemente dallo stato di sys_cfg.5 (opzionale, default <0)
Risultato	Rende l'handle di gestione delle comunicazioni che individua univocamente la comunicazione. Viene usato in tutte le altre funzioni di gestione Modbus.

Valore	Significato
> 0	Codice MBC di riferimento.
-1	Indirizzo IP illegale.
-2	Numero di porta illegale.
-3	Impossibile creare client.

Validità	Task.

© 2025 Robox SpA

Sintassi	**i32 mbc_init (string strIp, i32 nPort, real timeout, real expTime, i32 buffDim [, i32 initCode<0])**
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Da RTE 34.30.0 è stata introdotta una nuova gestione. Il client Modbus viene creato, ma non connesso, la gestione della connessione sarà poi gestibile tramite i comandi mbc_connect 391() e mbc_shutdown 410(). La riconnessione automatica al server non è disponibile per questi tipo di client.

Sintassi	**i32 mbc_init (real timeout, real expTime, i32 buffDim)**
timeout	Timeout in secondi su esecuzione comandi.
expTime	Tempo minimo garantito, in secondi, in cui la risposta rimane nel buffer prima di essere scartata per far posto a nuovi messaggi, nel caso in cui il buffer della gestione messaggi Modbus sia pieno.
buffDim	Dimensione del buffer della gestione messaggi Modbus. Rappresenta il numero massimo di comandi (funzioni lettura/scrittura Modbus) che si possono usare contemporaneamente.
Risultato	Rende l'handle di gestione delle comunicazioni che individua univocamente la comunicazione. Viene usato in tutte le altre funzioni di gestione Modbus.

Valore	Significato
> 0	Codice MBC di riferimento.
-1	Argomenti illegali.
-3	Impossibile creare client.

Validità	Task.
Requisiti di sistema	Da RTE 34.30.0.
Note	

Esempi

Esempio di utilizzo:

```
...
; per MODBUS client
lit MODBUS_IP_ADDR      "192.168.0.12"
```

© 2025 Robox SpA

```
lit MODBUS_PORT          502
....
....
int mbc
....
....
select(comPhase)
    case 1
        mbc = mbc_init(MODBUS_IP_ADDR, MODBUS_PORT, 2.0, 2.0, 2, 1)
        if(mbc <= 0)
            alarm_set(ALM_BAD_MBC_INIT, 0, mbc)
            comPhase = -1
        else
            comPhase = 2
        endif
        break
    case 2
        ....
```

Funzione mbc_mask_write_reg()

Questa funzione scrive i bit di un HOLDING REGISTER specificato su un server modbus.

Sintassi	**i32 mbc_mask_write_reg (i32 mbc, i32 offsHRStart, i32 endian, i32 bitMsk, i32 bitVal)**
mbc	Valore handle reso da mbc_init().
offsHRStart	Indice iniziale dell'HOLDING REGISTER.
endian	Tipo di endian da usare:

Valore	Significato
0 [EM_B_16]	Big Endian nella singola word. (default)
1 [EM_L_16]	Little Endian nella singola word.
2 [EM_B_32]	Big Endian in 32 bit.
3 [EM_L_32]	Little Endian in 32 bit.
4 [EM_MB_32]	Middle Big Endian in 32 bit.
5 [EM_ML_32]	Middle little Endian in 32 bit.
6 [EM_B_64]	Big Endian in 64 bit.
7 [EM_L_64]	Little Endian in 64 bit.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbc_mask_write_reg (i32 mbc, i32 offsHRStart, i32 endian, i32 bitMsk, i32 bitVal)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>8 [EM_MB_64]</td><td>Middle Big Endian in 64 bit.</td></tr>
<tr><td>9 [EM_ML_64]</td><td>Middle little Endian in 64 bit.</td></tr>
</table>
</td></tr>
<tr><td>bitMsk</td><td>Maschera dei bit che devono essere scritti.

NOTA: La maschera deve avere in 1 i bit interessati (R3-style, al contrario di quello che invece vuole il modbus).

<code>result = (result r_and r_not bitMsk) r_or (bitVal r_and bitMsk)</code></td></tr>
<tr><td>bitVal</td><td>Maschera contenente lo stato dei bit interessati.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
<tr><td>-9 [MBC_TIMEOUT]</td><td>Timeout comando bloccante.</td></tr>
<tr><td>-10 [MBC_EXC_CODE_1]</td><td>Risposta con eccezione 1 (function code non supportato dal server).</td></tr>
<tr><td>-11 [MBC_EXC_CODE_2]</td><td>Risposta con eccezione 2 (indirizzo non valido).</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 mbc_mask_write_reg (i32 mbc, i32 offsHRStart, i32 endian, i32 bitMsk, i32 bitVal)**
	Valore / Significato -12 [MBC_EXC_CODE_3] — Risposta con eccezione 3 (quantità o valore non validi). -13 [MBC_EXC_CODE_4] — Risposta con eccezione 4 (operazione fallita). -14 [MBC_WRONG_ANSWER] — Risposta non valida. -15 [MBC_INVAL_MBC] — Id *mbc* non valido.
Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione mbc_mask_write_reg_s()

Questa funzione scrive i bit di un HOLDING REGISTER specificato su un server modbus tramite una struttura di supporto in modalità passante.

Sintassi	**i32 mbc_mask_write_reg_s (i32 mbc, stru_modbus stru, i32 bitMsk, i32 bitVal)**
mbc	Valore handle reso da mbc_init 391 ().
stru	Struttura di tipo stru_modbus 876 contenente i parametri.
bitMsk	Maschera dei bit che devono essere scritti. **NOTA**: La maschera deve avere in 1 i bit interessati (R3-style, al contrario di quello che invece vuole il modbus). `result = (result r_and r_not bitMsk) r_or (bitVal r_and bitMsk)`
bitVal	Maschera contenente lo stato dei bit interessati.
Risultato	Rende: Valore / Significato 1 [MBC_WAIT] — Comando in attesa di risposta.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbc_mask_write_reg_s (i32 mbc, stru_modbus stru, i32 bitMsk, i32 bitVal)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
<tr><td>-9 [MBC_TIMEOUT]</td><td>Timeout comando bloccante.</td></tr>
<tr><td>-10 [MBC_EXC_CODE_1]</td><td>Risposta con eccezione 1 (function code non supportato dal server).</td></tr>
<tr><td>-11 [MBC_EXC_CODE_2]</td><td>Risposta con eccezione 2 (indirizzo non valido).</td></tr>
<tr><td>-12 [MBC_EXC_CODE_3]</td><td>Risposta con eccezione 3 (quantità o valore non validi).</td></tr>
<tr><td>-13 [MBC_EXC_CODE_4]</td><td>Risposta con eccezione 4 (operazione fallita).</td></tr>
<tr><td>-14 [MBC_WRONG_ANSWER]</td><td>Risposta non valida.</td></tr>
<tr><td>-15 [MBC_INVAL_MBC]</td><td>Id mbc non valido.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Task.</td></tr>
<tr><td>Note</td><td>La funzione è passante. Fino a che rende MBC_WAIT, va continuamente chiamata all'interno del task.</td></tr>
</table>

© 2025 Robox SpA

Funzione mbc_read_coils()

Funzione per la lettura di COILS su un server modbus.

Sintassi	**i32 mbc_read_coils (i32 mbc, i32 offsCoilsStart, i32 nCoil, dest)**
mbc	Valore handle reso da mbc_init 391 ().
offsCoilsStart	Indice iniziale del COIL.
nCoil	Numero di COILS da leggere a partire da *offsCoilsStart*. **NOTA**: Il massimo numero di COILS che possono essere letti è 2000.
dest	Area dove mettere i dati letti. Può essere i16/u16/i32 o array di i16/u16/i32.
Risultato	Rende:

Valore	Significato
0 [MBC_OK]	Comando eseguito.
-1 [MBC_INVAL]	Argomenti non validi.
-2 [MBC_BUSY]	Risorsa occupata.
-3 [MBC_PERM]	Permesso negato.
-4 [MBC_CONN]	Connessione non riuscita.
-5 [MBC_NOBUFS]	Buffer esauriti.
-6 [MBC_NOMEM]	Memoria insufficiente.
-7 [MBC_TX_ERR]	Errore di trasmissione.
-8 [MBC_RX_ERR]	Errore di ricezione.
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4

© 2025 Robox SpA

Sintassi	**i32 mbc_read_coils (i32 mbc, i32 offsCoilsStart, i32 nCoil, dest)**
	Valore / Significato
	(operazione fallita).
	-14 [MBC_WRONG_ANSWER] — Risposta non valida.
	-15 [MBC_INVAL_MBC] — Id *mbc* non valido.
Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione mbc_read_coils_s()

Funzione per la lettura di COILS su un server modbus tramite struttura di supporto in modalità passante.

Sintassi	**i32 mbc_read_coils_s (i32 mbc, stru_modbus stru, dest)**
mbc	Valore handle reso da mbc_init 391 ().
stru	Struttura di tipo stru_modbus 876 contenente i parametri.
dest	Area dove mettere i dati letti. Può essere i16/u16/i32 o array di i16/u16/i32.
Risultato	Rende:

Valore	Significato
1 [MBC_WAIT]	Comando in attesa di risposta.
0 [MBC_OK]	Comando eseguito.
-1 [MBC_INVAL]	Argomenti non validi.
-2 [MBC_BUSY]	Risorsa occupata.
-3 [MBC_PERM]	Permesso negato.
-4 [MBC_CONN]	Connessione non riuscita.
-5 [MBC_NOBUFS]	Buffer esauriti.
-6 [MBC_NOMEM]	Memoria insufficiente.

© 2025 Robox SpA

Sintassi	**i32 mbc_read_coils_s (i32 mbc, stru_modbus stru, dest)**

Valore	Significato
-7 [MBC_TX_ERR]	Errore di trasmissione.
-8 [MBC_RX_ERR]	Errore di ricezione.
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4 (operazione fallita).
-14 [MBC_WRONG_ANSWER]	Risposta non valida.
-15 [MBC_INVAL_MBC]	Id *mbc* non valido.

Validità	Task.
Note	La funzione è passante. Fino a che rende MBC_WAIT, va continuamente chiamata all'interno del task.

Funzione mbc_read_discr_inp()

Funzione per la lettura di DISCRETE INPUTS su server modbus.

Sintassi	**i32 mbc_read_discr_inp (i32 mbc, i32 offsDIStart, i32 nDI, dest)**
mbc	Valore handle reso da mbc_init 391 ().
offsDIStart	Indice iniziale del DISCRETE INPUT.
nDI	Numero di DISCRETE INPUTS da leggere a partire da offsDIStart.
dest	Area dove mettere i dati letti. Può essere i16/u16/i32 o array di i16/u16/i32.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbc_read_discr_inp (i32 mbc, i32 offsDIStart, i32 nDI, dest)</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
<tr><td>-9 [MBC_TIMEOUT]</td><td>Timeout comando bloccante.</td></tr>
<tr><td>-10 [MBC_EXC_CODE_1]</td><td>Risposta con eccezione 1 (function code non supportato dal server).</td></tr>
<tr><td>-11 [MBC_EXC_CODE_2]</td><td>Risposta con eccezione 2 (indirizzo non valido).</td></tr>
<tr><td>-12 [MBC_EXC_CODE_3]</td><td>Risposta con eccezione 3 (quantità o valore non validi).</td></tr>
<tr><td>-13 [MBC_EXC_CODE_4]</td><td>Risposta con eccezione 4 (operazione fallita).</td></tr>
<tr><td>-14 [MBC_WRONG_ANSWER]</td><td>Risposta non valida.</td></tr>
<tr><td>-15 [MBC_INVAL_MBC]</td><td>Id mbc non valido.</td></tr>
</table></td></tr>
<tr><td>Validità</td><td>Task.</td></tr>
<tr><td>Note</td><td>La funzione è bloccante. Esce solo quando è stata completata l'operazione.</td></tr>
</table>

© 2025 Robox SpA

Funzione mbc_read_discr_inp_s()

Funzione per la lettura di DISCRETE INPUTS su server modbus specificati tramite una struttura di supporto in modalità passante.

Sintassi	**i32 mbc_read_discr_inp_s (i32 mbc, stru_modbus stru, dest)**
mbc	Valore handle reso da mbc_init 391 ().
stru	Struttura tipo stru_modbus 876 contenente i parametri.
dest	Area dove mettere i dati letti. Può essere i16/u16/i32 o array di i16/u16/i32.
Risultato	Rende:

Valore	Significato
1 [MBC_WAIT]	Comando in attesa di risposta.
0 [MBC_OK]	Comando eseguito.
-1 [MBC_INVAL]	Argomenti non validi.
-2 [MBC_BUSY]	Risorsa occupata.
-3 [MBC_PERM]	Permesso negato.
-4 [MBC_CONN]	Connessione non riuscita.
-5 [MBC_NOBUFS]	Buffer esauriti.
-6 [MBC_NOMEM]	Memoria insufficiente.
-7 [MBC_TX_ERR]	Errore di trasmissione.
-8 [MBC_RX_ERR]	Errore di ricezione.
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4 (operazione fallita).

© 2025 Robox SpA

Sintassi	**i32 mbc_read_discr_inp_s (i32 mbc, stru_modbus stru, dest)**
	Valore / Significato
	-14 [MBC_WRONG_ANSWER] — Risposta non valida.
	-15 [MBC_INVAL_MBC] — Id *mbc* non valido.
Validità	Task.
Note	La funzione è passante. Fino a che rende MBC_WAIT, va continuamente chiamata all'interno del task.

Funzione mbc_read_hold_reg()

Funzione per la lettura di HOLDING REGISTER specificati su server modbus.

Sintassi	**i32 mbc_read_hold_reg (i32 mbc, i32 offsHRStart, i32 nHR, i32 endian, dest)**
mbc	Valore handle reso da mbc_init391().
offsHRStart	Indice iniziale dell'HOLDING REGISTER.
nHR	Numero di HOLDING REGISTER da leggere. **NOTA**: Il massimo numero di HOLDING REGISTER che possono essere letti è 125.
endian	Tipo di endian da usare:

Valore	Significato
0 [EM_B_16]	Big Endian nella singola word. (default)
1 [EM_L_16]	Little Endian nella singola word.
2 [EM_B_32]	Big Endian in 32 bit.
3 [EM_L_32]	Little Endian in 32 bit.
4 [EM_MB_32]	Middle Big Endian in 32 bit.
5 [EM_ML_32]	Middle little Endian in 32 bit.
6 [EM_B_64]	Big Endian in 64 bit.
7 [EM_L_64]	Little Endian in 64 bit.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbc_read_hold_reg (i32 mbc, i32 offsHRStart, i32 nHR, i32 endian, dest)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>8 [EM_MB_64]</td><td>Middle Big Endian in 64 bit.</td></tr>
<tr><td>9 [EM_ML_64]</td><td>Middle little Endian in 64 bit.</td></tr>
</table>
</td></tr>
<tr><td>dest</td><td>Area dove mettere i dati letti.
Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
<tr><td>-9 [MBC_TIMEOUT]</td><td>Timeout comando bloccante.</td></tr>
<tr><td>-10 [MBC_EXC_CODE_1]</td><td>Risposta con eccezione 1 (function code non supportato dal server).</td></tr>
<tr><td>-11 [MBC_EXC_CODE_2]</td><td>Risposta con eccezione 2 (indirizzo non valido).</td></tr>
<tr><td>-12 [MBC_EXC_CODE_3]</td><td>Risposta con eccezione 3 (quantità o valore non validi).</td></tr>
<tr><td>-13 [MBC_EXC_CODE_4]</td><td>Risposta con eccezione 4 (operazione fallita).</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 mbc_read_hold_reg (i32 mbc, i32 offsHRStart, i32 nHR, i32 endian, dest)**
	Valore / Significato
	-14 [MBC_WRONG_ANSWER] — Risposta non valida.
	-15 [MBC_INVAL_MBC] — Id *mbc* non valido.
Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione mbc_read_hold_reg_s()

Funzione per la lettura di HOLDING REGISTER specificati su server modbus tramite una struttura di supporto in modalità passante.

Sintassi	**i32 mbc_read_hold_reg_s (i32 mbc, stru_modbus stru, dest)**
mbc	Valore handle reso da mbc_init 391 ().
stru	Struttura di tipo stru_modbus 876 contenente i parametri.
dest	Area dove mettere i dati letti. Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.
Risultato	Rende:

Valore	Significato
1 [MBC_WAIT]	Comando in attesa di risposta.
0 [MBC_OK]	Comando eseguito.
-1 [MBC_INVAL]	Argomenti non validi.
-2 [MBC_BUSY]	Risorsa occupata.
-3 [MBC_PERM]	Permesso negato.
-4 [MBC_CONN]	Connessione non riuscita.
-5 [MBC_NOBUFS]	Buffer esauriti.
-6 [MBC_NOMEM]	Memoria insufficiente.
-7 [MBC_TX_ERR]	Errore di trasmissione.

© 2025 Robox SpA

Sintassi	**i32 mbc_read_hold_reg_s (i32 mbc, stru_modbus stru, dest)**

Valore	Significato
-8 [MBC_RX_ERR]	Errore di ricezione.
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4 (operazione fallita).
-14 [MBC_WRONG_ANSWER]	Risposta non valida.
-15 [MBC_INVAL_MBC]	Id *mbc* non valido.

Validità	Task.
Note	La funzione è passante. Fino a che rende MBC_WAIT, va continuamente chiamata all'interno del task.

Funzione mbc_read_inp_reg()

Funzione per la lettura di INPUT REGISTER specificati su server modbus.

Sintassi	**i32 mbc_read_inp_reg (i32 mbc, i32 offsIRStart, i32 nIR, i32 endian, dest)**
mbc	Valore handle reso da mbc_init 391 ().
offsIRStart	Indice iniziale dell'INPUT REGISTER.
nIR	Numero di INPUT REGISTER da leggere.
endian	Tipo di endian da usare:

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbc_read_inp_reg (i32 mbc, i32 offsIRStart, i32 nIR, i32 endian, dest)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [EM_B_16]</td><td>Big Endian nella singola word. (default)</td></tr>
<tr><td>1 [EM_L_16]</td><td>Little Endian nella singola word.</td></tr>
<tr><td>2 [EM_B_32]</td><td>Big Endian in 32 bit.</td></tr>
<tr><td>3 [EM_L_32]</td><td>Little Endian in 32 bit.</td></tr>
<tr><td>4 [EM_MB_32]</td><td>Middle Big Endian in 32 bit.</td></tr>
<tr><td>5 [EM_ML_32]</td><td>Middle little Endian in 32 bit.</td></tr>
<tr><td>6 [EM_B_64]</td><td>Big Endian in 64 bit.</td></tr>
<tr><td>7 [EM_L_64]</td><td>Little Endian in 64 bit.</td></tr>
<tr><td>8 [EM_MB_64]</td><td>Middle Big Endian in 64 bit.</td></tr>
<tr><td>9 [EM_ML_64]</td><td>Middle little Endian in 64 bit.</td></tr>
</table>
</td></tr>
<tr><td>dest</td><td>Area dove mettere i dati letti.
Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 mbc_read_inp_reg (i32 mbc, i32 offsIRStart, i32 nIR, i32 endian, dest)**

Valore	Significato
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4 (operazione fallita).
-14 [MBC_WRONG_ANSWER]	Risposta non valida.
-15 [MBC_INVAL_MBC]	Id *mbc* non valido.

Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione mbc_read_inp_reg_s()

Funzione per la lettura di INPUT REGISTER specificati su server modbus tramite una struttura di supporto in modalità passante.

Sintassi	**i32 mbc_read_inp_reg_s (i32 mbc, stru_modbus stru, dest)**
mbc	Valore handle reso da mbc_init 391 ().
stru	Struttura di tipo stru_modbus 876 contenente i parametri.
dest	Area dove mettere i dati letti. Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.
Risultato	Rende:

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbc_read_inp_reg_s (i32 mbc, stru_modbus stru, dest)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>1 [MBC_WAIT]</td><td>Comando in attesa di risposta.</td></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
<tr><td>-9 [MBC_TIMEOUT]</td><td>Timeout comando bloccante.</td></tr>
<tr><td>-10 [MBC_EXC_CODE_1]</td><td>Risposta con eccezione 1 (function code non supportato dal server).</td></tr>
<tr><td>-11 [MBC_EXC_CODE_2]</td><td>Risposta con eccezione 2 (indirizzo non valido).</td></tr>
<tr><td>-12 [MBC_EXC_CODE_3]</td><td>Risposta con eccezione 3 (quantità o valore non validi).</td></tr>
<tr><td>-13 [MBC_EXC_CODE_4]</td><td>Risposta con eccezione 4 (operazione fallita).</td></tr>
<tr><td>-14 [MBC_WRONG_ANSWER]</td><td>Risposta non valida.</td></tr>
<tr><td>-15 [MBC_INVAL_MBC]</td><td>Id mbc non valido.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Task.</td></tr>
<tr><td>Note</td><td>La funzione è passante. Fino a che rende MBC_WAIT, va continuamente chiamata all'interno del task.</td></tr>
</table>

© 2025 Robox SpA

Funzione mbc_shutdown()

Funzione utilizzata per chiudere la connessione con il server Modbus TCP specificato.

Sintassi	**i32 mbc_shutdown (i32 mbc)**
mbc	Handle Modbus restituito dalla funzione precedente mbc_init391().
Risultato	Rende: Valore: >= 0 — Significato: ERRNO se operazione di disconnessione OK o uno dei codici MBC_const68 di riferimento. Valore: -1 — Significato: Argomenti illegali. Valore: -4 — Significato: Client già disconnesso dal server (MBC_CONN).
Validità	Task.
Requisiti di sistema	Da RTE 34.30.0.
Note	

Funzione mbc_write_coils()

Questa funzione scrive i COILS specificati su un server modbus.

Sintassi	**i32 mbc_write_coils (i32 mbc, i32 offsCoilStart, i32 nCoil, source)**
mbc	Valore handle reso da mbc_init391().
offsCoilStart	Indice iniziale del COIL.
nCoil	Numero di COILS da scrivere. **NOTA**: Il massimo numero di COILS che possono essere scritti è 1968.
source	Area dove prendere i dati da scrivere. Può essere i16/u16/i32 o array di i16/u16/i32.
Risultato	Rende: Valore: 0 [MBC_OK] — Significato: Comando eseguito. Valore: -1 [MBC_INVAL] — Significato: Argomenti non validi. Valore: -2 [MBC_BUSY] — Significato: Risorsa occupata.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbc_write_coils (i32 mbc, i32 offsCoilStart, i32 nCoil, source)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
<tr><td>-9 [MBC_TIMEOUT]</td><td>Timeout comando bloccante.</td></tr>
<tr><td>-10 [MBC_EXC_CODE_1]</td><td>Risposta con eccezione 1 (function code non supportato dal server).</td></tr>
<tr><td>-11 [MBC_EXC_CODE_2]</td><td>Risposta con eccezione 2 (indirizzo non valido).</td></tr>
<tr><td>-12 [MBC_EXC_CODE_3]</td><td>Risposta con eccezione 3 (quantità o valore non validi).</td></tr>
<tr><td>-13 [MBC_EXC_CODE_4]</td><td>Risposta con eccezione 4 (operazione fallita).</td></tr>
<tr><td>-14 [MBC_WRONG_ANSWER]</td><td>Risposta non valida.</td></tr>
<tr><td>-15 [MBC_INVAL_MBC]</td><td>Id mbc non valido.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Task.</td></tr>
<tr><td>Note</td><td>La funzione è bloccante. Esce solo quando è stata completata l'operazione.</td></tr>
</table>

© 2025 Robox SpA

Funzione mbc_write_coils_s()

Questa funzione scrive i COILS specificati su un server modbus tramite una struttura di supporto in modalità passante.

Sintassi	**i32 mbc_write_coils_s (i32 mbc, stru_modbus stru, source)**
mbc	Valore handle reso da mbc_init 391 ().
stru	Struttura di tipo stru_modbus 876 contenente i parametri.
source	Area dove prendere i dati da scrivere. Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.
Risultato	Rende:

Valore	Significato
1 [MBC_WAIT]	Comando in attesa di risposta.
0 [MBC_OK]	Comando eseguito.
-1 [MBC_INVAL]	Argomenti non validi.
-2 [MBC_BUSY]	Risorsa occupata.
-3 [MBC_PERM]	Permesso negato.
-4 [MBC_CONN]	Connessione non riuscita.
-5 [MBC_NOBUFS]	Buffer esauriti.
-6 [MBC_NOMEM]	Memoria insufficiente.
-7 [MBC_TX_ERR]	Errore di trasmissione.
-8 [MBC_RX_ERR]	Errore di ricezione.
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4

© 2025 Robox SpA

Sintassi	**i32 mbc_write_coils_s (i32 mbc, stru_modbus stru, source)**
	Valore / Significato
	(operazione fallita).
	-14 [MBC_WRONG_ANSWER] — Risposta non valida.
	-15 [MBC_INVAL_MBC] — Id *mbc* non valido.
Validità	Task.
Note	La funzione è passante. Fino a che rende MBC_WAIT, va continuamente chiamata all'interno del task.

Funzione mbc_write_multiple_coils()

Questa funzione scrive i COILS specificati usando sempre il codice funzione Modbus delle scritture multiple corrispettive FC 15, diversamente da mbc_write_coils 410() che può utilizzare anche FC 05 (se COIL singolo).

Sintassi	**i32 mbc_write_multiple_coils (i32 mbc, i32 offsCoilStart, i32 nCoil, source)**
mbc	Valore handle reso da mbc_init 391().
offsCoilStart	Indice iniziale del COIL.
nCoil	Numero di COILS da scrivere. **NOTA**: Il massimo numero di COILS che possono essere scritti è 1968.
source	Area dove prendere i dati da scrivere. Può essere i16/u16/i32 o array di i16/u16/i32.
Risultato	Rende:
	Valore / Significato
	0 [MBC_OK] — Comando eseguito.
	-1 [MBC_INVAL] — Argomenti non validi.
	-2 [MBC_BUSY] — Risorsa occupata.
	-3 [MBC_PERM] — Permesso negato.
	-4 [MBC_CONN] — Connessione non riuscita.
	-5 [MBC_NOBUFS] — Buffer esauriti.

© 2025 Robox SpA

Sintassi	**i32 mbc_write_multiple_coils (i32 mbc, i32 offsCoilStart, i32 nCoil, source)**

Valore	Significato
-6 [MBC_NOMEM]	Memoria insufficiente.
-7 [MBC_TX_ERR]	Errore di trasmissione.
-8 [MBC_RX_ERR]	Errore di ricezione.
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4 (operazione fallita).
-14 [MBC_WRONG_ANSWER]	Risposta non valida.
-15 [MBC_INVAL_MBC]	Id *mbc* non valido.

Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione mbc_write_multiple_coils_s()

Questa funzione scrive i COILS specificati su un server Modbus tramite una struttura di supporto in modalità passante e usando sempre il codice funzione Modbus delle scritture multiple corrispettive FC 15, diversamente da mbc_write_coils_s 412() che può utilizzare anche FC 05.

Sintassi	**i32 mbc_write_multiple_coils_s (i32 mbc, stru_modbus stru, source)**
mbc	Valore handle reso da mbc_init 391().
stru	Struttura di tipo stru_modbus 876 contenente i parametri.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbc_write_multiple_coils_s (i32 mbc, stru_modbus stru, source)</td></tr>
<tr><td>source</td><td>Area dove prendere i dati da scrivere.
Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>1 [MBC_WAIT]</td><td>Comando in attesa di risposta.</td></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
<tr><td>-9 [MBC_TIMEOUT]</td><td>Timeout comando bloccante.</td></tr>
<tr><td>-10 [MBC_EXC_CODE_1]</td><td>Risposta con eccezione 1 (function code non supportato dal server).</td></tr>
<tr><td>-11 [MBC_EXC_CODE_2]</td><td>Risposta con eccezione 2 (indirizzo non valido).</td></tr>
<tr><td>-12 [MBC_EXC_CODE_3]</td><td>Risposta con eccezione 3 (quantità o valore non validi).</td></tr>
<tr><td>-13 [MBC_EXC_CODE_4]</td><td>Risposta con eccezione 4 (operazione fallita).</td></tr>
<tr><td>-14 [MBC_WRONG_ANSWER]</td><td>Risposta non valida.</td></tr>
<tr><td>-15 [MBC_INVAL_MBC]</td><td>Id mbc non valido.</td></tr>
</table></td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 mbc_write_multiple_coils_s (i32 mbc, stru_modbus stru, source)**
Validità	Task.
Note	La funzione è passante. Fino a che rende MBC_WAIT, va continuamente chiamata all'interno del task.

Funzione mbc_write_multiple_regs()

Questa funzione scrive gli HOLDING REGISTER specificati su un server Modbus usando il codice funzione Modbus delle scritture multiple corrispettive FC 16, diversamente da mbc_write_reg 419() che può utilizzare anche FC 06.

Sintassi	**i32 mbc_write_multiple_regs (i32 mbc, i32 offsHRStart, i32 nHR, i32 endian, source)**
mbc	Valore handle reso da mbc_init 391().
offsHRStart	Indice iniziale dell'HOLDING REGISTER.
nHR	Numero di HOLDING REGISTER da scrivere. **NOTA**: il massimo numero di HOLDING REGISTER che possono essere scritti è 123.
endian	Tipo di endian da usare:

Valore	Significato
0 [EM_B_16]	Big Endian nella singola word. (default)
1 [EM_L_16]	Little Endian nella singola word.
2 [EM_B_32]	Big Endian in 32 bit.
3 [EM_L_32]	Little Endian in 32 bit.
4 [EM_MB_32]	Middle Big Endian in 32 bit.
5 [EM_ML_32]	Middle little Endian in 32 bit.
6 [EM_B_64]	Big Endian in 64 bit.
7 [EM_L_64]	Little Endian in 64 bit.
8 [EM_MB_64]	Middle Big Endian in 64 bit.
9 [EM_ML_64]	Middle little Endian in 64 bit.

source	Area dove prendere i dati da scrivere.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbc_write_multiple_regs (i32 mbc, i32 offsHRStart, i32 nHR, i32 endian, source)</td></tr>
<tr><td></td><td>Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
<tr><td>-9 [MBC_TIMEOUT]</td><td>Timeout comando bloccante.</td></tr>
<tr><td>-10 [MBC_EXC_CODE_1]</td><td>Risposta con eccezione 1 (function code non supportato dal server).</td></tr>
<tr><td>-11 [MBC_EXC_CODE_2]</td><td>Risposta con eccezione 2 (indirizzo non valido).</td></tr>
<tr><td>-12 [MBC_EXC_CODE_3]</td><td>Risposta con eccezione 3 (quantità o valore non validi).</td></tr>
<tr><td>-13 [MBC_EXC_CODE_4]</td><td>Risposta con eccezione 4 (operazione fallita).</td></tr>
<tr><td>-14 [MBC_WRONG_ANSWER]</td><td>Risposta non valida.</td></tr>
<tr><td>-15 [MBC_INVAL_MBC]</td><td>Id mbc non valido.</td></tr>
</table></td></tr>
<tr><td>Validità</td><td>Task.</td></tr>
<tr><td>Note</td><td>La funzione è bloccante. Esce solo quando è stata completata l'operazione.</td></tr>
</table>

© 2025 Robox SpA

Funzione mbc_write_multiple_regs_s()

Questa funzione scrive gli HOLDING REGISTER specificati su un server Modbus tramite una struttura di supporto in modalità passante e usando il codice funzione Modbus delle scritture multiple corrispettive FC 16, diversamente da mbc_write_reg_s() che può utilizzare anche FC 06.

Sintassi	**i32 mbc_write_multiple_regs_s (i32 mbc, stru_modbus stru, source)**
mbc	Valore handle reso da mbc_init().
stru	Struttura di tipo stru_modbus contenente i parametri.
source	Area dove prendere i dati da scrivere. Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.
Risultato	Rende:

Valore	Significato
1 [MBC_WAIT]	Comando in attesa di risposta.
0 [MBC_OK]	Comando eseguito.
-1 [MBC_INVAL]	Argomenti non validi.
-2 [MBC_BUSY]	Risorsa occupata.
-3 [MBC_PERM]	Permesso negato.
-4 [MBC_CONN]	Connessione non riuscita.
-5 [MBC_NOBUFS]	Buffer esauriti.
-6 [MBC_NOMEM]	Memoria insufficiente.
-7 [MBC_TX_ERR]	Errore di trasmissione.
-8 [MBC_RX_ERR]	Errore di ricezione.
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità

© 2025 Robox SpA

Sintassi	**i32 mbc_write_multiple_regs_s (i32 mbc, stru_modbus stru, source)**

Valore	Significato
	o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4 (operazione fallita).
-14 [MBC_WRONG_ANSWER]	Risposta non valida.
-15 [MBC_INVAL_MBC]	Id *mbc* non valido.

Validità	Task.
Note	La funzione è passante. Fino a che rende MBC_WAIT, va continuamente chiamata all'interno del task.

Funzione mbc_write_reg()

Questa funzione scrive gli HOLDING REGISTER specificati su un server modbus.

Sintassi	**i32 mbc_write_reg (i32 mbc, i32 offsHRStart, i32 nHR, i32 endian, source)**
mbc	Valore handle reso da mbc_init 391 ().
offsHRStart	Indice iniziale dell'HOLDING REGISTER.
nHR	Numero di HOLDING REGISTER da scrivere. **NOTA**: il massimo numero di HOLDING REGISTER che possono essere scritti è 123.
endian	Tipo di endian da usare:

Valore	Significato
0 [EM_B_16]	Big Endian nella singola word. (default)
1 [EM_L_16]	Little Endian nella singola word.
2 [EM_B_32]	Big Endian in 32 bit.
3 [EM_L_32]	Little Endian in 32 bit.
4 [EM_MB_32]	Middle Big Endian in 32 bit.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbc_write_reg (i32 mbc, i32 offsHRStart, i32 nHR, i32 endian, source)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>5 [EM_ML_32]</td><td>Middle little Endian in 32 bit.</td></tr>
<tr><td>6 [EM_B_64]</td><td>Big Endian in 64 bit.</td></tr>
<tr><td>7 [EM_L_64]</td><td>Little Endian in 64 bit.</td></tr>
<tr><td>8 [EM_MB_64]</td><td>Middle Big Endian in 64 bit.</td></tr>
<tr><td>9 [EM_ML_64]</td><td>Middle little Endian in 64 bit.</td></tr>
</table>
</td></tr>
<tr><td>source</td><td>Area dove prendere i dati da scrivere.
Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
<tr><td>-9 [MBC_TIMEOUT]</td><td>Timeout comando bloccante.</td></tr>
<tr><td>-10 [MBC_EXC_CODE_1]</td><td>Risposta con eccezione 1 (function code non supportato dal server).</td></tr>
<tr><td>-11 [MBC_EXC_CODE_2]</td><td>Risposta con eccezione 2 (indirizzo non valido).</td></tr>
<tr><td>-12 [MBC_EXC_CODE_3]</td><td>Risposta con eccezione 3 (quantità</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 mbc_write_reg (i32 mbc, i32 offsHRStart, i32 nHR, i32 endian, source)**
	Valore / Significato
	o valore non validi).
	-13 [MBC_EXC_CODE_4] — Risposta con eccezione 4 (operazione fallita).
	-14 [MBC_WRONG_ANSWER] — Risposta non valida.
	-15 [MBC_INVAL_MBC] — Id *mbc* non valido.
Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione mbc_write_reg_s()

Questa funzione scrive gli HOLDING REGISTER specificati su un server modbus tramite una struttura di supporto in modalità passante.

Sintassi	**i32 mbc_write_reg_s (i32 mbc, stru_modbus stru, source)**
mbc	Valore handle reso da mbc_init 391 ().
stru	Struttura di tipo stru_modbus 876 contenente i parametri.
source	Area dove prendere i dati da scrivere. Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.
Risultato	Rende:
	Valore / Significato
	1 [MBC_WAIT] — Comando in attesa di risposta.
	0 [MBC_OK] — Comando eseguito.
	-1 [MBC_INVAL] — Argomenti non validi.
	-2 [MBC_BUSY] — Risorsa occupata.
	-3 [MBC_PERM] — Permesso negato.
	-4 [MBC_CONN] — Connessione non riuscita.

© 2025 Robox SpA

Sintassi	**i32 mbc_write_reg_s (i32 mbc, stru_modbus stru, source)**

Valore	Significato
-5 [MBC_NOBUFS]	Buffer esauriti.
-6 [MBC_NOMEM]	Memoria insufficiente.
-7 [MBC_TX_ERR]	Errore di trasmissione.
-8 [MBC_RX_ERR]	Errore di ricezione.
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4 (operazione fallita).
-14 [MBC_WRONG_ANSWER]	Risposta non valida.
-15 [MBC_INVAL_MBC]	Id *mbc* non valido.

Validità	Task.
Note	La funzione è passante. Fino a che rende MBC_WAIT, va continuamente chiamata all'interno del task.

Funzione mbcu_mask_write_reg()

Questa funzione scrive i bit di un HOLDING REGISTER su un server modbus specificando lo unit ID.

Sintassi	**i32 mbcu_mask_write_reg (i32 mbc, i32 unitID, i32 offsHRStart, i32 endian, i32 bitMsk, i32 bitVal)**
mbc	Valore handle reso da mbc_init 391 ().
unitID	Valore dello unit ID da usarsi.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbcu_mask_write_reg (i32 mbc, i32 unitID, i32 offsHRStart, i32 endian, i32 bitMsk, i32 bitVal)</td></tr>
<tr><td>offsHRStart</td><td>Indice iniziale dell'HOLDING REGISTER.</td></tr>
<tr><td>endian</td><td>Tipo di endian da usare:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [EM_B_16]</td><td>Big Endian nella singola word. (default)</td></tr>
<tr><td>1 [EM_L_16]</td><td>Little Endian nella singola word.</td></tr>
<tr><td>2 [EM_B_32]</td><td>Big Endian in 32 bit.</td></tr>
<tr><td>3 [EM_L_32]</td><td>Little Endian in 32 bit.</td></tr>
<tr><td>4 [EM_MB_32]</td><td>Middle Big Endian in 32 bit.</td></tr>
<tr><td>5 [EM_ML_32]</td><td>Middle little Endian in 32 bit.</td></tr>
<tr><td>6 [EM_B_64]</td><td>Big Endian in 64 bit.</td></tr>
<tr><td>7 [EM_L_64]</td><td>Little Endian in 64 bit.</td></tr>
<tr><td>8 [EM_MB_64]</td><td>Middle Big Endian in 64 bit.</td></tr>
<tr><td>9 [EM_ML_64]</td><td>Middle little Endian in 64 bit.</td></tr>
</table></td></tr>
<tr><td>bitMsk</td><td>Maschera dei bit che devono essere scritti.

NOTA: La maschera deve avere in 1 i bit interessati (R3-style, al contrario di quello che invece vuole il modbus).

<code>result = (result r_and r_not bitMsk) r_or (bitVal r_and bitMsk)</code></td></tr>
<tr><td>bitVal</td><td>Maschera contenente lo stato dei bit interessati.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
</table></td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 mbcu_mask_write_reg (i32 mbc, i32 unitID, i32 offsHRStart, i32 endian, i32 bitMsk, i32 bitVal)**

Valore	Significato
-4 [MBC_CONN]	Connessione non riuscita.
-5 [MBC_NOBUFS]	Buffer esauriti.
-6 [MBC_NOMEM]	Memoria insufficiente.
-7 [MBC_TX_ERR]	Errore di trasmissione.
-8 [MBC_RX_ERR]	Errore di ricezione.
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4 (operazione fallita).
-14 [MBC_WRONG_ANSWER]	Risposta non valida.
-15 [MBC_INVAL_MBC]	Id *mbc* non valido.

Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione mbcu_read_coils()

Funzione per la lettura di COILS su un server modbus specificando lo unit ID.

Sintassi	**i32 mbcu_read_coils (i32 mbc, i32 unitID, i32 offsCoilsStart, i32 nCoil, dest)**
mbc	Valore handle reso da mbc_init 391 ().

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbcu_read_coils (i32 mbc, i32 unitID, i32 offsCoilsStart, i32 nCoil, dest)</td></tr>
<tr><td>unitID</td><td>Valore dello unit ID da usarsi.</td></tr>
<tr><td>offsCoilsStart</td><td>Indice iniziale del COIL.</td></tr>
<tr><td>nCoil</td><td>Numero di COILS da leggere a partire da offsCoilsStart.

NOTA: Il massimo numero di COILS che possono essere letti è 2000.</td></tr>
<tr><td>dest</td><td>Area dove mettere i dati letti.

Può essere i16/u16/i32 o array di i16/u16/i32.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
<tr><td>-9 [MBC_TIMEOUT]</td><td>Timeout comando bloccante.</td></tr>
<tr><td>-10 [MBC_EXC_CODE_1]</td><td>Risposta con eccezione 1 (function code non supportato dal server).</td></tr>
<tr><td>-11 [MBC_EXC_CODE_2]</td><td>Risposta con eccezione 2 (indirizzo non valido).</td></tr>
<tr><td>-12 [MBC_EXC_CODE_3]</td><td>Risposta con eccezione 3 (quantità o valore non validi).</td></tr>
<tr><td>-13 [MBC_EXC_CODE_4]</td><td>Risposta con eccezione 4 (operazione fallita).</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 mbcu_read_coils (i32 mbc, i32 unitID, i32 offsCoilsStart, i32 nCoil, dest)**
	Valore / Significato
	-14 [MBC_WRONG_ANSWER] — Risposta non valida.
	-15 [MBC_INVAL_MBC] — Id *mbc* non valido.
Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione mbcu_read_discr_inp()

Funzione per la lettura di DISCRETE INPUTS su server modbus specificando lo unit ID.

Sintassi	**i32 mbcu_read_discr_inp (i32 mbc, i32 unitID, i32 offsDIStart, i32 nDI, dest)**
mbc	Valore handle reso da mbc_init 391().
unitID	Valore dello unit ID da usarsi.
offsDIStart	Indice iniziale del DISCRETE INPUT.
nDI	Numero di DISCRETE INPUTS da leggere a partire da offsDIStart.
dest	Area dove mettere i dati letti. Può essere i16/u16/i32 o array di i16/u16/i32.
Risultato	Rende:

Valore	Significato
0 [MBC_OK]	Comando eseguito.
-1 [MBC_INVAL]	Argomenti non validi.
-2 [MBC_BUSY]	Risorsa occupata.
-3 [MBC_PERM]	Permesso negato.
-4 [MBC_CONN]	Connessione non riuscita.
-5 [MBC_NOBUFS]	Buffer esauriti.
-6 [MBC_NOMEM]	Memoria insufficiente.
-7 [MBC_TX_ERR]	Errore di trasmissione.

© 2025 Robox SpA

Sintassi	**i32 mbcu_read_discr_inp (i32 mbc, i32 unitID, i32 offsDIStart, i32 nDI, dest)**

Valore	Significato
-8 [MBC_RX_ERR]	Errore di ricezione.
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4 (operazione fallita).
-14 [MBC_WRONG_ANSWER]	Risposta non valida.
-15 [MBC_INVAL_MBC]	Id *mbc* non valido.

Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione mbcu_read_hold_reg()

Funzione per la lettura di HOLDING REGISTER su server modbus specificando lo unit ID.

Sintassi	**i32 mbcu_read_hold_reg (i32 mbc, i32 unitID, i32 offsHRStart, i32 nHR, i32 endian, dest)**
mbc	Valore handle reso da mbc_init 391 ().
unitID	Valore dello unit ID da usarsi.
offsHRStart	Indice iniziale dell'HOLDING REGISTER.
nHR	Numero di HOLDING REGISTER da leggere. **NOTA**: Il massimo numero di HOLDING REGISTER che possono essere letti è 125.

© 2025 Robox SpA

Sintassi	**i32 mbcu_read_hold_reg (i32 mbc, i32 unitID, i32 offsHRStart, i32 nHR, i32 endian, dest)**
endian	Tipo di endian da usare: Valore / Significato 0 [EM_B_16] — Big Endian nella singola word. (default) 1 [EM_L_16] — Little Endian nella singola word. 2 [EM_B_32] — Big Endian in 32 bit. 3 [EM_L_32] — Little Endian in 32 bit. 4 [EM_MB_32] — Middle Big Endian in 32 bit. 5 [EM_ML_32] — Middle little Endian in 32 bit. 6 [EM_B_64] — Big Endian in 64 bit. 7 [EM_L_64] — Little Endian in 64 bit. 8 [EM_MB_64] — Middle Big Endian in 64 bit. 9 [EM_ML_64] — Middle little Endian in 64 bit.
dest	Area dove mettere i dati letti. Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.
Risultato	Rende: Valore / Significato 0 [MBC_OK] — Comando eseguito. -1 [MBC_INVAL] — Argomenti non validi. -2 [MBC_BUSY] — Risorsa occupata. -3 [MBC_PERM] — Permesso negato. -4 [MBC_CONN] — Connessione non riuscita. -5 [MBC_NOBUFS] — Buffer esauriti. -6 [MBC_NOMEM] — Memoria insufficiente. -7 [MBC_TX_ERR] — Errore di trasmissione.

© 2025 Robox SpA

Sintassi	**i32 mbcu_read_hold_reg (i32 mbc, i32 unitID, i32 offsHRStart, i32 nHR, i32 endian, dest)**

Valore	Significato
-8 [MBC_RX_ERR]	Errore di ricezione.
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4 (operazione fallita).
-14 [MBC_WRONG_ANSWER]	Risposta non valida.
-15 [MBC_INVAL_MBC]	Id *mbc* non valido.

Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione mbcu_read_inp_reg()

Funzione per la lettura di INPUT REGISTER su server modbus specificando lo unit ID.

Sintassi	**i32 mbcu_read_inp_reg (i32 mbc, i32 unitID, i32 offsIRStart, i32 nIR, i32 endian, dest)**
mbc	Valore handle reso da mbc_init 391 ().
unitID	Valore dello unit ID da usarsi.
offsIRStart	Indice iniziale dell'INPUT REGISTER.
nIR	Numero di INPUT REGISTER da leggere.
endian	Tipo di endian da usare:

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbcu_read_inp_reg (i32 mbc, i32 unitID, i32 offsIRStart, i32 nIR, i32 endian, dest)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [EM_B_16]</td><td>Big Endian nella singola word. (default)</td></tr>
<tr><td>1 [EM_L_16]</td><td>Little Endian nella singola word.</td></tr>
<tr><td>2 [EM_B_32]</td><td>Big Endian in 32 bit.</td></tr>
<tr><td>3 [EM_L_32]</td><td>Little Endian in 32 bit.</td></tr>
<tr><td>4 [EM_MB_32]</td><td>Middle Big Endian in 32 bit.</td></tr>
<tr><td>5 [EM_ML_32]</td><td>Middle little Endian in 32 bit.</td></tr>
<tr><td>6 [EM_B_64]</td><td>Big Endian in 64 bit.</td></tr>
<tr><td>7 [EM_L_64]</td><td>Little Endian in 64 bit.</td></tr>
<tr><td>8 [EM_MB_64]</td><td>Middle Big Endian in 64 bit.</td></tr>
<tr><td>9 [EM_ML_64]</td><td>Middle little Endian in 64 bit.</td></tr>
</table>
</td></tr>
<tr><td>dest</td><td>Area dove mettere i dati letti.
Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 mbcu_read_inp_reg (i32 mbc, i32 unitID, i32 offsIRStart, i32 nIR, i32 endian, dest)**

Valore	Significato
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4 (operazione fallita).
-14 [MBC_WRONG_ANSWER]	Risposta non valida.
-15 [MBC_INVAL_MBC]	Id *mbc* non valido.

Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione mbcu_write_coils()

Questa funzione scrive i COILS su un server modbus specificando lo unit ID.

Sintassi	**i32 mbcu_write_coils (i32 mbc, i32 unitID, i32 offsCoilStart, i32 nCoil, source)**
mbc	Valore handle reso da mbc_init 391 ().
unitID	Valore dello unit ID da usarsi.
offsCoilStart	Indice iniziale del COIL.
nCoil	Numero di COILS da scrivere. **NOTA**: Il massimo numero di COILS che possono essere scritti è 1968.
source	Area dove prendere i dati da scrivere. Può essere i16/u16/i32 o array di i16/u16/i32.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbcu_write_coils (i32 mbc, i32 unitID, i32 offsCoilStart, i32 nCoil, source)</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
<tr><td>-9 [MBC_TIMEOUT]</td><td>Timeout comando bloccante.</td></tr>
<tr><td>-10 [MBC_EXC_CODE_1]</td><td>Risposta con eccezione 1 (function code non supportato dal server).</td></tr>
<tr><td>-11 [MBC_EXC_CODE_2]</td><td>Risposta con eccezione 2 (indirizzo non valido).</td></tr>
<tr><td>-12 [MBC_EXC_CODE_3]</td><td>Risposta con eccezione 3 (quantità o valore non validi).</td></tr>
<tr><td>-13 [MBC_EXC_CODE_4]</td><td>Risposta con eccezione 4 (operazione fallita).</td></tr>
<tr><td>-14 [MBC_WRONG_ANSWER]</td><td>Risposta non valida.</td></tr>
<tr><td>-15 [MBC_INVAL_MBC]</td><td>Id mbc non valido.</td></tr>
</table></td></tr>
<tr><td>Validità</td><td>Task.</td></tr>
<tr><td>Note</td><td>La funzione è bloccante. Esce solo quando è stata completata l'operazione.</td></tr>
</table>

© 2025 Robox SpA

Funzione mbcu_write_multiple_coils()

Questa funzione scrive i COILS su un server Modbus specificando lo unit ID e usando il codice funzione Modbus delle scritture multiple corrispettive FC 15, diversamente da mbcu_write_coils() che può utilizzare anche FC 05.

Sintassi	**i32 mbcu_write_multiple_coils (i32 mbc, i32 unitID, i32 offsCoilStart, i32 nCoil, source)**
mbc	Valore handle reso da mbc_init().
unitID	Valore dello unit ID da usarsi.
offsCoilStart	Indice iniziale del COIL.
nCoil	Numero di COILS da scrivere. **NOTA**: Il massimo numero di COILS che possono essere scritti è 1968.
source	Area dove prendere i dati da scrivere. Può essere i16/u16/i32 o array di i16/u16/i32.
Risultato	Rende:

Valore	Significato
0 [MBC_OK]	Comando eseguito.
-1 [MBC_INVAL]	Argomenti non validi.
-2 [MBC_BUSY]	Risorsa occupata.
-3 [MBC_PERM]	Permesso negato.
-4 [MBC_CONN]	Connessione non riuscita.
-5 [MBC_NOBUFS]	Buffer esauriti.
-6 [MBC_NOMEM]	Memoria insufficiente.
-7 [MBC_TX_ERR]	Errore di trasmissione.
-8 [MBC_RX_ERR]	Errore di ricezione.
-9 [MBC_TIMEOUT]	Timeout comando bloccante.
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità

© 2025 Robox SpA

Sintassi	**i32 mbcu_write_multiple_coils (i32 mbc, i32 unitID, i32 offsCoilStart, i32 nCoil, source)**
	Valore / Significato o valore non validi). -13 [MBC_EXC_CODE_4] — Risposta con eccezione 4 (operazione fallita). -14 [MBC_WRONG_ANSWER] — Risposta non valida. -15 [MBC_INVAL_MBC] — Id *mbc* non valido.
Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione mbcu_write_multiple_regs()

Questa funzione scrive gli HOLDING REGISTER su un server modbus specificando lo unit ID e usando il codice funzione Modbus delle scritture multiple corrispettive FC 16, diversamente da mbcu_write_reg 436() che può utilizzare anche FC 06.

Sintassi	**i32 mbcu_write_multiple_regs (i32 mbc, i32 unitID, i32 offsHRStart, i32 nHR, i32 endian, source)**
mbc	Valore handle reso da mbc_init 391().
unitID	Valore dello unit ID da usarsi.
offsHRStart	Indice iniziale dell'HOLDING REGISTER.
nHR	Numero di HOLDING REGISTER da scrivere. **NOTA**: il massimo numero di HOLDING REGISTER che possono essere scritti è 123.
endian	Tipo di endian da usare: Valore / Significato 0 [EM_B_16] — Big Endian nella singola word. (default) 1 [EM_L_16] — Little Endian nella singola word. 2 [EM_B_32] — Big Endian in 32 bit.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbcu_write_multiple_regs (i32 mbc, i32 unitID, i32 offsHRStart, i32 nHR, i32 endian, source)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>3 [EM_L_32]</td><td>Little Endian in 32 bit.</td></tr>
<tr><td>4 [EM_MB_32]</td><td>Middle Big Endian in 32 bit.</td></tr>
<tr><td>5 [EM_ML_32]</td><td>Middle little Endian in 32 bit.</td></tr>
<tr><td>6 [EM_B_64]</td><td>Big Endian in 64 bit.</td></tr>
<tr><td>7 [EM_L_64]</td><td>Little Endian in 64 bit.</td></tr>
<tr><td>8 [EM_MB_64]</td><td>Middle Big Endian in 64 bit.</td></tr>
<tr><td>9 [EM_ML_64]</td><td>Middle little Endian in 64 bit.</td></tr>
</table>
</td></tr>
<tr><td>source</td><td>Area dove prendere i dati da scrivere.
Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
<tr><td>-9 [MBC_TIMEOUT]</td><td>Timeout comando bloccante.</td></tr>
<tr><td>-10 [MBC_EXC_CODE_1]</td><td>Risposta con eccezione 1 (function code non supportato dal server).</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 mbcu_write_multiple_regs (i32 mbc, i32 unitID, i32 offsHRStart, i32 nHR, i32 endian, source)**

Valore	Significato
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4 (operazione fallita).
-14 [MBC_WRONG_ANSWER]	Risposta non valida.
-15 [MBC_INVAL_MBC]	Id *mbc* non valido.

Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione mbcu_write_reg()

Questa funzione scrive gli HOLDING REGISTER su un server modbus specificando lo unit ID.

Sintassi	**i32 mbcu_write_reg (i32 mbc, i32 unitID, i32 offsHRStart, i32 nHR, i32 endian, source)**
mbc	Valore handle reso da mbc_init 391().
unitID	Valore dello unit ID da usarsi.
offsHRStart	Indice iniziale dell'HOLDING REGISTER.
nHR	Numero di HOLDING REGISTER da scrivere. **NOTA**: il massimo numero di HOLDING REGISTER che possono essere scritti è 123.
endian	Tipo di endian da usare:

Valore	Significato
0 [EM_B_16]	Big Endian nella singola word. (default)

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mbcu_write_reg (i32 mbc, i32 unitID, i32 offsHRStart, i32 nHR, i32 endian, source)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>1 [EM_L_16]</td><td>Little Endian nella singola word.</td></tr>
<tr><td>2 [EM_B_32]</td><td>Big Endian in 32 bit.</td></tr>
<tr><td>3 [EM_L_32]</td><td>Little Endian in 32 bit.</td></tr>
<tr><td>4 [EM_MB_32]</td><td>Middle Big Endian in 32 bit.</td></tr>
<tr><td>5 [EM_ML_32]</td><td>Middle little Endian in 32 bit.</td></tr>
<tr><td>6 [EM_B_64]</td><td>Big Endian in 64 bit.</td></tr>
<tr><td>7 [EM_L_64]</td><td>Little Endian in 64 bit.</td></tr>
<tr><td>8 [EM_MB_64]</td><td>Middle Big Endian in 64 bit.</td></tr>
<tr><td>9 [EM_ML_64]</td><td>Middle little Endian in 64 bit.</td></tr>
</table>
</td></tr>
<tr><td>source</td><td>Area dove prendere i dati da scrivere.
Può essere i16/u16/i32/float/real o array di i16/u16/i32/float/real.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0 [MBC_OK]</td><td>Comando eseguito.</td></tr>
<tr><td>-1 [MBC_INVAL]</td><td>Argomenti non validi.</td></tr>
<tr><td>-2 [MBC_BUSY]</td><td>Risorsa occupata.</td></tr>
<tr><td>-3 [MBC_PERM]</td><td>Permesso negato.</td></tr>
<tr><td>-4 [MBC_CONN]</td><td>Connessione non riuscita.</td></tr>
<tr><td>-5 [MBC_NOBUFS]</td><td>Buffer esauriti.</td></tr>
<tr><td>-6 [MBC_NOMEM]</td><td>Memoria insufficiente.</td></tr>
<tr><td>-7 [MBC_TX_ERR]</td><td>Errore di trasmissione.</td></tr>
<tr><td>-8 [MBC_RX_ERR]</td><td>Errore di ricezione.</td></tr>
<tr><td>-9 [MBC_TIMEOUT]</td><td>Timeout comando bloccante.</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 mbcu_write_reg (i32 mbc, i32 unitID, i32 offsHRStart, i32 nHR, i32 endian, source)**

Valore	Significato
-10 [MBC_EXC_CODE_1]	Risposta con eccezione 1 (function code non supportato dal server).
-11 [MBC_EXC_CODE_2]	Risposta con eccezione 2 (indirizzo non valido).
-12 [MBC_EXC_CODE_3]	Risposta con eccezione 3 (quantità o valore non validi).
-13 [MBC_EXC_CODE_4]	Risposta con eccezione 4 (operazione fallita).
-14 [MBC_WRONG_ANSWER]	Risposta non valida.
-15 [MBC_INVAL_MBC]	Id *mbc* non valido.

Validità	Task.
Note	La funzione è bloccante. Esce solo quando è stata completata l'operazione.

Funzione minimum()

Funzione che restituisce il minore fra i due valori forniti in ingresso.

Sintassi	**real minimum (real val1, real val2)**
val1	Primo valore da confrontare.
val2	Secondo valore da confrontare.
Risultato	Rende il minore fra *val1* e *val2*.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
velCalc = 60
```

© 2025 Robox SpA

```
velMin = 30

vel = minimum438(velCalc, velMin)
; vel = 30
```

Funzione mod()

Funzione che rende il resto della divisione fra variabili numeriche.

Sintassi	**real mod (real dividend, real divisor)**
dividend	Valore di cui si vuole il modulo
divisor	Valore modulante
Risultato	Rende il resto della divisione, calcolato con la logica seguente: mod = *dividend* - (*divisor* * (int)(*dividend* / *divisor*))
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
r(11) = 20
r(12) = mod439(r(11), 16)
; dopo l'esecuzione r(12) conterrà 4, ossia 4 = (20 - 16 * 1)

qMod = mod(ip(master), stepAx2)
; ip(master) varia da 0 ad infinito mentre qMod varia da 0 a stepAx2
```

Funzione mot_append()

Funzione per aggiungere dati all'insieme di punti della MOT.

Sintassi	**i32 mot_append (i32 motId, real posX, real posY, real velY, real accY [, real other=NULL])**
motId	Codice di riferimento della MOT (ottenuto dalla funzione mot_create443()).
posX	Valore della chiave di interpolazione associata al punto (ascissa, coordinata x). A seconda del valore di flag usato nella mot_create(), il valore di *posX* dovrà essere obbligatoriamente più grande rispetto

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mot_append (i32 motId, real posX, real posY, real velY, real accY [, real other=NULL])</td></tr>
<tr><td></td><td>all'ultimo inserito oppure potrà assumere qualsiasi valore.</td></tr>
<tr><td>posY</td><td>Array contenente i dati delle n ordinate (coordinate di posizione relative all'ascissa posX).</td></tr>
<tr><td>velY</td><td>Array contenente i dati delle n derivate prime (coordinate di velocità relative all'ascissa posX).

ATTENZIONE: Sono necessari se il tipo di interpolazione è uguale a 3 o a 5. Altrimenti specificare NULL.</td></tr>
<tr><td>accY</td><td>Array contenente i dati delle n derivate seconde (coordinate di accelerazione relative all'ascissa posX).

ATTENZIONE: Sono necessari se il tipo di interpolazione è uguale a 5. Altrimenti specificare NULL.</td></tr>
<tr><td>other</td><td>Struttura dati contenente i dati ausiliari. (opzionale, default NULL)

NOTA: Per gestire i dati ausiliari deve necessariamente essere utilizzata una struttura. Se si è scelto come interpolazione il tipo misto (flag = 0xF0000, vedi mot_create()), il primo campo della struttura other deve essere un i32 che specifichi il tipo di interpolazione del segmento (1=lineare, 2=cubica automatica, 4=pentica automatica). Non ci sono restrizioni per gli altri campi della struttura.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>1</td><td>Impossibile inserire il punto, contenitore pieno.</td></tr>
<tr><td>0</td><td>Operazione OK.</td></tr>
<tr><td>-1</td><td>motID non esiste.</td></tr>
<tr><td>-2</td><td>Chiave minore o uguale dell'ultima introdotta e flag di inserzione assente.</td></tr>
<tr><td>-3</td><td>Valore posY illegale (dimensione dati</td></tr>
</table></td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mot_append (i32 motId, real posX, real posY, real velY, real accY [, real other=NULL])</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td></td><td>minore di quello atteso).</td></tr>
<tr><td>-4</td><td>Valore other illegale (dimensione diversa da quella specificata in fase di creazione).</td></tr>
<tr><td>-5</td><td>Errore interno.</td></tr>
<tr><td>-6</td><td>Gestore MOT occupato (altra operazione in corso).</td></tr>
<tr><td>-10</td><td>Mancano dati della derivata prima.</td></tr>
<tr><td>-11</td><td>Mancano dati della derivata seconda.</td></tr>
<tr><td>-12</td><td>Tipo di interpolazione non gestito.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td></td></tr>
</table>

Esempi

Esempio di utilizzo:

© 2025 Robox SpA

```
function int writeMotParameter(int idMot, int numPointsCam, int numPar

; -------- WARNING ---------
; use this function only if you have extracted the cam points from a f
; master position in the first coloumn, the slave position int the sec
; the slave speed in the third and the slave acceleration in the fourt
; -------------------------

; this function insert the cam points previusly extracted from a file

; numPointsCam is equal to the number of row of the file
; numParameters is equal to the number of coloumn of the file

int idx
real velRef, velXrefQ, posX, posY, velY, accY

idx = 0
velXrefQ = velRef * velRef
; erase previous data
reso = mot_clear(443)(idMot)
if(reso < 0)
    return -1
endif

str = "Adding points to the MOT " # str_i(629)(idMot, 10, 16)
inf_report(724)(0x0100, str)

; Data saved
;-------------------------------------------

; in this example the master is assumed to go from 0 to 360 degree
; speed and acceleration data usually must be normalized, this is why
; are divided to velRef and velXrefQ

; initial data before 0 deg equal to previous cycle end, to well calcu
for(k = 4, k > 0, k--)

    idx =(numPointsCam - (k+1)) * numParameters

    posX = data[idx] - rMasterCycle
    posY = data[idx+1]
    velY = data[idx+2] / velRef    ; [mm/deg]
    accY = data[idx+3] / velXrefQ ; [mm/deg^2]

    select(numParameters)

        case 2
            reso = mot_append(439)(idMotot, posX, posY, NULL, NULL)
            break

        case 3
            reso = mot_append(idMot, posX, posY, velY, NULL)
            break

        case 4
            reso = mot_append(idMot, posX, posY, velY, accY)
            break

        default
            reso = mot_append(idMot, posX, posY, NULL, NULL)

    end_select
```

Funzione mot_clear()

Funzione per cancellare tutti i dati dell'insieme di punti della MOT.

Sintassi	**i32 mot_clear (i32 motId)**
motId	Codice di riferimento della MOT (ottenuto dalla funzione mot_create 443()).
Risultato	Rende: Valore: 0 — Operazione OK. -1 — *motID* non esiste. -5 — Errore interno. -6 — Gestore MOT occupato (altra operazione in corso).
Validità	Rule, Task.
Note	

Funzione mot_create()

Funzione per creare la MOT. Viene creata la zona di memoria per la gestione dell'insieme di punti.

Sintassi	**i32 mot_create (i32 nEl, i32 nInt, i32 sizeOfOther, i32 flag)**
nEl	Dimensione in numero di punti della MOT (>=1).
nInt	Numero delle grandezze da interpolare (ordinate) per ogni punto (>= 1).
sizeOfOther	Dimensione in byte delle informazioni ausiliarie associate al punto (>=0).
flag	Flag di gestione: Bit 0 — Valore esadecimale 0x00000001 — Se TRUE, permette l'inserimento dei punti con ascissa qualsiasi invece che con ascissa crescente (vedi

© 2025 Robox SpA

Sintassi	**i32 mot_create (i32 nEl, i32 nInt, i32 sizeOfOther, i32 flag)**

Bit	Valore esadecimale	Significato
		mot_append 439 ()).
1	0x00000002	Se TRUE, i dati aggiuntivi resi con la mv_mot_exec 470 () sono riferiti al punto finale anzichè quello iniziale del tratto di interpolazione.
4	0x00000010	Se TRUE, elimina i punti dopo l'esecuzione (viene eliminato il punto (n-1) quando l'esecuzione oltrepassa il punto n).
Codice a valore del tipo di interpolazione (Bit 16-:-19, 0x000F00xx):		
16	0x000100xx	Interpolazione lineare.
17	0x000200xx	Interpolazione cubica con calcolo automatico dei parametri.
16-17	0x000300xx	Interpolazione cubica calcolata specificando il valore della derivata prima nel punto (la velocità, se si tratta di interpolazione di posizioni).
18	0x000400xx	Interpolazione pentica (di quinto ordine) con calcolo automatico dei parametri.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mot_create (i32 nEl, i32 nInt, i32 sizeOfOther, i32 flag)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Significato</th></tr>
<tr><td>16-18</td><td>0x000500xx</td><td>Interpolazione pentica calcolata specificando i valori delle derivate prima e seconda nel punto (la velocità e l'accelerazione, se si tratta di interpolazione di posizioni).</td></tr>
<tr><td>16-17-18-19</td><td>0x000F00xx</td><td>Interpolazione mista (diversa per ogni tratto).

Il tipo viene specificato nella struttura other (vedi mot_append 439 ()). Il primo campo della struttura other usata nella mot_append() deve necessariamente essere di tipo i32 e contenere il tipo di interpolazione: 1=lineare, 2=cubica automatica, 4=pentica automatica.</td></tr>
</table>
</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>> 0</td><td>Operazione OK.

Il valore rappresenta il codice della MOT.</td></tr>
<tr><td>-1</td><td>Valore nEl illegale.</td></tr>
<tr><td>-2</td><td>Valore nInt illegale.</td></tr>
<tr><td>-3</td><td>Valore nEl eccessivo, memoria insufficiente.</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 mot_create (i32 nEl, i32 nInt, i32 sizeOfOther, i32 flag)**

Valore	Significato
-4	Errore interno.
-8	Selezionato interpolazione mista, ma la struttura *other* non esiste o ha dimensione < *sizeOfOther*.
-9	Codice di interpolazione sconosciuto.

Validità	Rule, Task.
Note	

NOTA: Ogni zona di memoria creata con mot_create() deve essere distrutta con una corrispondente mot_delete 447(); chiamate successive a mot_create() creano differenti zone di memoria.

Esempi

Esempio di utilizzo:

```
int flag, flag2
string appo

; MOT 1
;-------------------------------------------------------------------
flag.(0) = 1    ; data input enable
flag = flag r_or (0x50000)    ; pentic with speed and acc data
rMotID = mot_create 443(1000, 1, 0, flag)
if(rMotID < 0)
    str_format 627(appo, "Unable to create MOT error code :%d", rMotID)
    inf_report 724(0x100, appo)
    rMotID = 0
endif

; MOT 2
;-------------------------------------------------------------------
flag2.(0) = 1    ; data input enable
flag2 = flag2 r_or (0x40000)    ; pentic without speed and acc data
rMotID2 = mot_create(1000, 1, 0, flag2)
if(rMotID2 < 0)
    str_format(appo, "Unable to create MOT2 error code :%d", rMotID2)
    inf_report(0x100, appo)
    rMotID2 = 0
endif
```

© 2025 Robox SpA

Funzione mot_delete()

Funzione per cancellare la MOT generata precedentemente con la mot_create 443(). Viene cancellata la zona di memoria MOT contrassegnata da *motId*.

Sintassi	**i32 mot_delete (i32 motId)**
motId	Codice di riferimento della MOT (ottenuto dalla funzione mot_create 443()).
Risultato	Rende: Valore / Significato 0 / Operazione OK. -1 / *motID* non esiste. -5 / Errore interno. -6 / Gestore MOT occupato (altra operazione in corso).
Validità	Rule, Task.
Note	

Funzione mot_get()

Funzione per leggere dati relativi a un punto di una MOT.

Sintassi	**i32 mot_get (i32 motId, i32 nPoint, real posX, real posY=NULL, real velY=NULL, real accY=NULL [, other=NULL])**
motId	Codice di riferimento della MOT (ottenuto dalla funzione mot_create 443()).
nPoint	Numero del punto (0 = primo punto, -1 = ultimo punto).
posX	Valore (restituito dalla funzione) della chiave di interpolazione associata al punto (ascissa, coordinata x).
posY	Array di double destinato a contenere i dati (restituiti dalla funzione) delle n ordinate (coordinate di posizione relative all'ascissa *posX*). Specificare NULL (default) se non si desidera ottenere i dati. Se specificato <>NULL, la funzione mot_get() scarica nell'array i dati del punto *nPoint*.
velY	Array di double destinato a contenere i dati (restituiti dalla funzione) delle n velocità

© 2025 Robox SpA

Sintassi	**i32 mot_get (i32 motId, i32 nPoint, real posX, real posY=NULL, real velY=NULL, real accY=NULL [, other=NULL])**
	(coordinate di velocità relative all'ascissa *posX*). Specificare NULL (default) se non si desidera ottenere i dati. Se specificato <>NULL, la funzione mot_get() scarica nell'array i dati del punto *nPoint*.
accY	Array di double destinato a contenere i dati (restituiti dalla funzione) delle n accelerazioni (coordinate di accelerazione relative all'ascissa *posX)*. Specificare NULL (default) se non si desidera ottenere i dati. Se specificato <>NULL, la funzione mot_get() scarica nell'array i dati del punto *nPoint*.
other	Struttura dati destinata a contenere i dati ausiliari (restituiti dalla funzione). (opzionale, default NULL) Se specificato, la funzione mot_get() scarica nell'array i dati del punto *nPoint*.
Risultato	Rende:

Valore	Significato
1	Impossibile inserire il punto, contenitore pieno.
0	Operazione OK.
-1	*motID* non esiste.
-2	Valore *nPoint* illegale.
-3	Valore *posY* illegale (dimensione dati minore di quella attesa).
-4	Valore *other* illegale (dimensione diversa da quella specificata in fase di creazione).
-6	Gestore MOT occupato (altra operazione in corso).
-7	Vuoto (nessun punto).

Validità	Rule, Task.
Note	

© 2025 Robox SpA

Funzione mot_remove()

Funzione per cancellare uno o più punti. Elimina tutti i punti aventi distanza <= *epsi* dalla *posX* specificata.

Sintassi	**i32 mot_remove (i32 motId, real posX [, real epsi=1e-7])**
motId	Codice di riferimento della MOT (ottenuto dalla funzione mot_create 443()).
posX	Valore della chiave di interpolazione associata al punto da rimuovere (ascissa, coordinata x).
epsi	Raggio d'azione della cancellazione. (opzionale, default 1e-7)
Risultato	Rende:

Valore	Significato
>= 0	Numero di punti eliminati.
-1	*motID* non esiste.
-5	Errore interno.
-6	Gestore MOT occupato (altra operazione in corso).

Validità	Rule, Task.
Note	

Funzione mt_ats()

Funzione utile per costruire un semaforo atomico.

NOTA: Deprecata. Si consiglia l'utilizzo di mt_ats2 450().

Sintassi	**i32 mt_ats (i32 var [, i32 value=1])**
var	Variabile usata come semaforo.
value	Valore da imporre (opzionale, default 1). Usare *value*<>0 per prendere possesso del semaforo, *value*=0 per rilasciare il semaforo.
Risultato	Rende:

Valore	Significato
0	Semaforo già occupato.

© 2025 Robox SpA

Sintassi	**i32 mt_ats (i32 var [, i32 value=1])**
	Valore / Significato
	1 — Semaforo libero e impegnato correttamente dalla funzione.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
int mySem
if(mt_ats 449(mySem))
    ; semaforo libero: scrivo i dati in modo atomico
    ...
    mySem = 0   ; rilascio il semaforo
else
    ; il semaforo è occupato: aspettiamo
end_if
```

Funzione mt_ats2()

Funzione utile per costruire un semaforo atomico.

Sintassi	**i32 mt_ats2 (i32 var [, i32 value=1])**
var	Variabile usata come semaforo.
value	Valore da imporre (opzionale, default 1). Usare *value*<>0 per prendere possesso del semaforo, *value*=0 per rilasciare il semaforo.
Risultato	Rende: Valore / Significato <> 0 — Semaforo già occupato. Rappresenta il valore della variabile della precedente scrittura. 0 — Semaforo libero e impegnato correttamente dalla funzione.
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**i32 mt_ats2 (i32 var [, i32 value=1])**
Note	

Esempi

Esempio di utilizzo:

```
int mySem
int result

if(mt_ats2 450(mySem)=0)
    rr(23) = rr(49)
    rr(24) = rr(52)
    ...
    ; adesso rilascio il semaforo
    result = mt_ats2(mySem, 0)
else
    ; qualcun altro sta utilizzando
    ; il semaforo, facciamo altro
    ...
endif
```

Funzione mv_cam()

Funzione che permette di eseguire una polinomiale del terzo ordine.

Sono impostabili due camme e tramite un opportuno parametro *par* è possibile passare da una camma all'altra.

Funzionamento

In funzione di una grandezza entrante (master), viene calcolato il valore corrispondente della grandezza slave utilizzando una tabella di informazioni (frutto ad esempio di fogli di calcolo o programmi matematici) che è stata preventivamente caricata sulla flash-card del controllo (f_read_table_s 333()).

Mediante i parametri opzionali è possibile eseguire il passaggio tra due polinomiali o la somma pesate di esse.

Sintassi	**real mv_cam (real xCurr, i32 nSegm1, stru_matcam stru1 [, real par=0] [, i32 nSegm2=0] [, stru_matcam stru2=NULL])**
xCurr	Valore corrente del master (ascissa).
nPoints1	Numero di punti di cui è composta la prima camma polinomiale. Deve corrispondere alla lunghezza dell'array di *stru1*.
stru1	Array di strutture di tipo stru_matcam 875 contenente i valori della prima camma.

© 2025 Robox SpA

Sintassi	**real mv_cam (real xCurr, i32 nSegm1, stru_matcam stru1 [, real par=0] [, i32 nSegm2=0] [, stru_matcam stru2=NULL])**
	NOTA: Deve essere inizializzato come array.
par	Parametro miscelatore che permette di passare tra le due camme. (opzionale, default 0) Valore: 0 — Significato: Esegue la camma 1. Valore: 0 -:- 1 (estremi non compresi) — Significato: Esegue una sovrapposizione pesata delle due camme. Valore: 1 — Significato: Esegue la camma 2.
nPoints2	Numero di punti di cui è composta la seconda camma polinomiale. (opzionale, default 0) Deve corrispondere alla lunghezza dell'array di *stru2*.
stru2	Array di strutture di tipo stru_matcam 875 contenente i valori della seconda camma. (opzionale, default NULL) **NOTA**: Deve essere inizializzato come array.
Risultato	Valore aggiornato del risultato, corrispondente alle coordinate y della camma. Il risultato viene calcolato come: dx1 = *xCurr* - xInitial1 dx2 = *xCurr* - xInitial2 risultato = (((a1*dx1^3) + (b1*dx1^2) + (c1*dx1) + d1) * (1-*par*)) + (((a2*dx2^3) + (b2*dx2^2) + (c2*dx2) + d2) * *par*)
Validità	Rule.
Note	I parametri a, b, c, d devono essere forniti in ingresso alla funzione.

Diagnostica

- Genera allarme 9900 ill. Arg A:5 T:x ST:xxx se *par* <> 0 e *nSegm2* = 0

Esempi

Esempio di utilizzo:

```
IP(AX_2) = mv_cam 451(IP(1), 5, traiettoria)
```

© 2025 Robox SpA

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Funzione mv_crimper()

Funzione che viene utilizzata per realizzare un movimento sincronizzato, in cui un asse (slave) si muove in continuo legato alla quota di un altro asse (master).

Funzionamento

Il movimento dello slave comprende una prima fase (aggancio) in cui la sua velocità è proporzionale a quella del master, seguita da una seconda fase (recupero) durante la quale lo slave esegue un movimento tale da ripresentarsi, a fine ciclo, nella giusta posizione di fase e con la giusta velocità per poter iniziare, senza perturbazioni, il successivo ciclo.

Il movimento di recupero viene calcolato impiegando una legge di moto sinusoidale, con velocità di picco calcolata in funzione dei parametri di lavoro. L'asse master può essere un asse controllato, oppure un riferimento esterno (es. un encoder di linea) o anche una quota virtuale costruita e fatta evolvere da parte del software applicativo R3.

L'asse slave deve essere necessariamente un asse controllato.

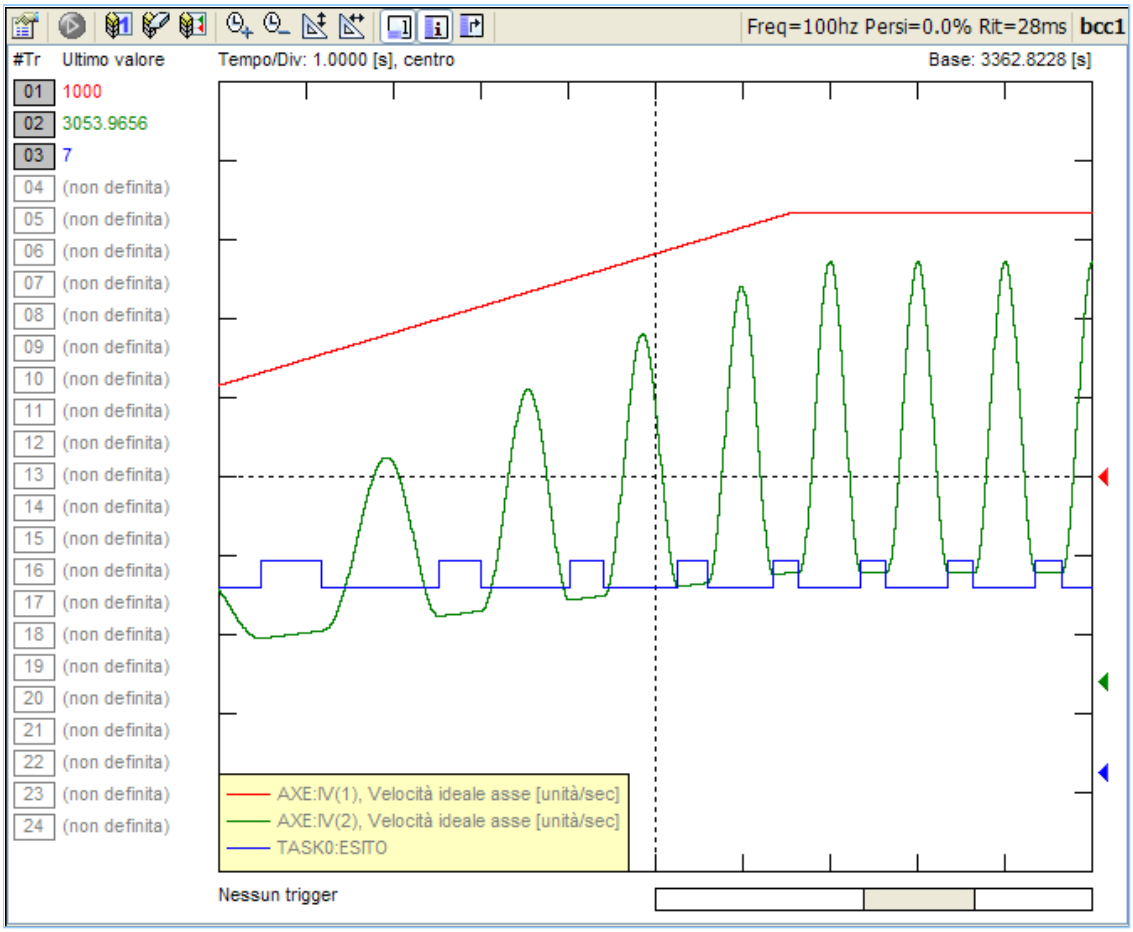
Operativamente bisogna chiamare, a master fermo, la funzione con il codice M_INITIALIZING ed inizializzare tutti i parametri della funzione, eseguire quindi un posizionamento sulla quota resa dalla funzione ed abilitare poi il movimento del master.

<table>
<tr><td>Sintassi</td><td>real mv_crimper (i32 return, stru_crimper stru)</td></tr>
<tr><td>return</td><td>Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63).
<table>
<tr><td>Valore</td><td>Descrizione</td></tr>
<tr><td>1 [M_INITIALING]</td><td>Alla prima chiamata della funzione l'utente deve impostare lo stato a M_INITIALING.</td></tr>
<tr><td>7 [M_HOMING]</td><td>È in corso la missione di recupero.</td></tr>
<tr><td>8 [M_SYNCHRONIZED]</td><td>Lo slave è agganciato al master.</td></tr>
</table></td></tr>
<tr><td>stru</td><td>Struttura di tipo stru_crimper 839 contenente i parametri relativi alla movimentazione.</td></tr>
<tr><td>Risultato</td><td>Valore aggiornato di posizione dell'asse slave.</td></tr>
<tr><td>Validità</td><td>Rule.</td></tr>
<tr><td>Note</td><td></td></tr>
</table>

Esempi

© 2025 Robox SpA

Esempio su oscilloscopio di un possibile output della funzione:



Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Funzione mv_follow()

Funzione che viene utilizzata per realizzare inseguimenti di un asse (slave) nei confronti di un altro (master).

SUGGERIMENTO: Si consiglia l'utilizzo della più performante funzione mv_follow2 458().

Funzionamento

L'asse slave (inseguitore) viene gestito essenzialmente tramite 4 fasi successive (vedere stru_follow 844 per la descrizione delle keyword):

- M_POSITIONING: l'asse slave si porta alla posizione *qParkSlave*, utilizzando i parametri di velocità *velMaxSlave* e l'accelerazione *accMaxSlave*.
- M_WAITING: l'asse rimane fermo; vengono eseguiti i calcoli per verificare se è arrivato il momento di partire in inseguimento. In questa fase vengono effettuati i controlli di congruenza che consentono la partenza dell'inseguimento (direzione master e velocità master, quota *qSyncMaster* rispetto alla quota master attuale, quota *qSyncSlave*

© 2025 Robox SpA

rispetto alla quota slave attuale). Viene inoltre verificato che l'accelerazione asse slave sia sufficiente. Viene dato codice M_LOW_ACCELERATION in caso contrario.

- M_FOLLOWING: l'asse si muove nella direzione specificata dal parametro *slaveDirection*, eseguendo un movimento ad accelerazione costante calcolato per raggiungere la velocità dell'asse master in corrispondenza della quota slave *qSyncSlave* e della quota master *qSyncMaster*. L'inseguimento tiene conto anche di eventuali variazioni di velocità del master durante il ciclo.
- M_REACHED: l'asse slave si mantiene agganciato al master, in base alla sua velocità corrente, fino a che sia stata raggiunta o superata la quota slave *qSyncOut*. A questo punto vengono incrementate di *masterCycle* la grandezza *qSyncMaster* e di *slaveCycle* le grandezze *qPark*, *qSyncSlave*, *qSyncOut* per predisporsi al ciclo successivo. La variabile fase viene impostata a M_POSITIONING e il ciclo ricomincia.

Il parametro *kSlaveMaster* risulta molto utile in quei casi in cui l'unità di misura dell'asse slave è differente da quella dell'asse master (caso tipico un master espresso in mm e uno slave in gradi). Specificando il fattore K esistente fra i due assi è possibile eseguire l'inseguimento come se le grandezze fossero omogenee fra loro.

Nel caso si desideri realizzare un inseguimento ciclico con assi monodirezionali (es. master convogliatore, slave ganascia di taglio a due pale) occorre specificare un valore di passo per entrambi gli assi (nell'esempio, *MasterCycle* = lunghezza prodotto, *SlaveCycle* = 180 gradi). Questi valori vengono utilizzati dalla funzione per incrementare automaticamente le quote di sincronismo (master e slave), la quota di fine inseguimento e la quota di attesa. Gli incrementi avvengono tutti insieme al momento dell'abbandono della fase di sincronismo.

Se l'asse slave è invece dotato di moto alternativo (es. asse di taglio mosso da vite senza fine) occorre specificare per esso *SlaveCycle* = 0. In questo modo, alla fine del ciclo, l'asse slave ritorna alla posizione *qPark* (che è la medesima per tutti i cicli) mentre il prossimo sincronismo sull'asse master (*qSyncMaster*) viene incrementato.

Se si desidera effettuare un inseguimento "one shot", cioè senza ripetizione automatica, è sufficiente specificare *MasterCycle* = 0. In questo modo, una volta terminato il primo ciclo di inseguimento, non si ripresentano le condizioni per partire con un nuovo ciclo fino a quando l'utente non inserisce direttamente un valore aggiornato di *qSyncMaster*.

La prima volta che la funzione viene chiamata, è necessario aver preventivamente caricato nella struttura stru_follow una serie di valori opportuni. Anche il parametro *status* deve essere stato inizializzato a M_POSITIONING (posizionamento iniziale). Questo parametro può essere utilmente consultato da parte del software applicativo R3 durante lo svolgimento del ciclo. È compito del software applicativo garantire che al momento della chiamata della funzione le tabelle contengano dati coerenti.

Sintassi	**real mv_follow (i32 return, stru_follow stru)**
return	Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63).

Valore	Descrizione
2 [M_POSITIONING]	Alla prima chiamata della funzione l'utente deve impostare lo stato a M_POSITIONING. L'asse slave si sta portando alla

© 2025 Robox SpA

Sintassi	**real mv_follow (i32 return, stru_follow stru)**
	<table><tr><th>Valore</th><th>Descrizione</th></tr><tr><td></td><td>posizione di parcheggio.</td></tr><tr><td>3 [M_WAITING]</td><td>L'asse slave sta aspettando di partire.</td></tr><tr><td>4 [M_WAITING_NEXT_CYCLE]</td><td>Non ci sono le condizioni per partire. Aspetto il prossimo appuntamento.</td></tr><tr><td>6 [M_REACHED]</td><td>L'asse slave è agganciato.</td></tr></table>
stru	Struttura di tipo stru_follow 844 contenente i parametri relativi alla movimentazione.
Risultato	Valore aggiornato di posizione dell'asse slave.
Validità	Rule.
Note	

L'inseguimento può essere singolo oppure ripetuto ciclicamente, ma in ogni caso deve esistere una fase di attesa inizio inseguimento, in cui l'asse è fermo.

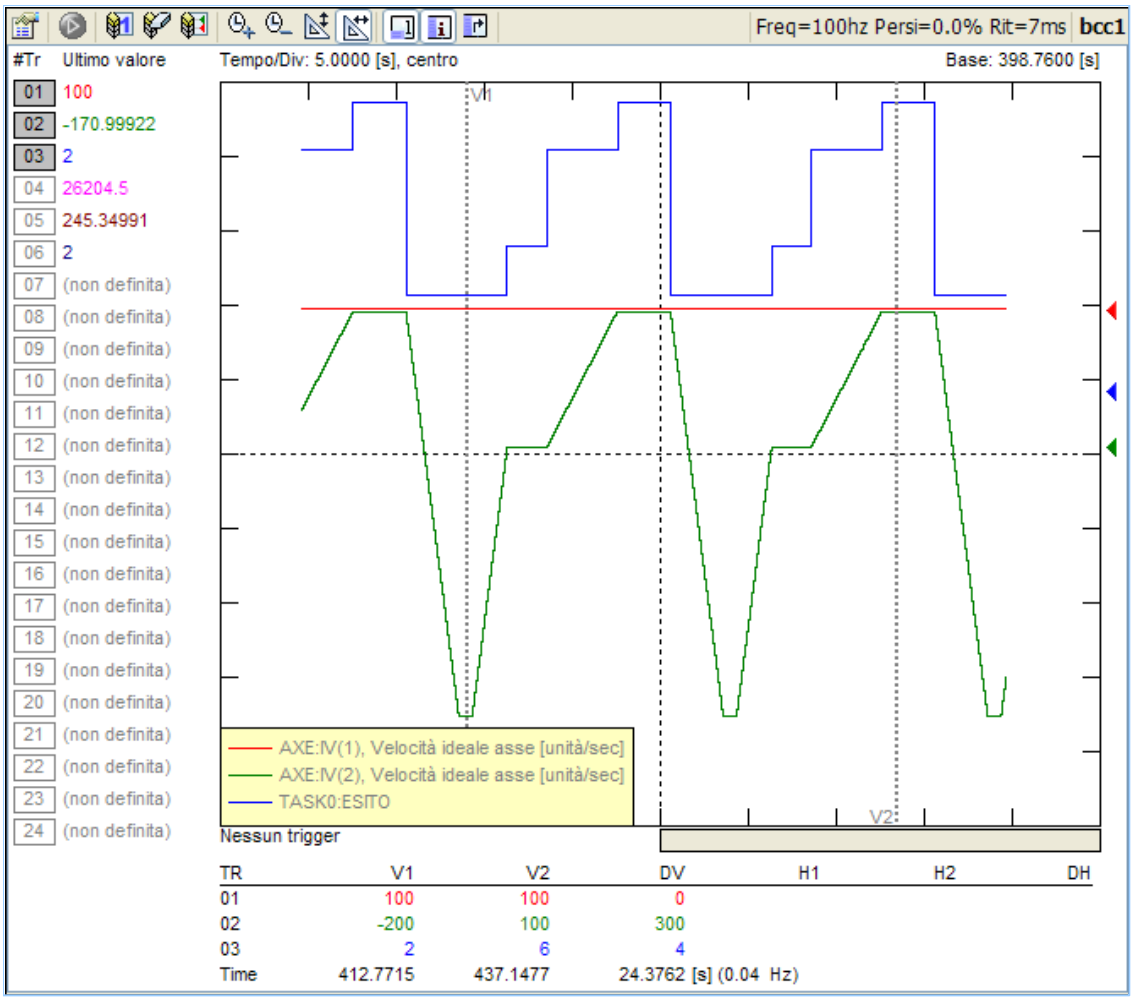
Il master potrebbe essere un asse fisico controllato, ma anche un semplice riferimento esterno (es. un encoder di linea) o anche una quota virtuale costruita e fatta evolvere da parte del software applicativo.

L'asse slave deve necessariamente essere un asse controllato.

Esempi

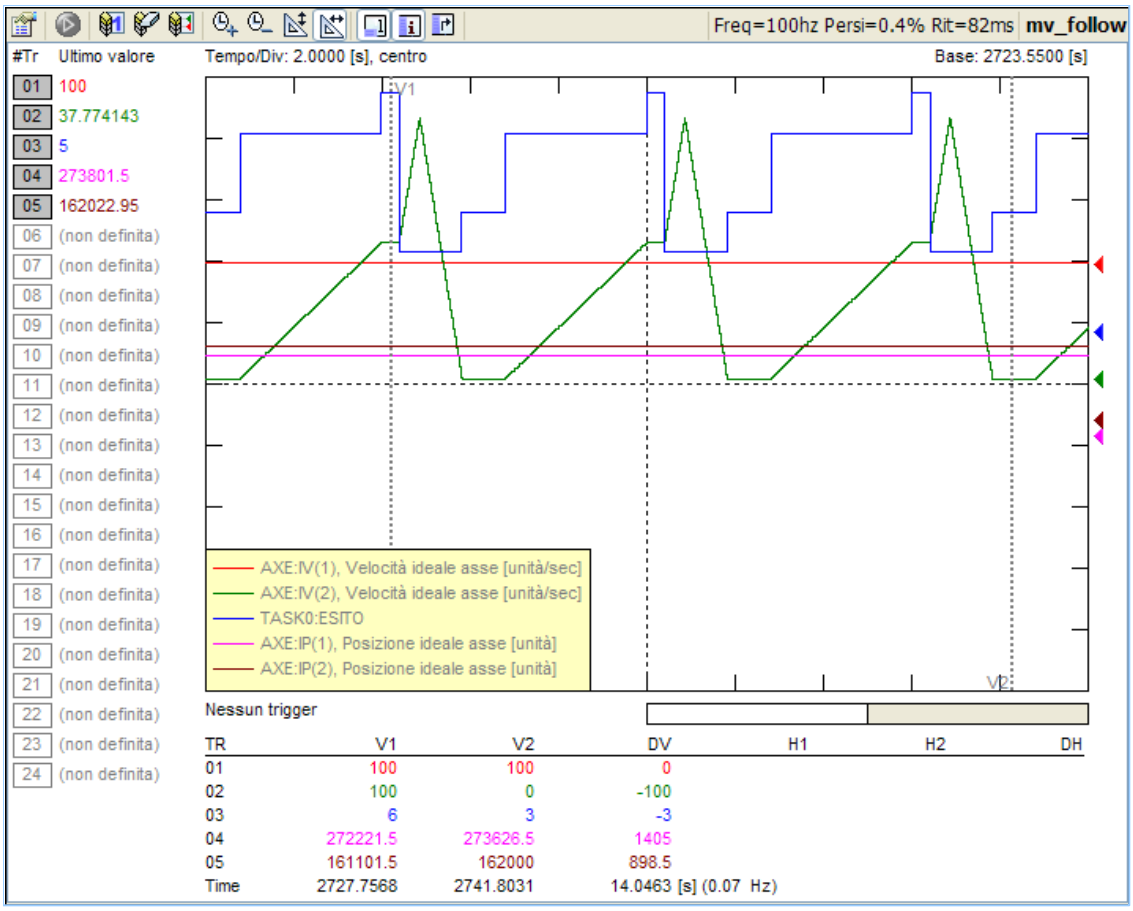
Esempio su oscilloscopio di un inseguimento lineare (lo slave va avanti e indietro):

© 2025 Robox SpA



Esempio su oscilloscopio di un inseguimento ciclico (slave con moto rotatorio):

© 2025 Robox SpA



Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Funzione mv_follow2()

Questa funzione viene utilizzata per realizzare inseguimenti di uno SLAVE nei confronti di un MASTER, entrambi non associati ad un asse. La funzione esegue movimenti jerkati calcolati con una curva pentica, camma della posizione dell'asse master.

Premessa

L'inseguimento può essere singolo oppure ripetuto ciclicamente a ciclo fisso o ciclo variabile.

Il master potrebbe essere un asse fisico controllato, ma anche un semplice riferimento esterno (es. un encoder di linea) o anche una quota virtuale costruita e fatta evolvere da parte del software applicativo.

In ogni caso, durante l'esecuzione della funzione, il master deve evolvere sempre nella stessa direzione.

Lo slave potrebbe essere un asse fisico controllato, ma anche una semplice grandezza costruita e fatta evolvere da parte del software applicativo. Esso può essere un asse lineare di escursione limitata oppure un asse ciclico (ad esempio rotativo).

Tipologie della funzione follow2

La sintassi della follow2 varia in funzione di:

- MASTER e SLAVE entrambi non associati ad un asse -> mv_follow2()

© 2025 Robox SpA

- MASTER non associato ad un asse, SLAVE associato ad un asse -> mva_follow2 467()
- MASTER e SLAVE entrambi associati ad un asse (caso tipico) -> mva_follow2_m 469()

Funzionamento

L'asse slave (inseguitore) viene gestito secondo le fasi seguenti:

- M_POSITIONING: la fase di posizionamento iniziale deve essere richiesta dall'utente impostando gli appositi bit del parametro *command* della struttura di riferimento. La richiesta di posizionamento deve necessariamente essere invocata a partire da una condizione in cui lo slave è fermo sia in velocità che in accelerazione (controllare che il bit della maschera ax_in_mov2 dell'asse corrispondente sia a 0). Se è stato richiesto il posizionamento alla quota di parcheggio, l'asse slave si porta alla posizione *qParkSlave*, utilizzando i parametri di cinematica *velPositSlave*, *accPositSlave* e *jerkPositSlave*. Nel caso in cui il movimento della funzione venisse interrotto (per esempio in seguito ad una condizione di emergenza), se il master nel frattempo non si è spostato di tanto, è possibile eseguire il posizionamento alla posizione del ciclo relativa alla quota attuale del master. Per fare questo utilizzare il bit corrispondente del parametro *command*. Durante la fase di posizionamento alla quota relativa del master è fondamentale che il master sia fermo. L'utente può anche eseguire il posizionamento in modo differente, calcolato tramite una funzione esterna, l'importante è che si preoccupi di portare l'asse slave alla posizione corretta prima di utilizzare la funzione mv_follow2(). Se alla prima esecuzione della funzione (con richiesta di posizionamento) lo slave si trova già nella posizione corretta è possibile lasciare a 0 i parametri di cinematica *velPositSlave*, *accPositSlave* e *jerkPositSlave*
- M_WAITING: l'asse rimane fermo alla quota di parcheggio *qParkSlave*; vengono eseguiti i calcoli per verificare se è arrivato il momento di partire in inseguimento in funzione della quota attuale del master, della quota *qSyncMaster* e delle posizioni impostate per le varie fasi dello slave. Se è troppo tardi per partire durante il ciclo attuale, la fase M_WAITING passa a M_WAITING_NEXT_CYCLE
- M_FOLLOWING: l'asse parte in inseguimento eseguendo un movimento calcolato in modo tale da raggiungere la velocità dell'asse master in corrispondenza della quota slave *qSyncSlave* e della quota master *qSyncMaster*. Se l'asse slave è lineare viene eseguita una rampa jerkata semplice (in cui lo spazio percorso dal master è doppio rispetto allo spazio percorso dallo slave). Se invece l'asse slave è ciclico il movimento eseguito in questa fase dipende da tutti gli altri parametri che compongono il ciclo; in questo caso la rampa di inseguimento può anche presentare sovra-elongazioni
- M_SYNCHRONIZED: l'asse slave si mantiene agganciato al master, fino a che sia stata raggiunta o superata la quota slave *qSyncOut*. Durante questa fase la velocità dell'asse slave è data dalla seguente formula: *vSlave* = *velMaster* * *kSlaveMaster*

Al termine della fase sincronizzata il comportamento differisce in funzione di come è stata configurata la struttura stru_follow2:

- Nel caso di slave lineare (*slaveCycle* = 0) si passa alla fase di decelerazione M_DECEL in cui lo slave rallenta fino a fermarsi percorrendo lo spazio di frenata *spaceStopSlave* ed eseguendo una rampa jerkata semplice (spazio percorso dal master doppio rispetto allo spazio percorso dallo slave).
  Al termine della fase di decelerazione lo slave lineare passa alla fase di ritorno alla posizione di parcheggio M_HOMING nella quale viene eseguito un movimento con profilo di velocità a campana tale da consentire di raggiungere la posizione di parcheggio *qParkSlave* e mantenerla per la durata *spaceWaitInPark* prima dell'inizio della successiva fase di inseguimento
- Nel caso di slave ciclico (*slaveCycle* > 0) con fermata alla quota *qParkSlave* abilitata durante il ciclo (vedi bit corrispondente del parametro *command*) si passa direttamente alla fase di M_HOMING nella quale viene eseguito un movimento calcolato in modo tale

© 2025 Robox SpA

da consentire di raggiungere la posizione di parcheggio *qParkSlave* e mantenerla per la durata *spaceWaitInPark* prima dell'inizio della successiva fase di inseguimento

- Nel caso di slave ciclico (*slaveCycle* > 0) con fermata alla quota *qParkSlave* disabilitata durante il ciclo (vedi bit corrispondente del parametro *command*) si passa direttamente alla fase successiva di inseguimento senza fermarsi alla quota *qParkSlave*. Il movimento viene calcolato in modo tale da partire in velocità e raggiungere il sincronismo successivo sempre in velocità alle quote *qSyncSlave* e *qSyncMaster*. Se lo spazio mancante al master per raggiungere la quota di sincronismo fosse significativamente maggiore dello spazio mancante allo slave, lo slave potrebbe rallentare e fermarsi ad una quota intermedia che potrebbe anche essere diversa dalla quota *qParkSlave*

Se il ciclo del master non ha durata fissa (vedi il bit corrispondente del parametro *command*) è necessario impostare ad ogni ciclo il nuovo valore di *qSyncMaster* per tempo: le fasi utili per impostare la quota di sincronismo *qSyncMaster* relativa al ciclo successivo sono la fase di inseguimento M_FOLLOWING e la fase sincronizzata M_SYNCHRONIZED relative al ciclo corrente.

Se la quota *qSyncMaster* viene impostata oltre la fine della fase sincronizzata M_SYNCHRONIZED l'asse slave andrà verso la quota di parcheggio *qParkSlave*.

Anche in caso di master a ciclo di durata variabile è obbligatorio impostare il parametro *masterCycle* in modo tale da fissare una durata minima del ciclo master. La durata minima serve per limitare la velocità della fase di recupero e, nel caso in cui la quota *qSyncMaster* non viene impostata per tempo, per stabilire la velocità della fase di homing.

In caso di slave ciclico con sosta alla quota di parcheggio *qParkSlave* disabilitata durante il ciclo è possibile recuperare una quota *qSyncMaster* assegnata oltre il tempo limite abilitando il bit corrispondente del parametro *command*. In questo caso, appena la nuova quota di sincronismo *qSyncMaster* viene impostata, verrà eseguito (se possibile) un raccordo al volo tra il movimento verso la quota di parcheggio *qParkSlave* e il movimento di inseguimento verso il nuovo sincronismo.

Il parametro *kSlaveMaster* risulta molto utile in quei casi in cui l'unità di misura dell'asse slave è differente da quella dell'asse master (caso tipico un master espresso in mm e uno slave in gradi). Specificando il fattore K esistente fra i due assi è possibile eseguire l'inseguimento come se le grandezze fossero omogenee fra loro. Inoltre, variando leggermente il parametro *kSlaveMaster* rispetto al suo valore nominale, è possibile eseguire il sincronismo a velocità differenti (ad esempio se l'asse slave è la ganascia che salda e taglia il film di una macchina incartatrice è utile ridurre la velocità dell'asse slave rispetto a quella del master in modo tale da evitare di strappare il film durante la saldatura).

La prima volta che la funzione mv_follow2() viene chiamata, è necessario aver preventivamente caricato nella struttura stru_follow2 una serie di valori opportuni. Inoltre, per avere la certezza di quale sarà la fase iniziale, è opportuno eseguire una richiesta di posizionamento tramite gli appositi bit del parametro *command*.

Se alcuni parametri che caratterizzano il ciclo non sono coerenti, la funzione restituisce l'esito negativo M_IN_ALARM. In questo caso è possibile testare il parametro *errorCode* per avere un'indicazione di cosa c'è di sbagliato.

Se vengono impostati i parametri di *maxSpe* e *maxAcc* e se non vengono disabilitati i test sui bit relativi del parametro *command*, ad ogni esecuzione della funzione mv_follow2() viene eseguita una verifica per evitare di superare i limiti impostati. In caso di limiti superati la funzione ritorna l'esito negativo M_IN_ALARM ed esegue una rampa di arresto lineare con accelerazione *maxAcc*. Anche in questo caso è possibile consultare il parametro *errorCode* per capire in quale fase la velocità e/o l'accelerazione sono state superate.

© 2025 Robox SpA

Sintassi	**i32 mv_follow2 (stru_follow2 stru1 [, real period=0 [, stru_follow2info stru2=NULL])**
stru1	Struttura di tipo stru_follow2 846 contenente i parametri relativi alla movimentazione.
period	Intervallo di tempo tra due esecuzioni consecutive di questa funzione. (opzionale, default 0) Se 0 si assume pari a SI.
stru2	Struttura di tipo stru_follow2info 854 dove salvare i dati accessori (out). (opzionale, default NULL)
Risultato	Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63).

Valore	Descrizione
2 [M_POSITIONING]	Posizionamento iniziale richiesto impostando i bit relativi del parametro *command*.
3 [M_WAITING]	L'asse slave è in attesa di partire.
4 [M_WAITING_NEXT_CYCLE]	Non ci sono le condizioni per partire, aspetto il prossimo appuntamento.
5 [M_FOLLOWING]	L'asse slave è partito all'inseguimento del master.
8 [M_SYNCHRONIZED]	L'asse slave si è agganciato ed è sincronizzato al master.
17 [M_DECEL]	L'asse slave ha terminato la fase di sincronismo e sta decelerando.
7 [M_HOMING]	L'asse slave sta ritornando al punto di parcheggio.
-6 [M_IN_ALARM]	La funzione è in allarme, vedi *errorCode*.

Validità	Rule.
Note	La configurazione della funzione viene fatta attraverso la struttura dati ausiliaria.

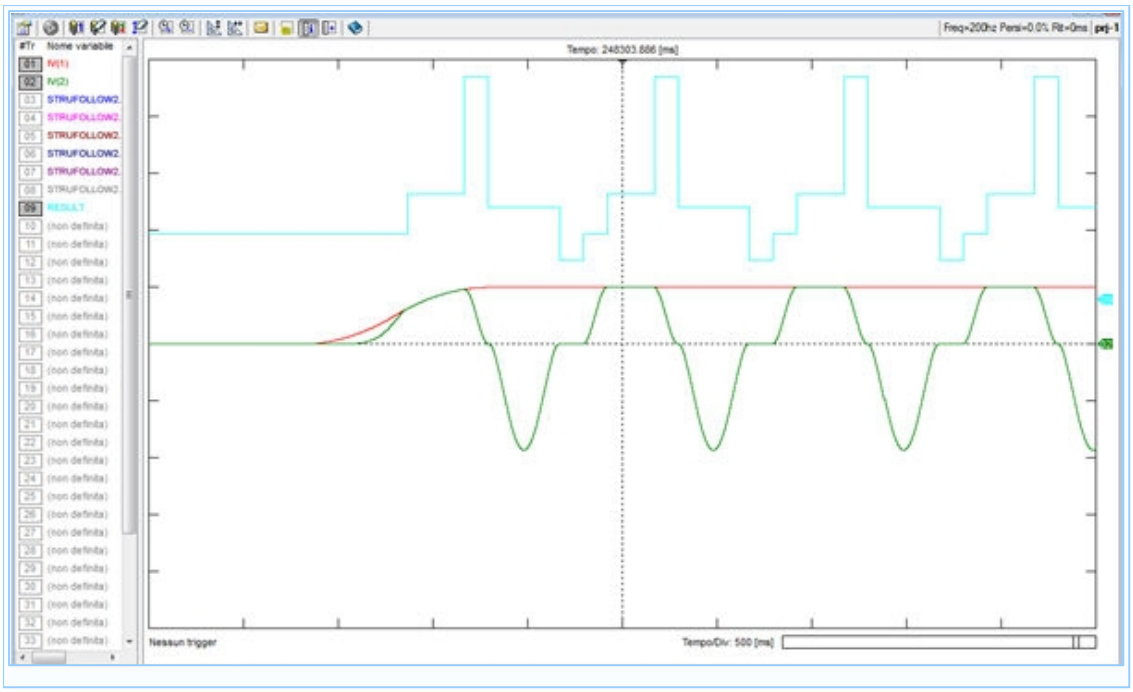
© 2025 Robox SpA

Esempi

Esempio di utilizzo (master a ciclo fisso e slave lineare):

```
stru_follow2 struFollow2

rule 1
motion
if(first_time 346())
    struFollow2.command.0        = 0      ; 1 = forzatura arresto slave
    struFollow2.command.1        = 1      ; 1 = posizionamento iniziale
    struFollow2.command.2        = 0      ; 1 = posizionamento iniziale
    struFollow2.command.16       = 0      ; 0 = master a ciclo fisso, 1
    struFollow2.command.17       = 0      ; 1 = disabilitazione test max
    struFollow2.command.18       = 0      ; 1 = disabilitazione test max
    struFollow2.command.19       = 0      ; 1 = qSyncMaster allineata a
    struFollow2.command.20       = 0      ; 1 = abilitazione fermata a q
    struFollow2.command.21       = 0      ; 1 = forzatura movimento slav
    struFollow2.command.22       = 0      ; 1 = abilita recupero del sin
    struFollow2.masterCycle      = 400    ; [mm] Master a ciclo fisso di
    struFollow2.qSyncMaster      = 0      ; [mm] Quota di sync master al
    struFollow2.spaceWaitInPark = 50      ; [mm] Spazio master con slave
    struFollow2.slaveCycle       = 0      ; [mm] Ciclo slave (0 = slave
    struFollow2.qParkSlave       = 0      ; [mm] Quota di parcheggio sla
    struFollow2.qSyncSlave       = 25     ; [mm] Quota di sync slave
    struFollow2.qQutSlave        = 125    ; [mm] Quota di fine sync slav
    struFollow2.spaceStopSlave   = 25     ; [mm] Spazio di arresto slave
    struFollow2.velPositSlave    = 100    ; [mm/s] Velocità di posiziona
    struFollow2.accPositSlave    = 200    ; [mm/s^2] Accelerazione di po
    struFollow2.jerkPositSlave   = 400    ; [mm/s^3] Jerk di posizioname
    struFollow2.maxSpe           = 1000   ; [mm/s] Velocità massima slav
    struFollow2.maxAcc           = 10000  ; [mm/s^2] Accelerazione massi
    struFollow2.kSlaveMaster     = 1      ; k di conversione master -> s
endif
struFollow2.qMaster   = ip(1)
struFollow2.velMaster = iv(1)
struFollow2.accMaster = ia(1)
struFollow2.qSlave    = ip(2)
struFollow2.velSlave  = iv(2)
struFollow2.accSlave  = ia(2)
result = mv_follow2 458(struFollow2)
set_kinematics 595(ax2, struFollow2.qSlave, struFollow2.velSlave, struFo
end_motion
end_rule
```

© 2025 Robox SpA

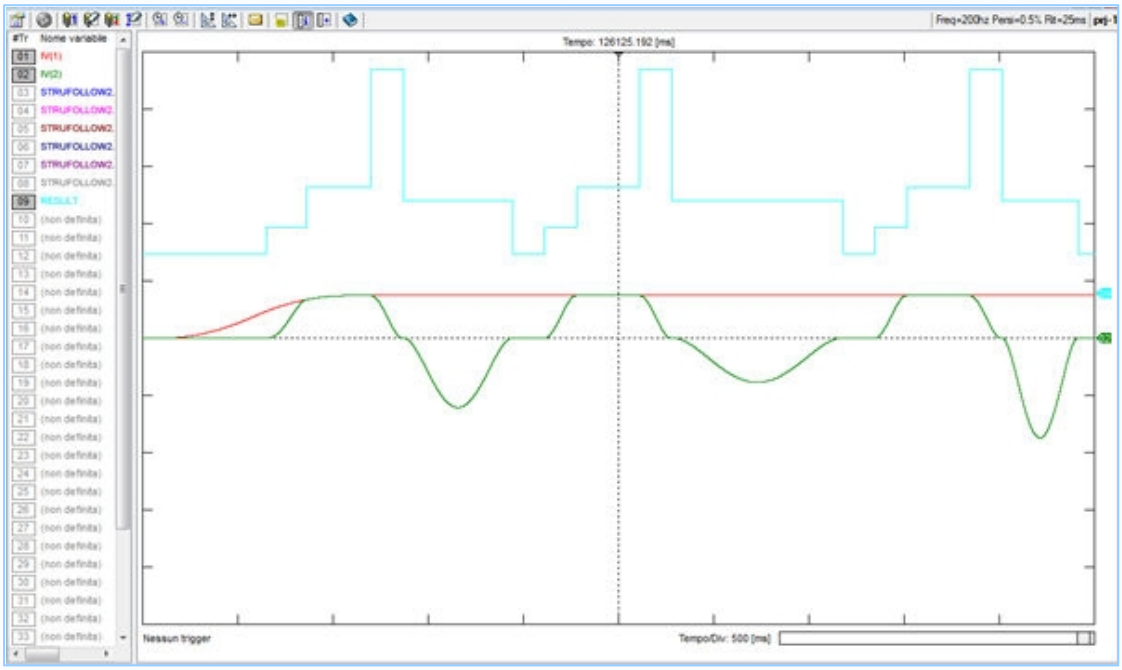


Esempio di utilizzo (master a ciclo variabile e slave lineare):

```
stru_follow2 struFollow2

rule 1
motion
if(first_time())
    struFollow2.command.0        = 0      ; 1 = forzatura arresto slave
    struFollow2.command.1        = 1      ; 1 = posizionamento iniziale
    struFollow2.command.2        = 0      ; 1 = posizionamento iniziale
    struFollow2.command.16       = 1      ; 0 = master a ciclo fisso, 1
    struFollow2.command.17       = 0      ; 1 = disabilitazione test max
    struFollow2.command.18       = 0      ; 1 = disabilitazione test max
    struFollow2.command.19       = 0      ; 1 = qSyncMaster allineata a
    struFollow2.command.20       = 0      ; 1 = abilitazione fermata a q
    struFollow2.command.21       = 0      ; 1 = forzatura movimento slav
    struFollow2.command.22       = 0      ; 1 = abilita recupero del sin
    struFollow2.masterCycle      = 300    ; [mm] Master a ciclo variabil
    struFollow2.spaceWaitInPark  = 50     ; [mm] Spazio master con slave
    struFollow2.slaveCycle       = 0      ; [mm] Ciclo slave (0 = slave
    struFollow2.qParkSlave       = 0      ; [mm] Quota di parcheggio sla
    struFollow2.qSyncSlave       = 25     ; [mm] Quota di sync slave
    struFollow2.qQutSlave        = 125    ; [mm] Quota di fine sync slav
    struFollow2.spaceStopSlave   = 25     ; [mm] Spazio di arresto slave
    struFollow2.velPositSlave    = 100    ; [mm/s] Velocità di posiziona
    struFollow2.accPositSlave    = 200    ; [mm/s^2] Accelerazione di po
    struFollow2.jerkPositSlave   = 400    ; [mm/s^3] Jerk di posizioname
    struFollow2.maxSpe           = 1000   ; [mm/s] Velocità massima slav
    struFollow2.maxAcc           = 10000  ; [mm/s^2] Accelerazione massi
```

© 2025 Robox SpA

```
    struFollow2.kSlaveMaster    = 1     ; k di conversione master -> s
endif
struFollow2.qMaster   = ip(1)
struFollow2.velMaster = iv(1)
struFollow2.accMaster = ia(1)
struFollow2.qSlave    = ip(2)
struFollow2.velSlave  = iv(2)
struFollow2.accSlave  = ia(2)
; Quota di sync master ricavata da un OB di tipo rc_elementlocation
struFollow2.qSyncMaster = ip(1) + buffer.get_distance_to_end(0)
result = mv_follow2(struFollow2)
set_kinematics(ax2, struFollow2.qSlave, struFollow2.velSlave, struFoll
end_motion
end_rule
```

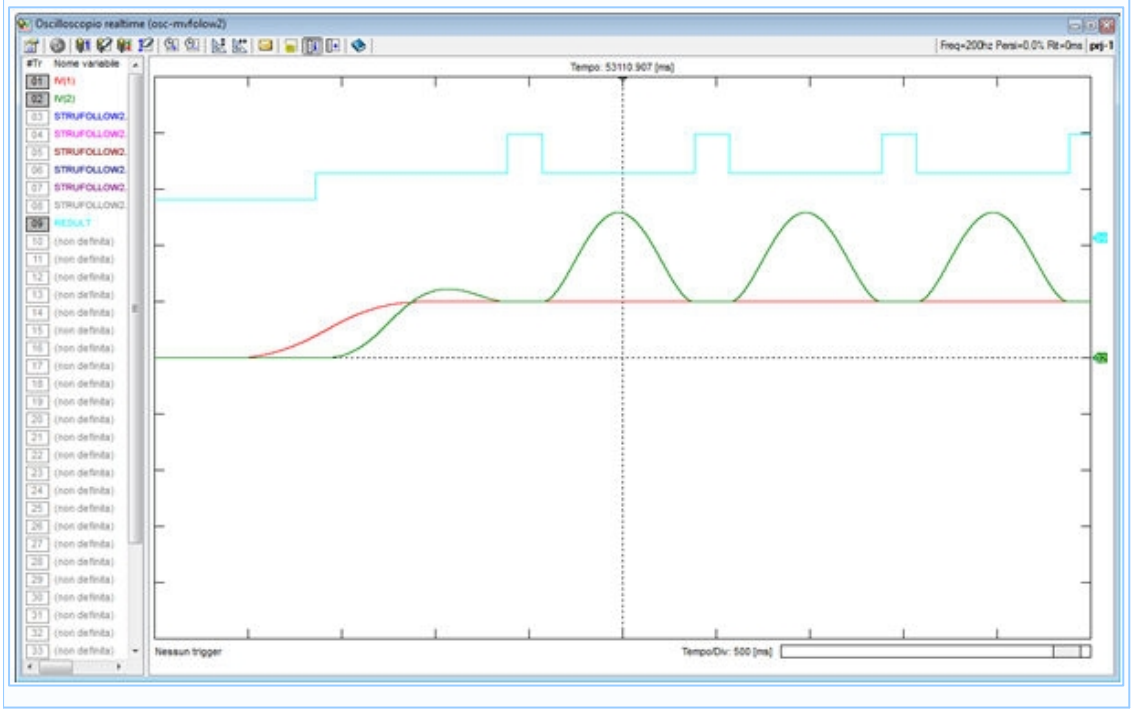


Esempio di utilizzo (master a ciclo fisso e slave rotativo):

```
stru_follow2 struFollow2

rule 1
motion
if(first_time())
    struFollow2.command.0       = 0     ; 1 = forzatura arresto slave
    struFollow2.command.1       = 1     ; 1 = posizionamento iniziale
    struFollow2.command.2       = 0     ; 1 = posizionamento iniziale
    struFollow2.command.16      = 0     ; 0 = master a ciclo fisso, 1
    struFollow2.command.17      = 0     ; 1 = disabilitazione test max
    struFollow2.command.18      = 0     ; 1 = disabilitazione test max
    struFollow2.command.19      = 1     ; 1 = qSyncMaster allineata a
    struFollow2.command.20      = 0     ; 1 = abilitazione fermata a q
```

© 2025 Robox SpA

```
    struFollow2.command.21       = 0      ; 1 = forzatura movimento slav
    struFollow2.command.22       = 0      ; 1 = abilita recupero del sin
    struFollow2.masterCycle      = 400    ; [mm] Master a ciclo fisso di
    struFollow2.qSyncMaster      = 0      ; [mm] Quota di sync master al
    struFollow2.spaceWaitInPark = 0      ; [mm] Spazio master con slave
    struFollow2.slaveCycle       = 360    ; [°] Ciclo slave
    struFollow2.qParkSlave       = 180    ; [°] Quota di parcheggio slav
    struFollow2.qSyncSlave       = -20    ; [°] Quota di sync slave
    struFollow2.qQutSlave        = 20     ; [°] Quota di fine sync slave
    struFollow2.spaceStopSlave   = 0      ; [°] Spazio di arresto slave
    struFollow2.velPositSlave    = 180    ; [°/s] Velocità di posizionam
    struFollow2.accPositSlave    = 360    ; [°/s^2] Accelerazione di pos
    struFollow2.jerkPositSlave   = 720    ; [°/s^3] Jerk di posizionamen
    struFollow2.maxSpe           = 1800   ; [°/s] Velocità massima slave
    struFollow2.maxAcc           = 18000 ; [°/s^2] Accelerazione massim
    struFollow2.kSlaveMaster     = 360 / (k_duepi * slaveRadius) ; [°/m
endif
struFollow2.qMaster    = ip(1)
struFollow2.velMaster = iv(1)
struFollow2.accMaster = ia(1)
struFollow2.qSlave     = ip(2)
struFollow2.velSlave   = iv(2)
struFollow2.accSlave   = ia(2)
result = mv_follow2(struFollow2)
set_kinematics(ax2, struFollow2.qSlave, struFollow2.velSlave, struFoll
end_motion
end_rule
```

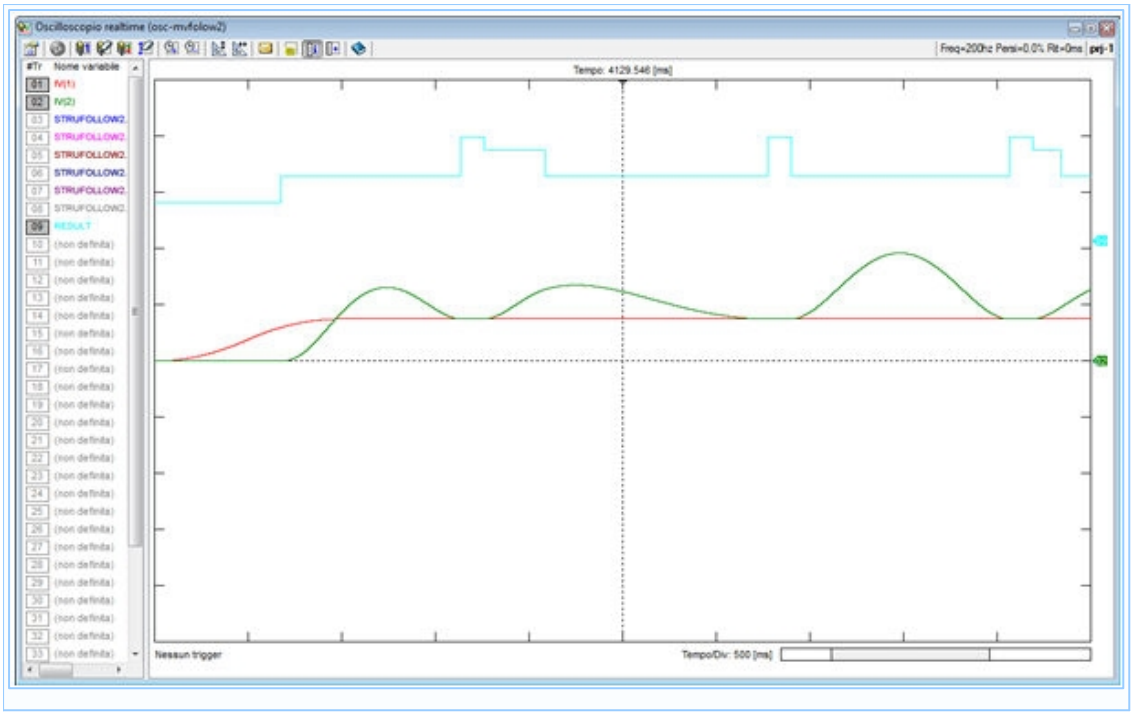


© 2025 Robox SpA

Esempio di utilizzo (master a ciclo variabile e slave rotativo):

```
stru_follow2 struFollow2

rule 1
motion
if(first_time())
    struFollow2.command.0        = 0      ; 1 = forzatura arresto slave
    struFollow2.command.1        = 1      ; 1 = posizionamento iniziale
    struFollow2.command.2        = 0      ; 1 = posizionamento iniziale
    struFollow2.command.16       = 1      ; 0 = master a ciclo fisso, 1
    struFollow2.command.17       = 0      ; 1 = disabilitazione test ma
    struFollow2.command.18       = 0      ; 1 = disabilitazione test ma
    struFollow2.command.19       = 1      ; 1 = qSyncMaster allineata a
    struFollow2.command.20       = 0      ; 1 = abilitazione fermata a
    struFollow2.command.21       = 0      ; 1 = forzatura movimento sla
    struFollow2.command.22       = 1      ; 1 = abilita recupero del si
    struFollow2.masterCycle      = 300    ; [mm] Master a ciclo variabi
    struFollow2.spaceWaitInPark  = 0      ; [mm] Spazio master con slav
    struFollow2.slaveCycle       = 360    ; [°] Ciclo slave
    struFollow2.qParkSlave       = 180    ; [°] Quota di parcheggio sla
    struFollow2.qSyncSlave       = -10    ; [°] Quota di sync slave
    struFollow2.qQutSlave        = 10     ; [°] Quota di fine sync slav
    struFollow2.spaceStopSlave   = 0      ; [°] Spazio di arresto slave
    struFollow2.velPositSlave    = 180    ; [°/s] Velocità di posiziona
    struFollow2.accPositSlave    = 360    ; [°/s^2] Accelerazione di po
    struFollow2.jerkPositSlave   = 720    ; [°/s^3] Jerk di posizioname
    struFollow2.maxSpe           = 1800   ; [°/s] Velocità massima slav
    struFollow2.maxAcc           = 18000  ; [°/s^2] Accelerazione massi
    struFollow2.kSlaveMaster     = 360 / (k_duepi * slaveRadius)  ; [°/
endif
struFollow2.qMaster   = ip(1)
struFollow2.velMaster = iv(1)
struFollow2.accMaster = ia(1)
struFollow2.qSlave    = ip(2)
struFollow2.velSlave  = iv(2)
struFollow2.accSlave  = ia(2)
; Quota di sync master ricavata da un OB di tipo rc_elementlocation
struFollow2.qSyncMaster = ip(1) + buffer.get_distance_to_end(0)
result = mv_follow2(struFollow2)
set_kinematics(ax2, struFollow2.qSlave, struFollow2.velSlave, struFoll
end_motion
end_rule
```

© 2025 Robox SpA



Funzione mva_follow2()

Questa funzione viene utilizzata per realizzare inseguimenti di uno SLAVE nei confronti di un MASTER, con il master non associato ad un asse. La funzione esegue movimenti jerkati calcolati con una curva pentica, camma della posizione dell'asse master.

Premessa

L'inseguimento può essere singolo oppure ripetuto ciclicamente a ciclo fisso o ciclo variabile.

Il master potrebbe essere un asse fisico controllato, ma anche un semplice riferimento esterno (es. un encoder di linea) o anche una quota virtuale costruita e fatta evolvere da parte del software applicativo.

In ogni caso, durante l'esecuzione della funzione, il master deve evolvere sempre nella stessa direzione.

Lo slave potrebbe essere un asse fisico controllato, ma anche una semplice grandezza costruita e fatta evolvere da parte del software applicativo. Esso può essere un asse lineare di escursione limitata oppure un asse ciclico (ad esempio rotativo).

Tipologie della funzione follow2

La sintassi della follow2 varia in funzione di:

- MASTER e SLAVE entrambi non associati ad un asse -> mv_follow2 458()
- MASTER non associato ad un asse, SLAVE associato ad un asse -> mva_follow2()
- MASTER e SLAVE entrambi associati ad un asse (caso tipico) -> mva_follow2_m 469()

Funzionamento

Il funzionamento segue lo stesso concetto spiegato nella funzione mv_follow2 458().

© 2025 Robox SpA

Sintassi	**i32 mva_follow2 (stru_follow2_s stru1 [, real period=0 [, stru_follow2info stru2=NULL])**
stru1	Struttura di tipo stru_follow2_s 855 contenente i parametri relativi alla movimentazione.
period	Intervallo di tempo tra due esecuzioni consecutive di questa funzione. (opzionale, default 0) Se 0 si assume pari a SI.
stru2	Struttura di tipo stru_follow2info 854 dove salvare i dati accessori (out). (opzionale, default NULL)
Risultato	Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63).

Valore	Descrizione
2 [M_POSITIONING]	Posizionamento iniziale richiesto impostando i bit relativi del parametro *command*.
3 [M_WAITING]	L'asse slave è in attesa di partire.
4 [M_WAITING_NEXT_CYCLE]	Non ci sono le condizioni per partire, aspetto il prossimo appuntamento.
5 [M_FOLLOWING]	L'asse slave è partito all'inseguimento del master.
8 [M_SYNCHRONIZED]	L'asse slave si è agganciato ed è sincronizzato al master.
17 [M_DECEL]	L'asse slave ha terminato la fase di sincronismo e sta decelerando.
7 [M_HOMING]	L'asse slave sta ritornando al punto di parcheggio.
-6 [M_IN_ALARM]	La funzione è in allarme, vedi *errorCode*.

Validità	Rule.

© 2025 Robox SpA

Sintassi	**i32 mva_follow2 (stru_follow2_s stru1 [, real period=0 [, stru_follow2info stru2=NULL])**
Note	La configurazione della funzione viene fatta attraverso la struttura dati ausiliaria.

Funzione mva_follow2_m()

Questa funzione viene utilizzata per realizzare inseguimenti di un asse SLAVE nei confronti di un asse MASTER. La funzione esegue movimenti jerkati calcolati con una curva pentica, camma della posizione dell'asse master.

Premessa

L'inseguimento può essere singolo oppure ripetuto ciclicamente a ciclo fisso o ciclo variabile.

Il master potrebbe essere un asse fisico controllato, ma anche un semplice riferimento esterno (es. un encoder di linea) o anche una quota virtuale costruita e fatta evolvere da parte del software applicativo.

In ogni caso, durante l'esecuzione della funzione, il master deve evolvere sempre nella stessa direzione.

Lo slave potrebbe essere un asse fisico controllato, ma anche una semplice grandezza costruita e fatta evolvere da parte del software applicativo. Esso può essere un asse lineare di escursione limitata oppure un asse ciclico (ad esempio rotativo).

Tipologie della funzione follow2

La sintassi della follow2 varia in funzione di:

- MASTER e SLAVE entrambi non associati ad un asse -> mv_follow2 458()
- MASTER non associato ad un asse, SLAVE associato ad un asse -> mva_follow2 467()
- MASTER e SLAVE entrambi associati ad un asse (caso tipico) -> mva_follow2_m()

Funzionamento

Il funzionamento segue lo stesso concetto spiegato nella funzione mv_follow2 458().

Sintassi	**i32 mva_follow2_m (stru_follow2_sm stru1 [, real period=0 [, stru_follow2info stru2=NULL])**
stru1	Struttura di tipo stru_follow2_sm 862 contenente i parametri relativi alla movimentazione.
period	Intervallo di tempo tra due esecuzioni consecutive di questa funzione. (opzionale, default 0) Se 0 si assume pari a SI.
stru2	Struttura di tipo stru_follow2info 854 dove salvare i dati accessori (out). (opzionale, default NULL)
Risultato	Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63).

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mva_follow2_m (stru_follow2_sm stru1 [, real period=0 [, stru_follow2info stru2=NULL])</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td>2 [M_POSITIONING]</td><td>Posizionamento iniziale richiesto impostando i bit relativi del parametro command.</td></tr>
<tr><td>3 [M_WAITING]</td><td>L'asse slave è in attesa di partire.</td></tr>
<tr><td>4 [M_WAITING_NEXT_CYCLE]</td><td>Non ci sono le condizioni per partire, aspetto il prossimo appuntamento.</td></tr>
<tr><td>5 [M_FOLLOWING]</td><td>L'asse slave è partito all'inseguimento del master.</td></tr>
<tr><td>8 [M_SYNCHRONIZED]</td><td>L'asse slave si è agganciato ed è sincronizzato al master.</td></tr>
<tr><td>17 [M_DECEL]</td><td>L'asse slave ha terminato la fase di sincronismo e sta decelerando.</td></tr>
<tr><td>7 [M_HOMING]</td><td>L'asse slave sta ritornando al punto di parcheggio.</td></tr>
<tr><td>-6 [M_IN_ALARM]</td><td>La funzione è in allarme, vedi errorCode.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule.</td></tr>
<tr><td>Note</td><td>La configurazione della funzione viene fatta attraverso la struttura dati ausiliaria.</td></tr>
</table>

Funzione mv_mot_exec()

Funzione che esegue una MOT (Multiple Output Table) creata precedentemente.

Funzionamento

Durante la scansione (esecuzione) si entra con il valore di chiave attuale (*key*) e vengono restituiti i valori interpolati corrispondenti (*arr*) e i dati aggiuntivi del punto che inizia il tratto interpolato (*stru*). I dati sono aggiunti al massimo fino al riempimento e ordinati in modo crescente per *key*.

© 2025 Robox SpA

Sintassi	**i32 mv_mot_exec (i32 motId, real key, real arr [, struct stru=NULL])**
motId	Codice di riferimento della MOT restituito da mot_create 443().
key	Coordinata master (ascissa). È il valore della chiave di interpolazione attuale. Se viene impostato un valore minore o uguale al primo, l'indice di esecuzione viene impostato immediatamente all'inizio della MOT. Analogamente imponendo un valore maggiore o uguale all'ultimo, l'indice di esecuzione viene impostato immediatamente alla fine della MOT.
arr	Array di *n* dati con il risultato della MOT (ordinate).
stru	Struttura dati di tipo struct 917 in cui scaricare i dati ausiliari del tratto. (opzionale, default NULL)
Risultato	Rende:

Valore	Significato
0	Operazione ok.
1	Ok, underflow (*key* < chiave più piccola)
2	Ok, overflow (*key* > chiave più grande)
-1	*motId* non esistente.
-3	*arr* illegale (dimensione dati inferiore a quella attesa).
-4	*stru* illegale (dimensione non coerente con quella specificata in fase di creazione).
-5	Errore interno.
-6	Gestore MOT busy (altra operazione in corso).
-7	Vuoto (nessun punto disponibile).
-12	Tipo di interpolazione non gestito.

Validità	Rule.

© 2025 Robox SpA

Sintassi	**i32 mv_mot_exec (i32 motId, real key, real arr [, struct stru=NULL])**
Note	

Esempi

Esempio di utilizzo:

```
real posSlaveMot, dataFromMot, posMasterMod
; se si ha più di un singolo dato associato con la stessa posizione ma
; allora dataFromMot dovrà essere un array di real

; rule
rule AUTO

    axes(ax_01)

    posMasterMod = abs_mod(ip(MASTER), 360.0)
    esitoMot = mv_mot_exec(rMotID, posMasterMod, dataFromMot)
    posSlaveMot = dataFromMot
    if(esitoMot = 0)
        ip(ax_01) = posSlaveMot
    end_if

end_rule
```

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Funzione mv_phase_adj()

Funzione che permette di correggere la posizione di fase di un asse controllato (slave), il quale deve muoversi in funzione della posizione di un master di riferimento (es. per il controllo di nastri fasatori).

SUGGERIMENTO: Si consiglia l'utilizzo della più performante funzione mv_phase_adj2().

Funzionamento

La correzione avviene sommando (o sottraendo) alla velocità base dello slave un triangolo di velocità, calcolato in modo da recuperare (o perdere) uno spazio pari all'ammontare dell'errore di fase.

Il triangolo di velocità agisce nel periodo impiegato dal master a percorrere uno spazio specificato. La missione viene calcolata con un triangolo eventualmente limitato al valore di velocità massima.

NOTA: Se velocità e/o accelerazione sono limitanti, il valore di errore non viene recuperato tutto (ma solo la parte che riesce a recuperare).

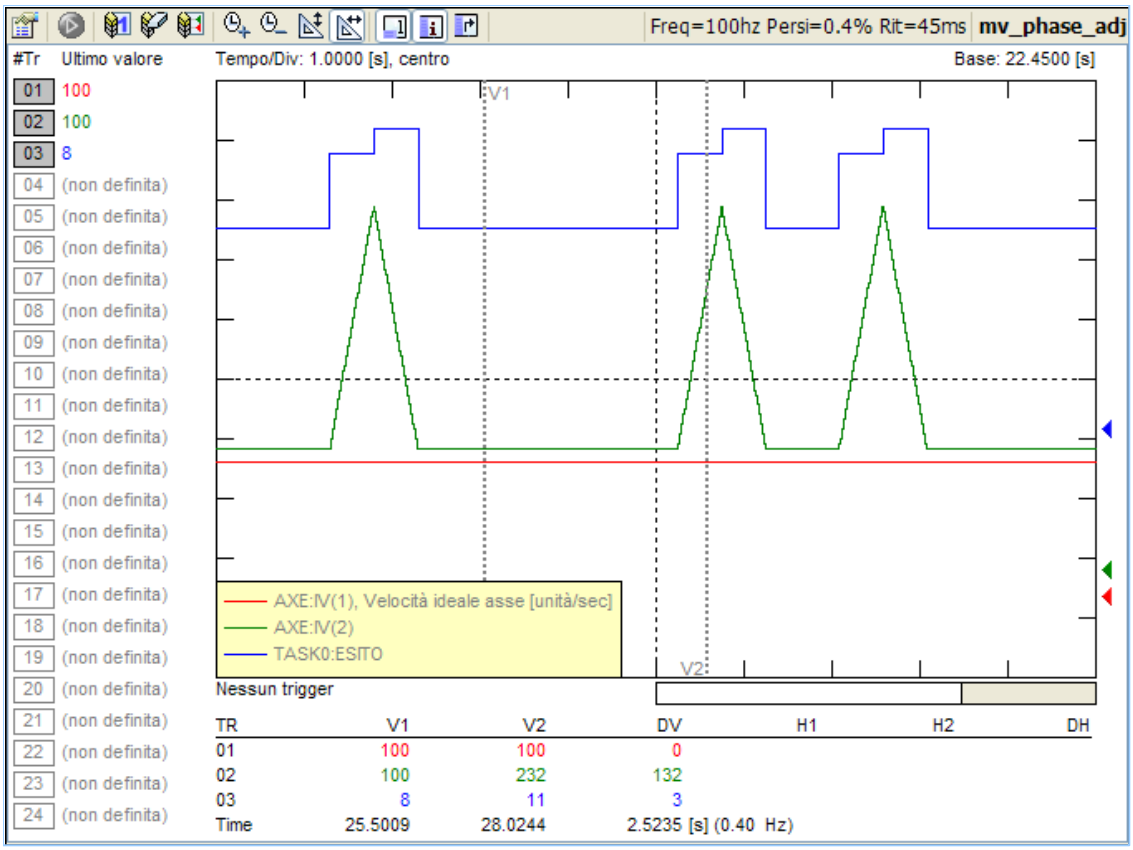
© 2025 Robox SpA

Sintassi	**real mv_phase_adj (i32 state, stru_phadj stru)**
state	Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63). Valore / Descrizione: 11 [M_ADJUSTING_UP] — Correzione in corso (salita triangolo). 12 [M_ADJUSTING_DOWN] — Correzione in corso (discesa triangolo). 8 [M_SYNCHRONIZED] — Movimento concluso (il master ha percorso lo spazio master a disposizione per la correzione dello slave).
stru	Struttura di tipo stru_phadj 898 contenente i parametri relativi alla movimentazione.
Risultato	Valore aggiornato di velocità dell'asse slave reso dalla funzione.
Validità	Rule.
Note	L'asse master può anche evolvere in verso negativo; la funzione provvede a trattare i parametri entranti *masterCycle* e *deltaMaster* come valori assoluti. L'asse slave deve sempre evolvere in senso positivo e la sua velocità *vRef* deve essere positiva.

Esempi

Esempio su oscilloscopio di un possibile output della funzione:

© 2025 Robox SpA



Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Funzione mv_phase_adj2()

Funzione che permette di correggere la posizione di fase di un asse controllato (slave) che deve muoversi in funzione della posizione di un master di riferimento (es. per il controllo di nastri fasatori).

Funzionamento

Per dare inizio alla correzione occorre impostare l'errore e poi assegnare alla variabile di stato il valore M_INITIALIZING (almeno la prima volta) oppure, per correzioni successive, ripartire da 0 con la posizione del master.

La correzione avviene sommando (o sottraendo) alla velocità base dello slave una velocità di correzione calcolata in modo da recuperare (o perdere) uno spazio pari all'ammontare dell'errore.

Mediante il parametro *corrType* è possibile scegliere fra diversi tipi di profilo della velocità di correzione:

- 0, triangolare
- 1, parabolica
- 2, sinusoidale
- 3, jerkata (parabolica con accelerazione continua)

© 2025 Robox SpA

- 4, coseno rialzato
- 5, pentica

In fase di inizializzazione della correzione viene calcolata la velocità di picco della correzione usando la velocità di riferimento del master (*vRefMaster*) e qualora questa superasse una delle velocità limite (*vMin* o *vMax*) il triangolo di correzione (o equivalente) viene automaticamente trasformato in un trapezio, aumentando l'accelerazione iniziale e finale, fino al raggiungimento della *aMax*.

La correzione agisce nel periodo impiegato dal master a percorrere uno spazio specificato.

La velocità e l'accelerazione dello slave vengono eventualmente limitate ai valori di velocità minima e massima ed accelerazione massima segnalando nella variabile di stato che ciò è avvenuto.

Se velocità e/o accelerazione sono limitanti, il valore di errore non viene recuperato tutto.

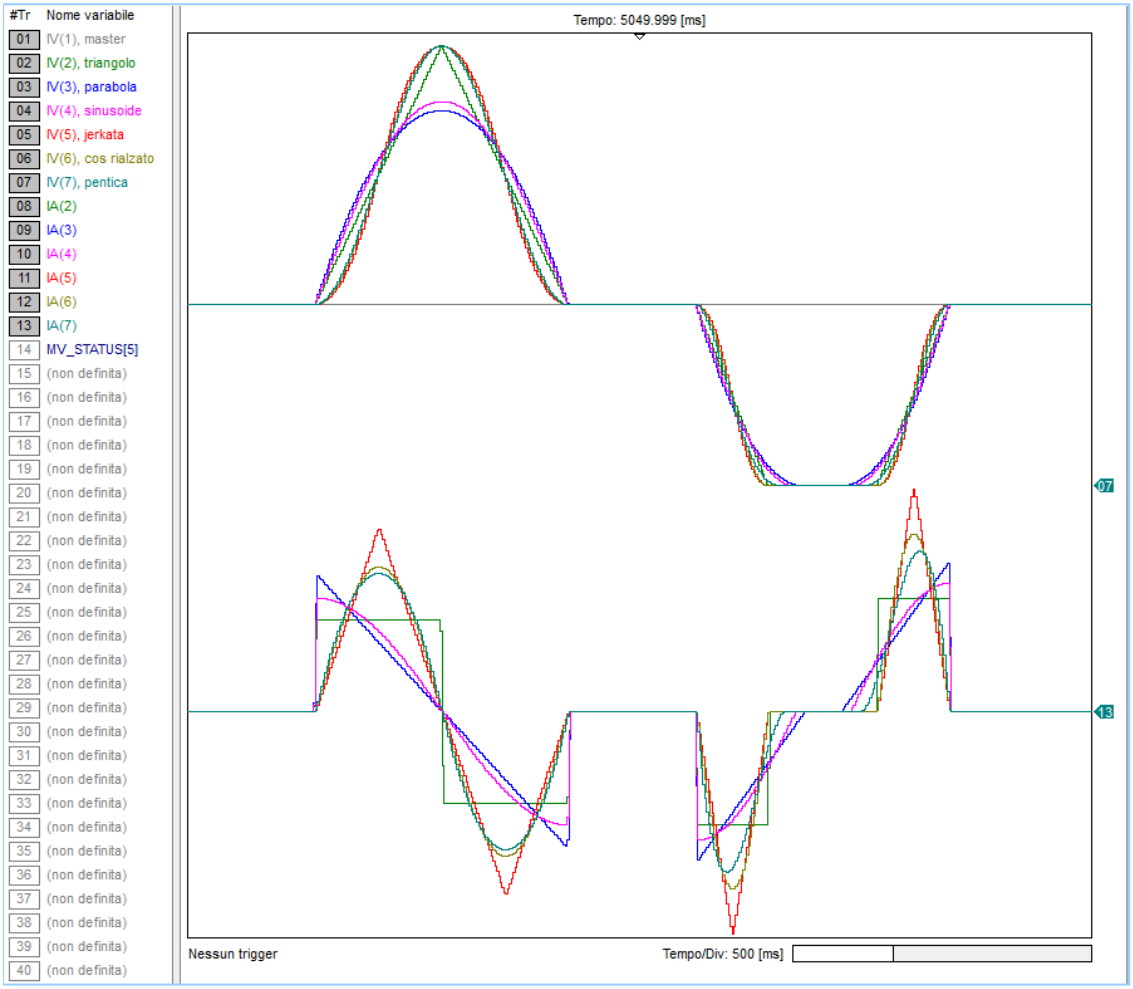
<table>
<tr><td>Sintassi</td><td>real mv_phase_adj2 (i32 state, stru_phadj2 stru)</td></tr>
<tr><td>state</td><td>Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63).
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td>0</td><td>Funzione non attiva, lo slave copia la velocità di riferimento.</td></tr>
<tr><td>1 [M_INITIALIZING]</td><td>Inizio della correzione (impostato dall'utente).</td></tr>
<tr><td>11 [M_ADJUSTING_UP]</td><td>Correzione in corso (salita triangolo).</td></tr>
<tr><td>12 [M_ADJUSTING_DOWN]</td><td>Correzione in corso (discesa triangolo).</td></tr>
<tr><td>8 [M_SYNCHRONIZED]</td><td>Movimento concluso (il master ha percorso lo spazio master a disposizione per la correzione dello slave).</td></tr>
<tr><td>10 [M_SPEED_LIMITED]</td><td>La velocità è stata limitata ai valori minimo o massimo (la correzione sarà parziale).</td></tr>
<tr><td>9 [M_ACCEL_LIMITED]</td><td>La velocità è stata limitata per rispettare il valore di accelerazione massimo (la</td></tr>
</table></td></tr>
</table>

© 2025 Robox SpA

Sintassi	**real mv_phase_adj2 (i32 state, stru_phadj2 stru)**
	<table><tr><th>Valore</th><th>Descrizione</th></tr><tr><td></td><td>correzione sarà parziale).</td></tr></table>
stru	Struttura di tipo stru_phadj2 901 contenente i parametri relativi alla movimentazione.
Risultato	Valore aggiornato di velocità dell'asse slave reso dalla funzione.
Validità	Rule.
Note	L'asse master deve sempre evolvere in senso positivo, non necessariamente a velocità costante.

Esempi

Esempio su oscilloscopio di un possibile output della funzione:



© 2025 Robox SpA

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Funzione mv_ramp()

ATTENZIONE: Deprecata. Usare funzione ramp 555().

Funzione mv_reach_target()

Funzione che calcola la velocità per muovere l'asse slave di uno spazio e in un tempo specificati.

Funzionamento

Calcola la velocità per muovere l'asse slave dello spazio indicato (parametro *ds*) nel tempo indicato (parametro *tm*), passando dalla velocità attuale dell'asse *velSlave* alla velocità di sincronismo *vTarget* (rispettando i parametri *velMinSlave*, *velMaxSlave*, *accMaxSlave*), con accelerazione (calcolata in automatico) costante e minore possibile.

Se *tm*<=0 oppure *ds*<=0 ci si porta il più velocemente possibile verso *vTarget*, usando come accelerazione *accMaxSlave*.

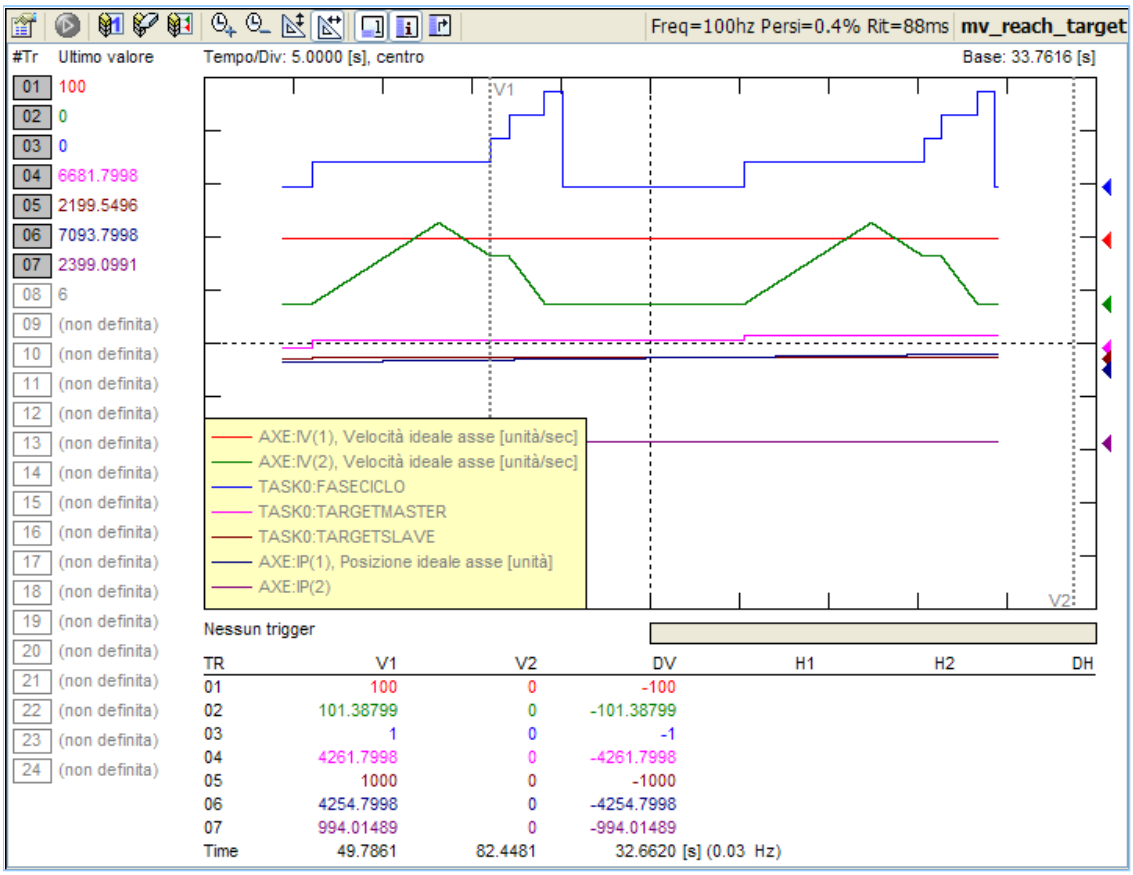
Sintassi	**real mv_reach_target (i32 return, stru_reachtarget stru)**
return	Variabile contenente informazioni relative allo stato della funzione (cedi codici di reso 63).
stru	Struttura di tipo stru_reachtarget 906 contenente i parametri relativi alla movimentazione
Risultato	Valore aggiornato di velocità dell'asse slave reso dalla funzione.
Validità	Rule.
Note	Il movimento di inseguimento dell'asse slave deve avvenire sempre in senso positivo (*ds* > 0).

Valore	Descrizione
-4 [M_LOW_ACCERERATION]	*accMaxSlave* = 0.
5 [M_FOLLOWING]	Movimentazione in corso.
6 [M_REACHED]	Movimentazione conclusa.

Esempi

© 2025 Robox SpA

Esempio su oscilloscopio di un possibile output della funzione:



Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Funzione mv_sine_cam()

Funzione che esegue una camma (y in funzione di x) con velocità di y sinusoidale.

Sintassi	**real mv_sine_cam (real xCurr, real xCycleLen, real yCycleLen)**
xCurr	Valore attuale di x (spazio attuale master). Può assumere valori compresi tra 0 e *xCycleLen*.
xCycleLen	Lunghezza del ciclo in x (spazio master).
yCycleLen	Lunghezza del ciclo in y (spazio slave).
Risultato	Posizione di y calcolata in base a xCurr.
Validità	Rule.
Note	

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

```
; movimento di un asse slave a ruotare sempre nella stessa direzione i
select(fase)
case 0
    ; inizializzazione
    initMaster = ip(ax_master)
    initSlave = ip(ax_slave)
    fase = 1
    ; break omesso volontariamente
case 1
    ; salita
    xCur = ip(ax_master)-initMaster
    if(xCur > masterCycle)
        initMaster = initMaster + masterCycle
        initSlave = initSlave  + masterSlave
        xCur = xcur - masterCycle
    endif
    ip(slave) = initSlave + mv_sine_cam(xCur, masterCycle, masterSla
    break
endselect
```

Funzione mv_synchro()

Funzione che permette di sincronizzare un asse slave ad un asse master in modo tale che lo slave raggiunga la quota di sincronismo, con velocità specificata, ad ogni ciclo del master. Il sincronismo avviene per un solo punto.

Funzionamento

La funzione sposta la grandezza slave specificata dalla posizione corrente *qSlave* verso una destinazione *qSyncSlave*, rispettando l'appuntamento con un master che si sposta a velocità *velMaster* verso il suo appuntamento alla quota *qSyncMaster*.

Il movimento avviene con accelerazioni e velocità ricalcolate in ciascuna chiamata della funzione.

Nel caso la velocità dello slave superi *velMaxSlave* o l'accelerazione superi la *accMaxSlave,* le medesime vengono limitate e viene data segnalazione tramite il codice di ritorno della funzione.

La quota *qSyncSlave* viene raggiunta dallo slave con la velocità *velSyncSlave*.

NOTA: *velSyncSlave* di solito coincide con la *velMaster*, ma non necessariamente.

La *velMaster* può cambiare, per cui i calcoli vengono ripetuti ad ogni battuta e servono solo a decidere l'opportunità, in quella battuta, di accelerare/decelerare lo slave.

La *velMaster* viene presa in valore assoluto e si intende sempre concorde con la velocità di avvicinamento dello slave al luogo di appuntamento *qSyncSlave*.

kappa permette di deformare la traiettoria dello slave: tanto più il valore è alto, tanto più il movimento slave viene anticipato, cioè si cerca di percorrere uno spazio maggiore

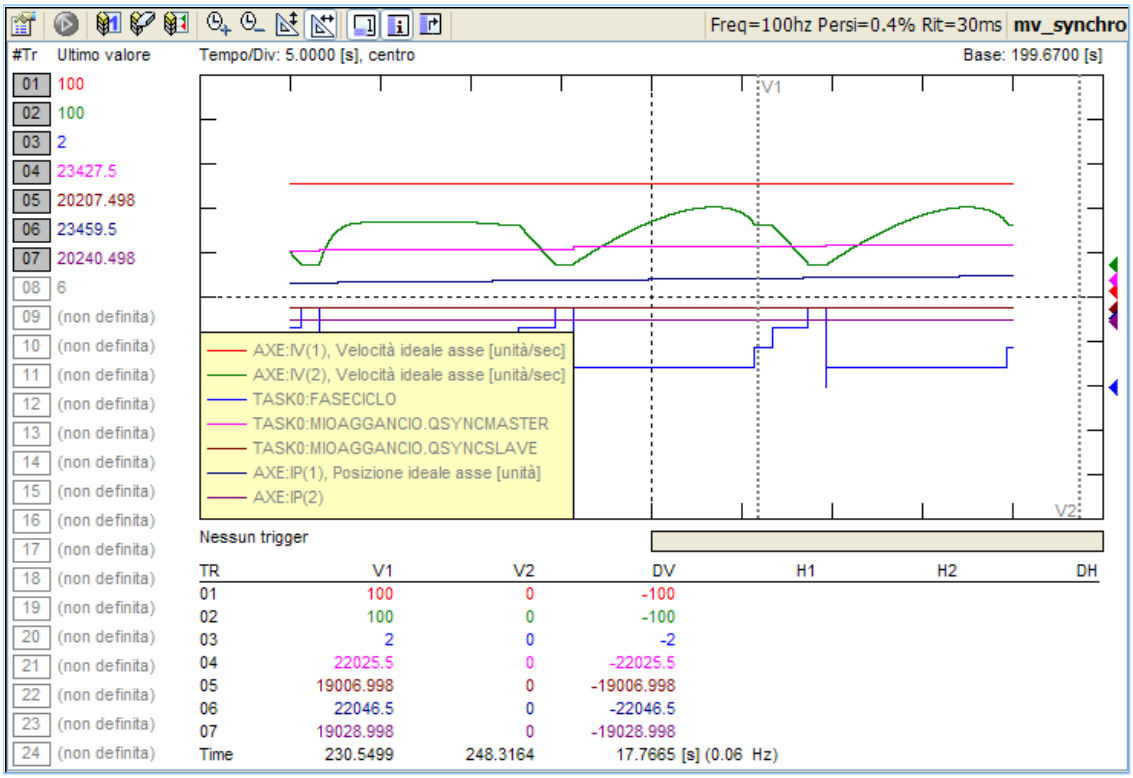
© 2025 Robox SpA

durante la prima metà della missione. Valori ragionevoli vanno da 1.0 (nessun anticipo) fino a 3.0 (massimo anticipo).

Sintassi	**real mv_synchro (i32 return, stru_synchro stru)**
return	Variabile contenente informazioni relative allo stato della funzione (codici di reso 63).
stru	Struttura di tipo stru_synchro 913 contenente i parametri relativi alla movimentazione.
Risultato	Valore aggiornato di posizione dell'asse slave reso dalla funzione.
Validità	Rule.
Note	

Esempi

Esempio su oscilloscopio di un possibile output della funzione:



Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

© 2025 Robox SpA

Funzione mv_table()

Funzione che permette di eseguire una camma per punti.

Funzionamento

In funzione di una grandezza entrante (master), viene calcolato il valore corrispondente della grandezza slave utilizzando una tabella di punti (frutto ad esempio di fogli di calcolo o programmi matematici) che è stata preventivamente caricata sulla CF del controllo (f_read_table_s 333()).

L'interpolazione utilizzata tra i punti è di tipo lineare.

Mediante i parametri opzionali è possibile eseguire il passaggio tra due camme o la somma pesate di essa.

Sintassi	**real mv_table (real xCurr, i32 nPoints1, stru_cam stru1 [, real par=0] [, i32 nPoints2=0] [, stru_cam stru2=NULL])**
xCurr	Coordinata master (ascissa).
nPoints1	Numero di punti di cui è composta la prima camma (coppie di valori x-y). Deve corrispondere alla dimensione dell'array *stru1*.
stru1	Array di strutture di tipo stru_cam 831 contenente i valori della prima camma (lunghezza *nPoints1*).
par	Parametro miscelatore che permette di passare tra le due camme. (opzionale, default 0)

Valore	Significato
0	Esegue la camma 1.
0 -:- 1 (estremi non compresi)	Esegue una sovrapposizione pesata delle due camme.
1	Esegue la camma 2.

nPoints2	Numero di punti di cui è composta la seconda camma. (opzionale, default 0) Deve corrispondere alla dimensione dell'array *stru2*.
stru2	Array di strutture di tipo stru_cam contenente i valori della seconda camma (lunghezza *nPoints2*). (opzionale, default NULL)
Risultato	Valore aggiornato della coordinata y.
Validità	Rule.
Note	

© 2025 Robox SpA

Diagnostica

- Genera 9900 ill. Arg A:5 T:x ST:xxx se *par*<>0 e *nPoints2*=0

Esempi

Esempio di utilizzo:

```
; definizione variabile tab1 di tipo stru_cam
stru_cam 831 tab1[7]
if(first_time 346())
    tab1[0].x = 0
    tab1[1].x = 90
    tab1[2].x = 135
    tab1[3].x = 180
    tab1[4].x = 270
    tab1[5].x = 330
    tab1[6].x = 360
    tab1[0].y = 10
    tab1[1].y = 90
    tab1[2].y = 250
    tab1[3].y = 300
    tab1[4].y = 270
    tab1[5].y = 100
    tab1[6].y = 10
    rr(100) = 1
endif
ip(1) = mod 439 (rr(100) * tfb * 360.0, 360)
ip(2) = mv_table 481 (ip(1), 7, tab1)
```

Funzione mv_to()

Funzione che porta la grandezza indicata al valore prefissato con un profilo velocità trapezoidale.

NOTA: È analoga alla funzione mv_to 484() omonima con parametrizzazione in linea.

Sintassi	**real mv_to (i32 result, stru_mvto stru [, real sovraDec=0])**
return	Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63).

Valore	Descrizione
14 [M_HOLDED]	Velocità desiderata o feedrate impostati a 0.
6 [M_REACHED]	Target raggiunto.
15 [M_ACCEL]	Accelerazione in corso.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>real mv_to (i32 result, stru_mvto stru [, real sovraDec=0])</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td>13 [M_OVERSHOOTING]</td><td>Target sorpassato.</td></tr>
<tr><td>17 [M_DECEL]</td><td>Decelerazione in corso.</td></tr>
<tr><td>16 [M_AT_SPEED]</td><td>Movimentazione a regime.</td></tr>
</table>
</td></tr>
<tr><td>stru</td><td>Struttura di tipo stru_mvto 881 contenente i parametri relativi alla movimentazione.</td></tr>
<tr><td>sovraDec</td><td>Percentuale di decelerazione da applicare nel caso venga avvicinato runtime il target della missione quando essa è già in fase di decelerazione. (opzionale, default 0)

Se negativo viene assunto 0.

L'accelerazione risultante sarà: dec + (sovraDec / 100) * dec.

(es. se dec = 120 e sovraDec = 25 la decelerazione risulterà 150).</td></tr>
<tr><td>Risultato</td><td>Valore aggiornato reso dalla funzione.</td></tr>
<tr><td>Validità</td><td>Rule.</td></tr>
<tr><td>Note</td><td>sovraDec può essere utile nel caso di calcolo del target di arresto, su variazione dell'input digitale (es. fotocellula).</td></tr>
</table>

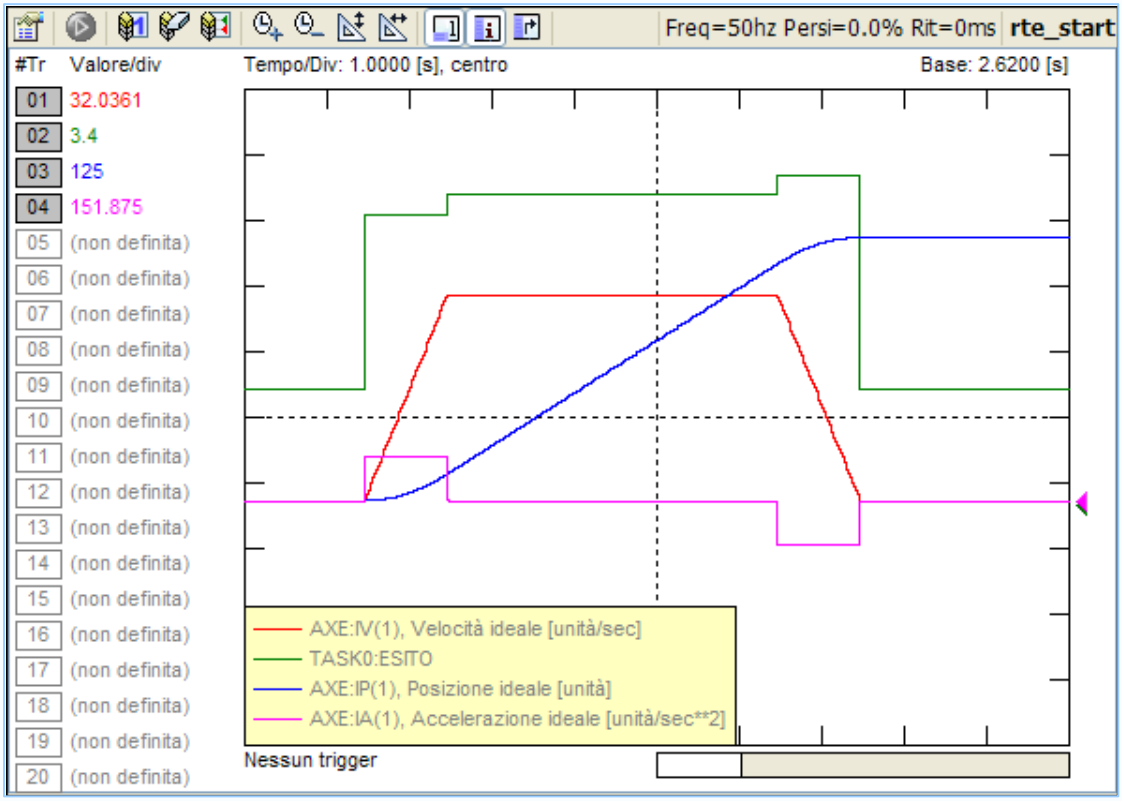
Esempi

Esempio di utilizzo:

```
; definizioni locali
stru_mvto 881 mioMov    ; variabile di tipo stru_mvto
int esito

; rule
; posizione attuale [units]
mioMov.cPos = ip(AX_1)
; velocità attuale [units/sec]
mioMov.cSpeed = iv(AX_1)
; posizione desiderata [units]
mioMov.dPos = rr(2000)
; velocià desiderata [units/sec]
mioMov.dSpeed = rr(2001)
; accelerazione [units/sec²]
mioMov.acc = rr(2002)
```

© 2025 Robox SpA

```
; decelerazione [units/sec²]
mioMov.dec = rr(2003)
; valore di SI
mioMov.delta = SI
ip(AX_1) = mv_to(esito, mioMov, sovradec)
; (vel 100, acc 100, dec 100, spazio 500)
```



Funzione mv_to(), parametrizzata in linea

Funzione che porta l'asse specificato al valore di posizione prefissato con un profilo velocità-tempo trapezoidale.

NOTA: È analoga alla funzione mv_to() omonima con parametrizzazione tramite struttura.

Sintassi	**real mv_to (i32 result, i32 nAx, real pDes, real sDes, real aDes [, real dDes=-aDes] [, real sovraDec=0])**
result	Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso).

Valore	Descrizione
14 [M_HOLDED] (se *sovraDec* esiste = 0)	Velocità desiderata o feedrate impostati a

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>real mv_to (i32 result, i32 nAx, real pDes, real sDes, real aDes [, real dDes=-aDes] [, real sovraDec=0])</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td></td><td>0.</td></tr>
<tr><td>6 [M_REACHED]</td><td>Target raggiunto.</td></tr>
<tr><td>15 [M_ACCEL]</td><td>Accelerazione in corso.</td></tr>
<tr><td>13 [M_OVERSHOOTING]</td><td>Target sorpassato.</td></tr>
<tr><td>17 [M_DECEL]</td><td>Decelerazione in corso.</td></tr>
<tr><td>16 [M_AT_SPEED]</td><td>Movimentazione a regime.</td></tr>
<tr><td>-5 [M_ILLEGAL_ARGS]</td><td>Asse errato.</td></tr>
</table>
</td></tr>
<tr><td>nAx</td><td>Numero dell'asse da muovere (1-32).</td></tr>
<tr><td>pDes</td><td>Posizione target.</td></tr>
<tr><td>sDes</td><td>Velocità di regime.</td></tr>
<tr><td>aDes</td><td>Accelerazione.</td></tr>
<tr><td>dDes</td><td>Decelerazione. (opzionale, default -aDes)</td></tr>
<tr><td>sovraDec</td><td>Percentuale di decelerazione da applicare nel caso venga avvicinato runtime il target della missione quando essa è già in fase di decelerazione. (opzionale, default 0)

Se negativo viene assunto 0.

L'accelerazione risultante sarà: dDes + (sovraDec / 100) * dDes.

(es. se dDes = 120 e sovraDec = 25 la decelerazione risulterà 150).</td></tr>
<tr><td>Risultato</td><td>Valore aggiornato reso dalla funzione.</td></tr>
<tr><td>Validità</td><td>Rule.</td></tr>
<tr><td>Note</td><td>sovraDec può essere utile nel caso di calcolo del target di arresto, su variazione di un input digitale (es. fotocellula).

Se allarme CAM abilitato la missione viene limitata ai paracarri. Non viene onorato il FR.</td></tr>
</table>

Esempi

Esempio di utilizzo:

© 2025 Robox SpA

```
ip(1) = mv_to(esito, 1, rr(2000), rr(2001), rr(2002))
; in rr(2000) è contenuto il valore di posizione target
; in rr(2001) è contenuto il valore di velocità
; in rr(2002) è contenuto il valore di accelerazione
```

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Funzione mv_to_cj()

Funzione che esegue la movimentazione verso un target (con partenza da fermo) di una grandezza generica specificando velocità desiderata, accelerazione e jerk.

I parametri di velocità desiderata, accelerazione e jerk vengono inseriti nella struttura relativa.

I dati sono assunti solo su *start*. In caso di partenza con velocità diversa da 0 si ha una fase di arresto della grandezza prima di partire verso il target programmato.

Hold e variazioni di fr sono gestiti utilizzando una rampa lineare di velocità.

NOTA: È analoga alla funzione mv_to_cjv() ma quest'ultima consente di partire da velocità iniziale diversa da 0 e di cambiare i parametri runtime.

Sintassi	**real mv_to_cj (i32 return, i32 start, stru_mvtocj stru [, i32 holdFlag=0] [, real fr=1])**
return	Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso).
start	Variabile usata per far acquisire i parametri di missione (richiesta con qualsiasi valore <> 0). La funzione resetta la richiesta scrivendoci 0.
stru	Variabile di tipo stru_mvtocj contenente i parametri della funzione.
holdFlag	Flag per comandare lo stato di hold alla missione. (opzionale, default 0)
fr	Valore di riduzione delle velocità (0-1). (opzionale, default 1) Vedi fr.
Risultato	Valore di posizione della grandezza.
Validità	Rule.
Note	

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

© 2025 Robox SpA

Funzione mv_to_cjv()

Funzione che porta la grandezza controllata dalla velocità attuale (corrispondente alla prima chiamata della funzione) a quella desiderata di regime, controllando l'accelerazione con il jerk specificato.

È possibile in qualsiasi momento la variazione delle grandezze desiderate (posizione target, velocità di regime, accelerazione e jerk).

La funzione termina il suo lavoro quando ci si arresta sulla posizione target specificata.

Funzionamento

I dati correnti (*cPos*, *cSpeed*, *cAcc*) tipicamente sono impostati solo quando si effettua la prima chiamata alla funzione, dopodiché vengono aggiornati dalla funzione.

I valori restituiti di *cPos*, *cSpeed* e *cAcc*, nel caso si utilizzi questa funzione per pilotare un asse, devono essere impostati tramite l'istruzione set_kinematics 595().

I dati desiderati (*dPos*, *dSpeed*, *dAcc*, *dJerk*) possono essere cambiati in un qualsiasi momento.

NOTA: È analoga alla funzione mv_to_cj 486() ma quest'ultima consente di partire solo con velocità iniziale pari a 0.

Sintassi	**i32 mv_to_cjv (stru_mvtocjv|stru_mvtocjve stru [, i32 holdFlag=0] [, real fr=1])**
stru	Variabile di tipo stru_mvtocjv 886 o stru_mvtocjve 890 contenente i parametri della funzione.
holdFlag	Flag per comandare lo stato di hold alla missione. (opzionale, default 0)
fr	Valore di riduzione delle velocità (0-1). (opzionale, default 1) Vedi fr.
Risultato	Rende lo stato della funzione (vedi codici di reso 63): Valore | Descrizione -6 [M_IN_ALARM] | Allarme presente. 2 [M_POSITIONING] | Posizionamento iniziale. 13 [M_OVERSHOOTING] | Posizione target superata. 15 [M_ACCEL] | Accelerazione in corso. 16 [M_AT_SPEED] | Movimentazione a regime. 17 [M_DECEL] | Decelerazione in corso.

© 2025 Robox SpA

Sintassi	**i32 mv_to_cjv (stru_mvtocjv| stru_mvtocjve stru [, i32 holdFlag=0] [, real fr=1])**
	Valore: 6 [M_REACHED] — Descrizione: Target raggiunto.
Validità	Rule.
Note	Nella stru_mvtocjve, in caso di *dSpeed*=0, si ha scrittura automatica della posizione di arresto e del relativo tempo necessario. Può generare allarme 9900 (vedi flag di configurazione sys_flag.4). È possibile aumentare la diagnostica nel report tramite sys_flag.10.

Esempi

Esempio di utilizzo:

```
if(first_time346())
    ; code executed just the fist time
    ; set parameter of the structure
    ; sample interval
    mv.delta = SI
    ; actual posiz [units]
    mv.cPos = IP(AX_1)
    ; actual speed [units/sec]
    mv.cSpeed = IV(AX_1)
    ; actual acc [units/sec*sec]
    mv.cAcc = IA(AX_1)
endif

if(rRichiestaStop)
    rRichiestaStop = FALSE
    HOLD = TRUE
endif

; desired posiz [units]
mv.dpos = rrDesPos
; desired speed [units/sec]
mv.dSpeed = rrDesSpeed
; desired acc [units/sec*sec]
mv.dacc = rrDesAcc
; desired jerk [units/sec*sec*sec]
mv.djerk = rrDesJerk
; call the function
esito = mv_to_cjv487(mv, HOLD, FR)
; link to the axe
set_kinematics595(AX_1, mv.cPos, mv.cSpeed, mv.cAcc)
```

© 2025 Robox SpA

Funzione mv_to_cjv_info()

Funzione che calcola lo spazio e il tempo necessario per raggiungere la velocità desiderata con accelerazione e jerk specificati.

Si devono fornire i dati di cinematica attuali (*cPos*, *cSpeed*, *cAcc*) e desiderati (*dAcc*, *dJerk*, *targetSpeed*). Vengono quindi restituiti i parametri *posInfo e timeInfo*, ovvero la posizione in cui sarà raggiunta la velocità target desiderata e il tempo necessario a raggiungerla.

Sintassi	**i32 mv_to_cjv_info (stru_mvtocjv_info stru)**
stru	Variabile di tipo stru_mvtocjv_info 888 contenente i parametri della funzione.
Risultato	Rende un valore TRUE se tutti i parametri sono corretti, altrimenti se l'accelerazione o il jerk sono uguali a 0 rende FALSE e viene generato l'allarme 9900 illegal arg.
Validità	Rule, Task.
Note	Tipicamente questa funzione viene usata in congiunzione con mv_to_cjv 487(). In tal caso è indispensabile fornire gli stessi dati di ingresso come valori correnti.

Funzione mv_to_vel()

Funzione che esegue la movimentazione di una grandezza generica, che si muove inizialmente ad una certa velocità, verso un punto target da raggiungere con velocità desiderata di crociera e velocità finale.

Sintassi	**real mv_to_vel (i32 return, real pAct, real vAct, real pDes, real vDes, real vFin, real baseTime, real acc [, real dec=acc] [, real perc=0.8])**
return	Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63).

Valore	Descrizione
-5 [M_ILLEGAL_ARGS]	Parametro illegale (*acc*=0 o *baseTime*=0).
13 [M_OVERSHOOTING]	Target sorpassato. La move procede inerzialmente.
-4 [M_LOW_ACCELERATION]	Troppo veloce per raggiungere il punto target con la *vFin* specificata.
6 [M_REACHED]	Target raggiunto (vale solo nel caso in cui si

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>real mv_to_vel (i32 return, real pAct, real vAct, real pDes, real vDes, real vFin, real baseTime, real acc [, real dec=acc] [, real perc=0.8])</td></tr>
<tr><td></td><td><table>
<tr><td>Valore</td><td>Descrizione</td></tr>
<tr><td></td><td>sia specificato vFin=0).</td></tr>
<tr><td>15 [M_ACCEL]</td><td>Accelerazione in corso.</td></tr>
<tr><td>16 [M_AT_SPEED]</td><td>Movimentazione a regime con vDes.</td></tr>
<tr><td>17 [M_DECEL]</td><td>Decelerazione in corso.</td></tr>
<tr><td>18 [M_LAST_CYCLE]</td><td>Ultimo ciclo prima di oltrepassare il target.</td></tr>
</table></td></tr>
<tr><td>pAct</td><td>Posizione attuale.</td></tr>
<tr><td>vAct</td><td>Velocità attuale.</td></tr>
<tr><td>pDes</td><td>Posizione da raggiungere.</td></tr>
<tr><td>vDes</td><td>Velocità desiderata (intesa in valore assoluto).</td></tr>
<tr><td>vFin</td><td>Velocità finale (intesa in valore assoluto).</td></tr>
<tr><td>baseTime</td><td>Base tempi (normalmente SI).</td></tr>
<tr><td>acc</td><td>Accelerazione desiderata (intesa in valore assoluto).</td></tr>
<tr><td>dec</td><td>Decelerazione desiderata (intesa in valore assoluto). (opzionale, default acc)</td></tr>
<tr><td>perc</td><td>Valore percentuale che influisce sulla decelerazione (0-1). (opzionale, default 0.8)

Durante i calcoli di arrivo sul punto target, viene valutato il numero di battute necessarie per arrivare dalla posizione attuale a quella desiderata con la velocità vFin specificata. Tale numero viene troncato all'intero inferiore se lo sfrido di battuta è < perc (quindi la decelerazione effettiva sarà un po' più grande di dec) oppure verrà aumentato all'intero seguente se lo sfrido di battuta è > perc (decelerazione risultante minore).</td></tr>
<tr><td>Risultato</td><td>Valore della posizione aggiornata.</td></tr>
<tr><td>Validità</td><td>Rule.</td></tr>
<tr><td>Note</td><td>vFin si intende concorde con la direzione del movimento richiesto (se pDes > pAct, la vFin si</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**real mv_to_vel (i32 return, real pAct, real vAct, real pDes, real vDes, real vFin, real baseTime, real acc [, real dec=acc] [, real perc=0.8])**
	intende positiva, diversamente si intende negativa). L'aggiornamento della velocità *vAct* va fatto esternamente alla funzione a cura dell'utente.

Se *vFin* > 0, la posizione target viene oltrepassata, dopodiché la funzione prosegue inerzialmente fintanto che viene invocata. Se *vFin* = 0, la funzione si ferma sul punto desiderato.

Se il parametro di decelerazione non consente di arrivare da *pAct* a *pDes* con la *vFin* specificata, la mv_to_vel() tende comunque a raggiungere *vFin*; *pDes* verrà oltrepassata con una velocità diversa da *vFin* e il movimento proseguirà inerzialmente con velocità tendente progressivamente a *vFin*.

NOTA: la funzione può essere chiamata anche con l'asse già in movimento.

Esempi

Esempio di utilizzo:

```
select(fase)
; ...
case 4 ; vado su P4 con velocità finale V2 e velocità di crociera V2
    ip(asse_4) = mv_to_vel(esito, ip(asse_4), iv(asse_4), P4, V2, V2
    if(esito = M_OVERSHOOTING)
        fase = 5
    endif
    break
case 5 ; mi fermo su P5 con targetDec
    ip(asse_4) = mv_to_vel(esito, ip(asse_4), iv(asse_4), P5, V2, 0, S
    if(esito = M_REACHED)
        tm_on(t1, TRUE)
        fase = 6
    endif
    break
; ...
endselect
```

Funzione mv_tracking()

ATTENZIONE: Deprecata. Usare funzione tracking().

Funzione mva_jog()

Funzione che esegue la movimentazione degli assi in modalità manuale.

Rispetto a mva_jog2():

- Non è possibile impostare distintamente accelerazione e decelerazione

© 2025 Robox SpA

- Non è possibile impostare distintamente jerk in accelerazione e in decelerazione
- Non è possibile gestire l'arresto su paracarro con movimento jerkato

Sintassi	**i32 mva_jog (i32 nAx, i32 dir, real spe, real acc [, real jer=1e100] [, i32 noEEOS=0])**
nAx	Numero dell'asse (1-:-32).
dir	Direzione: Valore: -1 — Direzione negativa. 0 — Asse fermo. 1 — Direzione positiva.
spe	Velocità di regime.
acc	Accelerazione.
jer	Jerk. (opzionale, default 1e100)
noEEOS	Flag per disabilitare l'utilizzo di EEOS (Electronic End Of Stroke) come fine corsa: (opzionale, default 0) Valore: 0 — Abilitato EEOS. 1 — Disabilitato. Se disabilitato non sono gestiti gli allarmi CRASH, anche se CAM abilitato. **NOTA**: Vedi variabili predefinite min_str e max_str.
Risultato	Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63). Valore: -5 [M_ILLEGAL_ARGS] — Parametro illegale (*acc*=0 o *nAx* illegale). -6 [M_IN_ALARM] — Allarme (ip asse già impostata). 6 [M_REACHED] — Target raggiunto. Ciò significa che o la direzione è 0 e l'asse è fermo, oppure la direzione è +1 o -1 e l'asse ha raggiunto uno dei paracarri.

© 2025 Robox SpA

Sintassi	**i32 mva_jog (i32 nAx, i32 dir, real spe, real acc [, real jer=1e100] [, i32 noEEOS=0])**
	<table><tr><th>Valore</th><th>Descrizione</th></tr><tr><td>15 [M_ACCEL]</td><td>Accelerazione in corso.</td></tr><tr><td>16 [M_AT_SPEED]</td><td>Movimentazione a regime.</td></tr><tr><td>17 [M_DECEL]</td><td>Decelerazione in corso.</td></tr></table>
Validità	Rule.
Note	Agendo sul jerk si ottiene un profilo di velocità a 'esse', quindi un movimento più dolce. La fermata su paracarro viene effettuata con jerk massimo.

Diagnostica

- Genera allarme 9900 ill. Arg. A:2 T:x ST:xxx se *nAx* illecito
- Genera allarme 9900 ill. Arg. A:4 T:x ST:xxx se *acc* = 0
- Genera allarme 2800 Double Position Set T:x ST:xxx se già imposta la ip dell'asse

Esempi

Esempio di utilizzo:

```
rule 100
axes 1
motion
    esito = mva_jog(1, jog, speed, accel, accel*10, 1)
end_motion
end_rule
```

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Funzione mva_jog2()

Funzione che esegue la movimentazione degli assi in modalità manuale.

Rispetto a mva_jog() permette di:

- Impostare distintamente accelerazione e decelerazione
- Impostare distintamente jerk in accelerazione e in decelerazione
- Gestire l'arresto su paracarro con movimento jerkato

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mva_jog2 (i32 nAx, i32 dir, real spe, real acc, real dec [, real jerAcc=0] [, real jerDec=0] [, i32 noEEOS=0])</td></tr>
<tr><td>nAx</td><td>Numero dell'asse (1-:-32).</td></tr>
<tr><td>dir</td><td>Direzione:<table><tr><th>Valore</th><th>Significato</th></tr><tr><td>-1</td><td>Direzione negativa.</td></tr><tr><td>0</td><td>Asse fermo.</td></tr><tr><td>1</td><td>Direzione positiva.</td></tr></table></td></tr>
<tr><td>spe</td><td>Velocità di regime.</td></tr>
<tr><td>acc</td><td>Accelerazione.</td></tr>
<tr><td>dec</td><td>Decelerazione.</td></tr>
<tr><td>jerAcc</td><td>Jerk in accelerazione (0 nessun jerk, 1 tutta la fase di accelerazione con jerk controllato). (opzionale, default 0)</td></tr>
<tr><td>jerDec</td><td>Jerk in decelerazione (0 nessun jerk, 1 tutta la fase di decelerazione con jerk controllato).
(opzionale, default 0)</td></tr>
<tr><td>noEEOS</td><td>Flag per disabilitare l'utilizzo di EEOS (Electronic End Of Stroke) come fine corsa: (opzionale, default 0)<table><tr><th>Valore</th><th>Significato</th></tr><tr><td>0</td><td>Abilitato EEOS.</td></tr><tr><td>1</td><td>Disabilitato.</td></tr></table>Se disabilitato non sono gestiti gli allarmi CRASH, anche se CAM abilitato.
NOTA: Vedi variabili predefinite min_str e max_str.</td></tr>
<tr><td>Risultato</td><td>Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63).<table><tr><th>Valore</th><th>Descrizione</th></tr><tr><td>-5 [M_ILLEGAL_ARGS]</td><td>Parametro illegale (acc=0 o nAx illegale).</td></tr><tr><td>-6 [M_IN_ALARM]</td><td>Allarme (ip asse già impostata).</td></tr><tr><td>6 [M_REACHED]</td><td>Target raggiunto.</td></tr></table></td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mva_jog2 (i32 nAx, i32 dir, real spe, real acc, real dec [, real jerAcc=0] [, real jerDec=0] [, i32 noEEOS=0])</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td></td><td>Ciò significa che o la direzione è 0 e l'asse è fermo, oppure la direzione è +1 o -1 e l'asse ha raggiunto uno dei paracarri.</td></tr>
<tr><td>15 [M_ACCEL]</td><td>Accelerazione in corso.</td></tr>
<tr><td>16 [M_AT_SPEED]</td><td>Movimentazione a regime.</td></tr>
<tr><td>17 [M_DECEL]</td><td>Decelerazione in corso.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule.</td></tr>
<tr><td>Note</td><td>Agendo sul jerk si ottiene un profilo di velocità a 'esse', quindi un movimento più dolce.</td></tr>
</table>

Esempi

Esempio di utilizzo:

```
; movimento con accelerazione e decelerazione eguali e con il massimo
rule 100
axes 1
motion
    esito = mva_jog2(1, jog, 10, 100, 100, 1, 1, 0)
end_motion
end_rule

; movimento con accelerazione più dolce della decelerazione e con l' e
rule 100
axes 1
motion
    esito = mva_jog2(1, jog, 10, 100, 200, 1, 0.5, 1)
end_motion
end_rule

rule 100
axes 1
motion
    esito = mva_jog2(1, jog, 10, 100,100) ; come mva_jog
end_motion
end_rule
```

© 2025 Robox SpA

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Funzione mva_open_loop()

Funzione che deve essere usata nelle rule di missing_power. Essa imposta:

- IP = CP (la quota letta dal trasduttore viene copiata nella Ideal Position)
- Non genera allarme CRASH anche se abilitato
- Imposta IV e IA a 0

Dalla versione RTE 34.27.02, appena l'asse viene abilitato viene impostata la velocità ideale a 0.

NOTA: Nel caso si volesse utilizzare il vecchio comportamento, impostare a 1 sys_cfg_2.12.

Sintassi	**i32 mva_open_loop (i32 nAx)**
nAx	Numero dell'asse (1-:-32).
Risultato	0 se Ok, altrimenti errore
Validità	Rule.
Note	

NOTA: Nel caso sia attiva la Planar Table Compensation tramite sys_cfg.12 si impone IP(ax) = CP_C(ax).

Esempi

Esempio di utilizzo:

```
rule 100
axes 1
motion
    esito = mva_open_loop(1)
end_motion
end_rule
```

Funzione mva_to_n()

Funzione che porta gli assi (1-:-32) indicati verso il valore di posizione target prefissato con un profilo velocità verso tempo trapezoidale.

NOTA: La differenza con la funzione mva_to_n_v() consiste nel fatto che non può essere chiamata ad asse/i già in movimento e non può essere specificata la velocità finale sul target.

Sintassi	**i32 mva_to_n (i32 start, stru_mvitp stru [, i32 nDef=1] [, i32 holdFlag=0] [, real fr=1])**
start	Flag per abilitare la riassunzione dei dati (solo se assi fermi).

© 2025 Robox SpA

Sintassi	**i32 mva_to_n (i32 start, stru_mvitp stru [, i32 nDef=1] [, i32 holdFlag=0] [, real fr=1])**
	Il flag viene azzerato quando i dati vengono assunti.
stru	Struttura di tipo stru_mvitp 879 contenente i parametri relativi alla movimentazione.
nDef	Dimensione dell'array (numero di assi coinvolti). (opzionale, default 1)
holdFlag	Flag per comandare lo stato di HOLD alla missione. (opzionale, default 0)
fr	Valore di riduzione delle velocità (0-1). (opzionale, default 1) Vedi fr.
Risultato	Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63).

Valore	Descrizione
-5 [M_ILLEGAL_ARGS]	Parametro illegale (*acc*=0 o numero asse illegale).
-6 [M_IN_ALARM]	Allarme (ip asse già impostata).
2 [M_POSITIONING]	Posizionamento iniziale.
6 [M_REACHED]	Target raggiunto. Ciò significa che o la direzione è 0 e l'asse è fermo, oppure la direzione è +1 o -1 e l'asse ha raggiunto uno dei paracarri.
14 [M_HOLDED]	Movimento arrestato causa flag HOLD. Viene restituito anche nei casi in cui la velocità desiderata dell'asse viene impostata a 0 oppure il feedrate viene impostato a zero.
15 [M_ACCEL]	Accelerazione in corso.
16 [M_AT_SPEED]	Movimentazione a regime.

© 2025 Robox SpA

Sintassi	**i32 mva_to_n (i32 start, stru_mvitp stru [, i32 nDef=1] [, i32 holdFlag=0] [, real fr=1])**
	Valore: 17 [M_DECEL] — Descrizione: Decelerazione in corso.
Validità	Rule.
Note	La missione può coinvolgere più assi (max 32). Essi verranno interpolati linearmente al fine di raggiungere tutti insieme i loro target.

Diagnostica

- Genera allarme 9900 ill. Arg. A:2 T:x ST:xxx se numero di asse illecito
- Genera allarme 9900 ill. Arg. A:3 T:x ST:xxx se *acc* = 0
- Genera allarme 9903 Par. are changed T:x ST:xxx se cambiata sequenza assi in corsa
- Genera allarme 2800 Double Position Set T:x ST:xxx se già imposta la ip dell'asse

Esempi

Esempio di utilizzo:

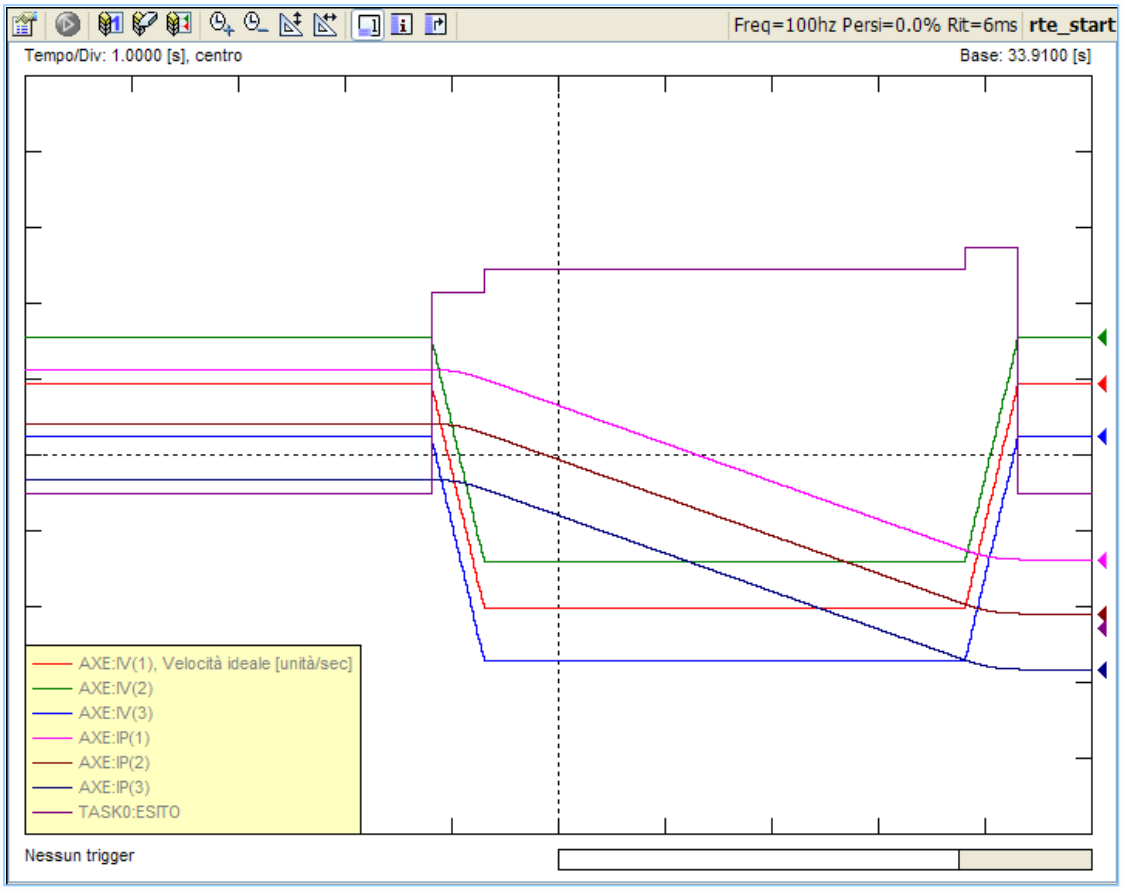
```
; esempio di tre assi che si spostano interpolati avanti ed indietro t
; definizioni
stru_mvitp 879 m1[3]
int restartFlag
int holdflg

motion
if(first_time() 346)
    rr(1000) = 100 ; target ax 1
    rr(1001) = 10  ; vel
    rr(1002) = 20  ; acc
    rr(1003) = 20  ; dec
    rr(2000) = 100 ; target ax 2
    rr(2001) = 10  ; vel
    rr(2002) = 20  ; acc
    rr(2003) = 20  ; dec
    rr(3000) = 100 ; target ax 3
    rr(3001) = 10  ; vel
    rr(3002) = 20  ; acc
    rr(3003) = 20  ; dec
    restartFlag = true
endif

; ASSE 1
```

© 2025 Robox SpA

```
;*********
; numero asse
m1[0].nAx = 1
; posizione desiderata [units]
m1[0].target = rr(1000)
; velocità desiderata [units/sec]
m1[0].v = rr(1001)
; accelerazione [units/sec²]
m1[0].acc = rr(1002)
; decelerazione [units/sec²]
m1[0].dec = rr(1003)
; valore di dec hold
m1[0].decH = max_acc(1)
; ASSE 2
;*********
; numero asse
m1[1].nAx = 2
; posizione desiderata [units]
m1[1].target = rr(2000)
; velocità desiderata [units/sec]
m1[1].v = rr(2001)
; accelerazione [units/sec²]
m1[1].acc = rr(2002)
; decelerazione [units/sec²]
m1[1].dec = rr(2003)
; valore di dec hold
m1[1].decH = max_acc(2)
; ASSE 3
;********
; numero asse
m1[2].nAx = 3
; posizione desiderata [units]
m1[2].target = rr(3000)
; velocità desiderata [units/sec]
m1[2].v = rr(3001)
; accelerazione [units/sec²]
m1[2].acc = rr(3002)
; decelerazione [units/sec²]
m1[2].dec = rr(3003)
; valore di dec hold
m1[2].decH = max_acc(3)
esito = mva_to_n(restartFlag, m1[0], 3, hold, fr)
if(rise(esito = M_REACHED))
    if(avanti)
        inc(rr(1000),100)
        inc(rr(2000),100)
        inc(rr(3000),100)
        avanti = false
        restartFlag = true
    else
        dec(rr(1000),100)
```

© 2025 Robox SpA

```
        dec(rr(2000),100)
        dec(rr(3000),100)
        avanti = true
        restartFlag = true
    endif
endif
end_motion
```



Funzione mva_to_n_cj()

Funzione che porta gli assi (1-:-32) indicati verso il valore di posizione target prefissato con un profilo velocità verso tempo a 'esse' (con jerk controllato).

La conseguenza è un movimento più dolce.

Sintassi	**i32 mva_to_n_cj (i32 start, stru_mvitpcj stru [, i32 nDef=1] [, i32 holdFlag=0] [, real fr=1])**
start	Flag per abilitare la riassunzione dei dati (solo se assi fermi). Il flag viene azzerato da RTE quando i dati vengono assunti.

© 2025 Robox SpA

Sintassi	**i32 mva_to_n_cj (i32 start, stru_mvitpcj stru [, i32 nDef=1] [, i32 holdFlag=0] [, real fr=1])**
stru	Struttura di tipo stru_mvitpcj 880 contenente i parametri relativi alla movimentazione.
nDef	Dimensione dell'array di informazione (numero di assi coinvolti). (opzionale, default 1)
holdFlag	Flag per comandare lo stato di HOLD alla missione. (opzionale, default 0)
fr	Valore di riduzione delle velocità (0-1). (opzionale, default 1) Vedi fr.
Risultato	Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63).

Valore	Descrizione
-5 [M_ILLEGAL_ARGS]	Parametro illegale.
-6 [M_IN_ALARM]	Allarme (ip asse già impostata).
2 [M_POSITIONING]	Posizionamento iniziale.
6 [M_REACHED]	Target raggiunto. Ciò significa che o la direzione è 0 e l'asse è fermo, oppure la direzione è +1 o -1 e l'asse ha raggiunto uno dei paracarri.
14 [M_HOLDED]	Movimento arrestato causa flag HOLD. Viene restituito anche nei casi in cui la velocità desiderata dell'asse viene impostata a 0 oppure il feedrate viene impostato a zero.
15 [M_ACCEL]	Accelerazione in corso.
16 [M_AT_SPEED]	Movimentazione a regime.
17 [M_DECEL]	Decelerazione in corso.

Validità	Rule.

© 2025 Robox SpA

Sintassi	**i32 mva_to_n_cj (i32 start, stru_mvitpcj stru [, i32 nDef=1] [, i32 holdFlag=0] [, real fr=1])**
Note	La missione può coinvolgere più assi (max 32). Essi verranno interpolati linearmente al fine di raggiungere tutti insieme i loro target.

Diagnostica

- Genera allarme 9900 ill. Arg. A:2 T:x ST:xxx se numero di asse illecito o accelerazione/velocità/jerk < 1e-9
- Genera allarme 9900 ill. Arg. A:3 T:x ST:xxx se *acc* = 0
- Genera allarme 9903 Par. are changed T:x ST:xxx se cambiata sequenza assi in corsa
- Genera allarme 2800 Double Position Set T:x ST:xxx se già imposta la ip dell'asse

Esempi

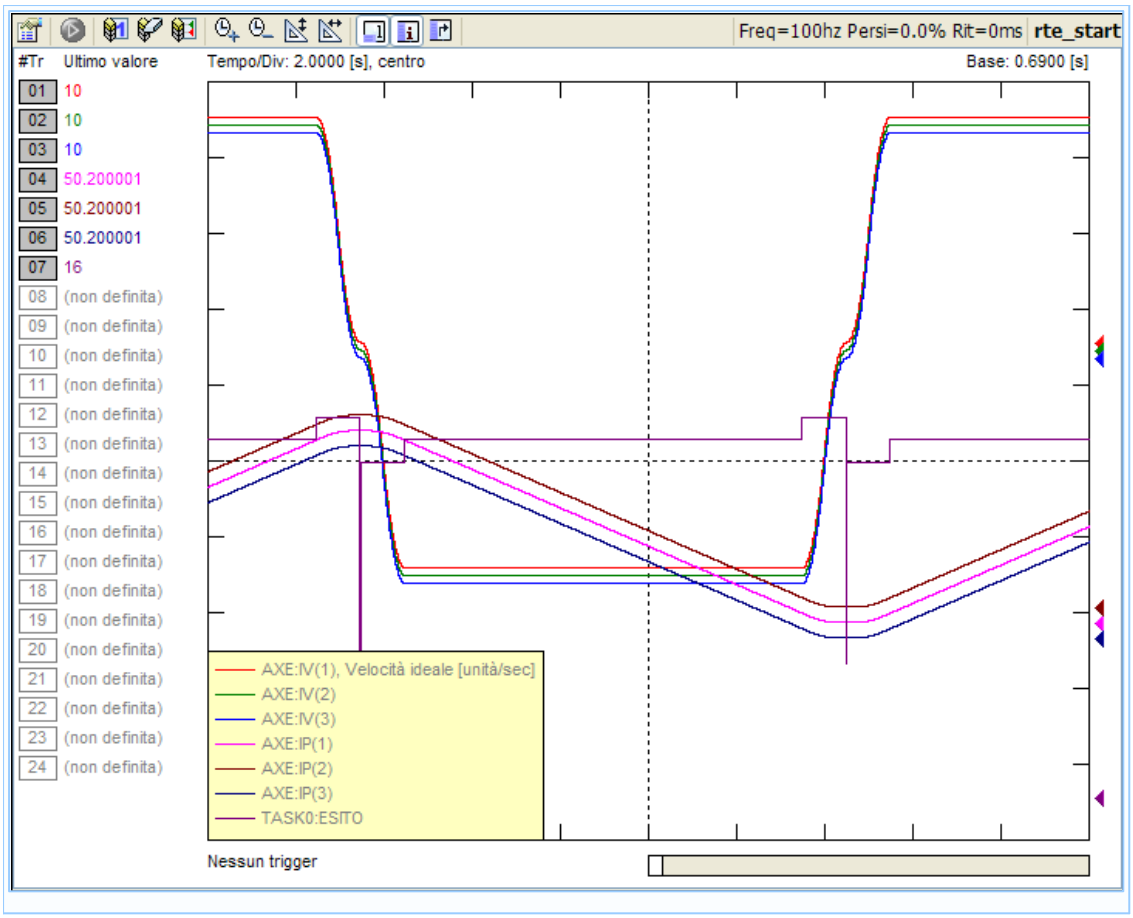
Esempio di utilizzo:

```
; esempio di tre assi che si spostano interpolati avanti ed indietro t
; definizioni
stru_mvitpcj 880 m1[3]
int restartFlag

motion
if(first_time 346())
    rr(1000) = 100 ; target ax 1
    rr(1001) = 10  ; vel
    rr(1002) = 20  ; acc/dec
    rr(1003) = 40  ; jerk
    rr(2000) = 100 ; target ax 2
    rr(2001) = 10  ; vel
    rr(2002) = 20  ; acc/dec
    rr(2003) = 40  ; jerk
    rr(3000) = 100 ; target ax 3
    rr(3001) = 10  ; vel
    rr(3002) = 20  ; acc/dec
    rr(3003) = 40  ; jerkdec
    restartFlag = true
endif
; ASSE 1
;*********
; numero asse
m1[0].nAx = 1
; posizione desiderata [units]
m1[0].target = rr(1000)
; velocità desiderata [units/sec]
m1[0].v = rr(1001)
; accelerazione/decelerazione [units/sec²]
```

© 2025 Robox SpA

```
m1[0].acc = rr(1002)
; jerk [units/sec²]
m1[0].dec = rr(1003)
; valore di dec hold
m1[0].decH = max_acc(1)
; ASSE 2
;*********
; numero asse
m1[1].nAx = 2
; posizione desiderata [units]
m1[1].target = rr(2000)
; velocità desiderata [units/sec]
m1[1].v = rr(2001)
; accelerazione/decelerazione [units/sec²]
m1[1].acc = rr(2002)
; jerk [units/sec²]
m1[1].dec = rr(2003)
; valore di dec hold
m1[1].decH = max_acc(2)
; ASSE 3
;********
; numero asse
m1[2].nAx = 3
; posizione desiderata [units]
m1[2].target = rr(3000)
; velocità desiderata [units/sec]
m1[2].v = rr(3001)
; accelerazione/decelerazione [units/sec²]
m1[2].acc = rr(3002)
; jerk [units/sec²]
m1[2].dec = rr(3003)
; valore di dec hold
m1[2].decH = max_acc(3)
esito = mva_to_n_cj(500)(restartFlag, m1[0], 3, hold, fr)
if(rise(562)(esito = M_REACHED))
    if(avanti)
    inc(377)(rr(1000),100)
    inc(rr(2000),100)
    inc(rr(3000),100)
    avanti = false
    restartFlag = true
else
    dec(295)(rr(1000),100)
    dec(rr(2000),100)
    dec(rr(3000),100)
    avanti = true
    restartFlag = true
endif
end_motion
```

© 2025 Robox SpA



Funzione mva_to_n_v()

Funzione che porta gli assi (1-:-32) indicati verso il valore di posizione target prefissato con un profilo velocità verso tempo trapezoidale.

NOTA: La differenza con la funzione mva_to_n 496() consiste nel fatto che può essere chiamata ad asse/i già in movimento e può essere specificata la velocità finale sul target, anche diversa da zero.

Sintassi	**i32 mva_to_n_v (i32 start, stru_mvgenitp stru [, i32 nDef=1] [, i32 holdFlag=0] [, real fr=1])**
start	Flag per abilitare la riassunzione dei dati (anche ad assi in movimento). Il flag viene azzerato quando i dati vengono assunti.
stru	Struttura di tipo stru_mvgenitp 877 contenente i parametri relativi alla movimentazione.
nDef	Dimensione dell'array di informazione (numero di assi coinvolti). (opzionale, default 1)

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 mva_to_n_v (i32 start, stru_mvgenitp stru [, i32 nDef=1] [, i32 holdFlag=0] [, real fr=1])</td></tr>
<tr><td>holdFlag</td><td>Flag per comandare lo stato di HOLD alla missione. (opzionale, default 0)</td></tr>
<tr><td>fr</td><td>Valore di riduzione delle velocità (0-1). (opzionale, default 1)
Vedi fr.</td></tr>
<tr><td>Risultato</td><td>Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63).
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td>-5 [M_ILLEGAL_ARGS]</td><td>Parametro illegale (acc=0 o numero asse illegale).</td></tr>
<tr><td>-6 [M_IN_ALARM]</td><td>Allarme (ip asse già impostata).</td></tr>
<tr><td>6 [M_REACHED]</td><td>Target raggiunto.
Ciò significa che o la direzione è 0 e l'asse è fermo, oppure la direzione è +1 o -1 e l'asse ha raggiunto uno dei paracarri.</td></tr>
<tr><td>14 [M_HOLDED]</td><td>Movimento arrestato causa flag HOLD.
Viene restituito anche nei casi in cui la velocità desiderata dell’asse viene impostata a 0 oppure il feedrate viene impostato a zero.</td></tr>
<tr><td>15 [M_ACCEL]</td><td>Accelerazione in corso.</td></tr>
<tr><td>16 [M_AT_SPEED]</td><td>Movimentazione a regime.</td></tr>
<tr><td>17 [M_DECEL]</td><td>Decelerazione in corso.</td></tr>
</table></td></tr>
<tr><td>Validità</td><td>Rule.</td></tr>
<tr><td>Note</td><td>La missione può coinvolgere più assi (max 32).
Essi verranno interpolati linearmente al fine di raggiungere tutti insieme i loro target.
Se il target è distante meno di (pos_thr/10), RTE considera lo spazio nullo. Quindi nel caso l'utente desideri effettuare dei micro</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 mva_to_n_v (i32 start, stru_mvgenitp stru [, i32 nDef=1] [, i32 holdFlag=0] [, real fr=1])**
	avanzamenti (< pos_thr/10) dovrà impostare la variabile pos_thr opportunamente.

Esempi

Esempio di utilizzo:

```
; esecuzione in un piano x y di una forma a trapezio
; vertice A (0,0)
; vertice B (30,100)
; vertice C (70,100)
; vertice D (100,0)
; inizializzazioni variabili
int fase, restartFlag, esito
stru_mvgenitp 877 m1[2]
; da inserire nella rule
motion
if(first_time 346())
    fase = 1
endif

select(fase)
case 1  ; x muove verso punto B
    rr(1000) = 30  ; target ax X
    rr(1001) = 10  ; vel regime
    rr(1002) = 10  ; vel fin
    rr(1003) = 10  ; acc ini
    rr(1004) = 0   ; acc fin

    ; y muove verso punto B
    rr(2000) = 100 ; target ax Y
    rr(2001) = 10  ; vel regime
    rr(2002) = 0   ; vel fin
    rr(2003) = 10  ; acc ini
    rr(2004) = 0   ; acc fin
    restartFlag = true
    fase = 2
    break

case 2
    if(esito = M_REACHED)
        fase = 3
    endif
    break

case 3
    ; x muove verso punto C
    rr(1000) = 70  ; target ax X
```

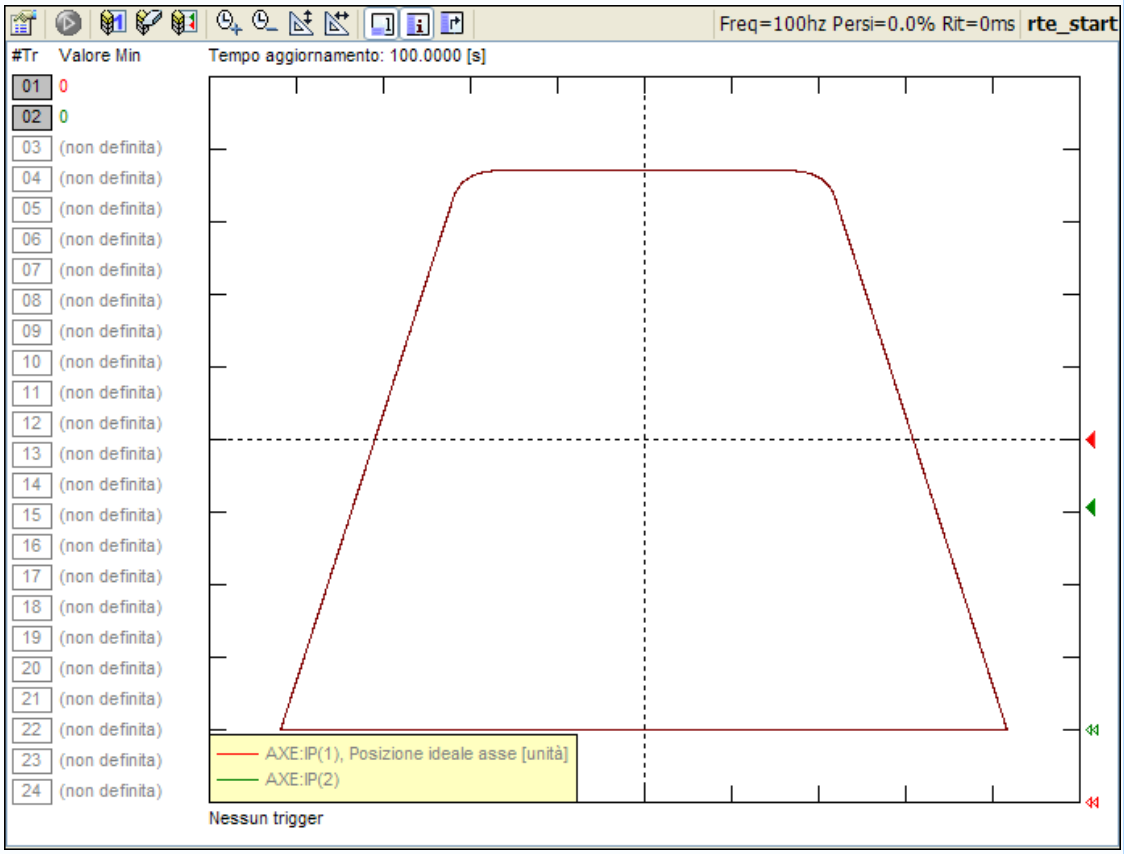
© 2025 Robox SpA

```
    rr(1001) = 10   ; vel regime
    rr(1002) = 10   ; vel fin
    rr(1003) = 10   ; acc ini
    rr(1004) = 0    ; acc fin
    ; y muove verso punto C
    rr(2000) = 100 ; target ax Y
    rr(2001) = 0    ; vel regime
    rr(2002) = 0    ; vel fin
    rr(2003) = 10   ; acc ini
    rr(2004) = 0    ; acc fin
    restartFlag = true
    fase = 4
    break

case 4
    if(esito = M_REACHED)
        fase = 5
    endif
    break

case 5
    ; x muove verso punto D
    rr(1000) = 100 ; target ax X
    rr(1001) = 10   ; vel regime
    rr(1002) = 0    ; vel fin
    rr(1003) = 10   ; acc ini
    rr(1004) = 0    ; acc fin
    ; y muove verso punto D
    rr(2000) = 0    ; target ax Y
    rr(2001) = 10   ; vel regime
    rr(2002) = 0    ; vel fin
    rr(2003) = 10   ; acc ini
    rr(2004) = 0    ; acc fin
    restartFlag = true
    fase = 6
    break

case 6
    if(esito = M_REACHED)
        fase = 7
    endif
    break

case 7
    ; x muove verso punto A
    rr(1000) = 0    ; target ax X
    rr(1001) = 10   ; vel regime
    rr(1002) = 0    ; vel fin
    rr(1003) = 10   ; acc ini
    rr(1004) = 0    ; acc fin
```

© 2025 Robox SpA

```
    ; y muove verso punto A
    rr(2000) = 0    ; target ax Y
    rr(2001) = 0    ; vel regime
    rr(2002) = 0    ; vel fin
    rr(2003) = 10   ; acc ini
    rr(2004) = 0    ; acc fin
    restartFlag = true
    fase = 8
    break

case 8
    if(esito = M_REACHED)
        fase = 1
    endif
    break
end_select

; ASSE 1
;*********
; numero asse
m1[0].nAx = 1
; posizione desiderata [units]
m1[0].target = rr(1000)
; velocità iniziale [units/sec]
m1[0].v = rr(1001)
; velocità finale [units/sec]
m1[0].vf = rr(1002)
; accelerazione iniziale [units/sec²]
m1[0].acci = rr(1003)
; accelerazione finale [units/sec²]
m1[0].accf = rr(1004)
; valore di dec hold
m1[0].accH = max_acc(1)
; ASSE 2
;*********
; numero asse
m1[1].nAx = 2
; posizione desiderata [units]
m1[1].target = rr(2000)
; velocità iniziale [units/sec]
m1[1].v = rr(2001)
; velocità finale [units/sec]
m1[1].vf = rr(2002)
; accelerazione iniziale [units/sec²]
m1[1].acci = rr(2003)
; accelerazione finale [units/sec²]
m1[1].accf = rr(2004)
; valore di dec hold
m1[1].accH = max_acc(2)
esito = mva_to_n_v(restartFlag, m1[0] , 2, hold, fr)
end_motion
```

© 2025 Robox SpA

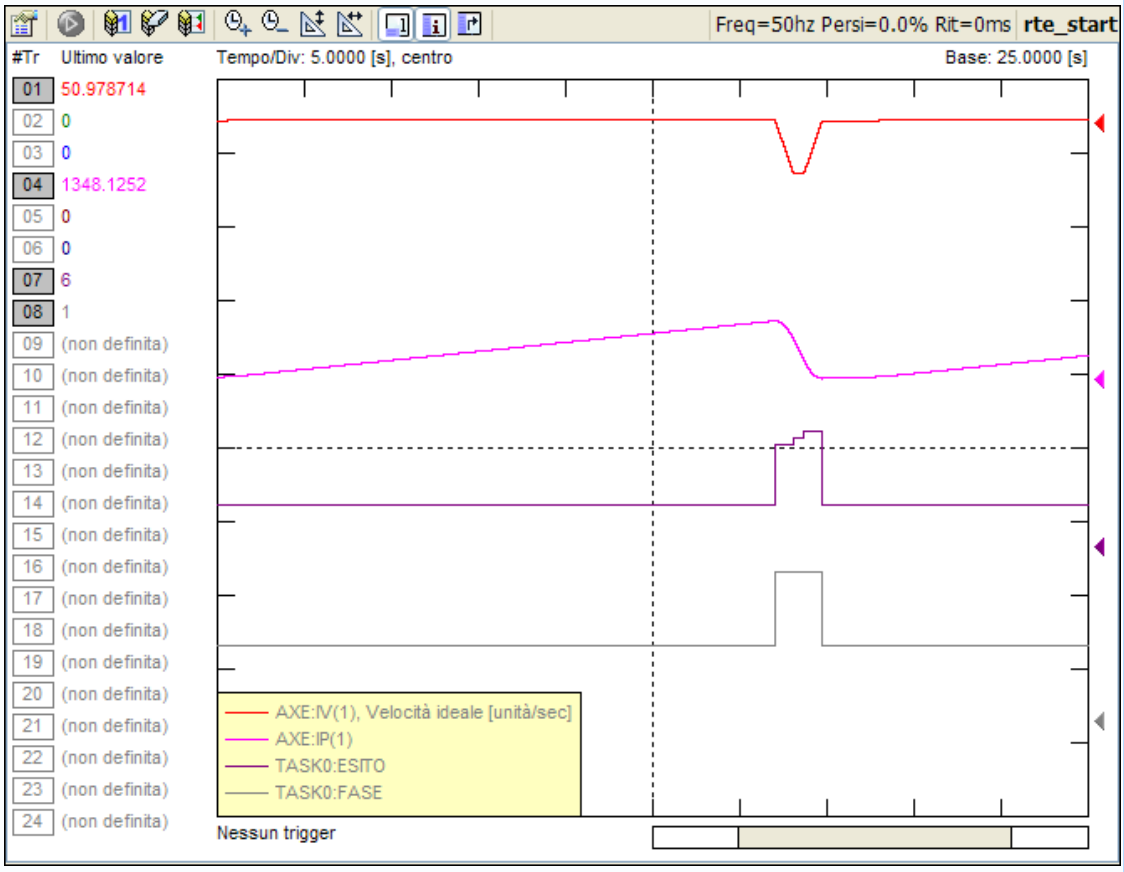


Esempio di utilizzo 2:

```
; sul fronte di salita input 1 l'asse 1 che sta viaggiando con la sua
; rapidamente a quota di home con parametri specificati
motion
if(first_time())
    fase = 1
endif
; se richiesto rientro a quota 0
if(rise(inp(1)))   ; torna alla base
    fase = 2
    ; numero asse
    m1[0].nAx = 1
    ; posizione desiderata [units]
    m1[0].target = 0
    ; velocità di crociera [units/sec]
    m1[0].v = max_spe(1)
    ; velocità finale [units/sec]
    m1[0].vf = 0
    ; accelerazione iniziale [units/sec²]
    m1[0].acci = max_acc(1)
    ; accelerazione finale [units/sec²]
    m1[0].accf = 0
```

© 2025 Robox SpA

```
    ; valore di dec hold
    m1[0].accH = max_acc(1)
    restartFlag = true ; assumi dati
endif
; ciclo
if(fase = 1)  ; vai con velocità sinusoidale
    iv(1) = ramp555(iv(1), 50 + sin602(torad647(mod439(tfb*100, 360))),
else   ; torna a quota 0
    esito = mva_to_n_v(restartFlag, m1[0], 1, hold, fr)
    if(esito = M_REACHED)
        fase = 1
    endif
endif
end_motion
```



© 2025 Robox SpA

Funzione mva_zc()

Funzione che esegue il ciclo di zero dell'asse.

Sintassi	**i32 mva_zc (i32 start, stru_mvzc stru [, i32 nDef=1] [, i32 holdFlag=0] [, real fr=1])**
start	Flag per abilitare la riassunzione dei dati (solo se assi fermi). Il flag viene azzerato da RTE quando i dati vengono assunti.
stru	Struttura di tipo stru_mvzc 892 contenente i parametri relativi alla movimentazione.
nDef	Dimensione dell'array di informazione (numero di assi coinvolto). (opzionale, default 1)
holdFlag	Flag per comandare lo stato di HOLD alla missione. (opzionale, default 0)
fr	Valore di riduzione delle velocità (0-1). (opzionale, default 1) Vedi fr.
Risultato	Variabile contenente informazioni relative allo stato della funzione (vedi codici di reso 63).

Valore	Descrizione
-5 [M_ILLEGAL_ARGS]	Parametro illegale.
-6 [M_IN_ALARM]	Allarme (ip asse già impostata).
6 [M_REACHED]	Ciclo di zero terminato con successo.
7 [M_HOMING]	Ciclo di zero in esecuzione.

Validità	Rule.
Note	Se si specificano n assi le movimentazioni sono contemporanee. È a cura di RTE l'aggiornamento della variabile predefinita c0_done se l'esecuzione del ciclo di zero è terminata con successo. È possibile aumentare la diagnostica nel report tramite la variabile predefinita sys_flag.14.

NOTA: L'indice del canale del micro va specificato nel configuratore di progetto -> assi -> trasduttore, può essere sovrascritto tramite la variabile predefinita u0_channel.

NOTA: Da RTE 34.23.5 è possibile modificare il valore di preset della posizione fino a che non viene effettivamente usato. Il che vuol dire che il valore di preset imposto sarà l'ultimo valore scritto nei parametri dell'istruzione.

© 2025 Robox SpA

Diagnostica

- Genera allarme 9900 ill. A:1 T:x ST:xxx se numero di asse illecito
- Genera allarme 9900 ill. A:3 T:x ST:xxx se parametro *velPos* = 0
- Genera allarme 9900 ill. A:4 T:x ST:xxx se parametro *velZC* = 0
- Genera allarme 9900 ill. A:5 T:x ST:xxx se parametro *accel* = 0
- Genera allarme 9900 ill. A:6 T:x ST:xxx se parametro *jerk* = 0
- Genera allarme 9900 ill. A:8 T:x ST:xxx se il ciclo di zero programmato non è applicabile
- Genera allarme 9900 ill. A:9 T:x ST:xxx se i flag non sono congrui
- Genera allarme 9900 ill. A:10 T:x ST:xxx se i flag 'abilitazione al riposizionamento sulla quota di preset' (*zcFlag*.9) e 'abilitazione riposizionamento all'interno dei paracarri software' (*zcFlag*.10) sono di tipo esclusivo
- Genera allarme 9900 ill. A:11 T:x ST:se il movimento monodirezionale è incompatibile con richiesta di posizionamento all'interno dei paracarri o su punto di preset
- Genera allarme 9900 ill. A:12 T:x ST:xxx se non è possibile effettuare la foto dal dispositivo
- Genera allarme 9903 Par. are changed T:x ST:xxx se i parametri vengono cambiati durante la movimentazione
- Genera allarme 2800 #nAx Double Position Set T:x ST:xxx se già imposta la IP dell'asse
- Genera allarme 2000#nAx Unable to perform homing

Tipologie di homing

Tipo di asse	Tipo di ciclo di zero
Asse lineare	Ciclo di zero a micro 69
Asse lineare	Ciclo di zero a micro + index (trasduttore encoder incrementale) 74
Asse lineare	Ciclo di zero a paracarro 79
Asse lineare	Ciclo di zero a paracarro + index (trasduttore encoder incrementale) 82
Asse rotativo monodirezionale	Ciclo di zero a micro con ricerca preliminare della variazione del micro 99
Asse rotativo monodirezionale	Ciclo di zero a micro + index (trasduttore encoder incrementale) con ricerca preliminare della variazione del micro 88
Asse rotativo monodirezionale	Ciclo di zero a micro senza ricerca preliminare della variazione del micro 104
Asse rotativo monodirezionale	Ciclo di zero a micro + index (trasduttore encoder incrementale) senza ricerca preliminare della variazione del micro 94

© 2025 Robox SpA

Funzione normalize()

Funzione che permette di calcolare un valore normalizzato fra 0 e 1.

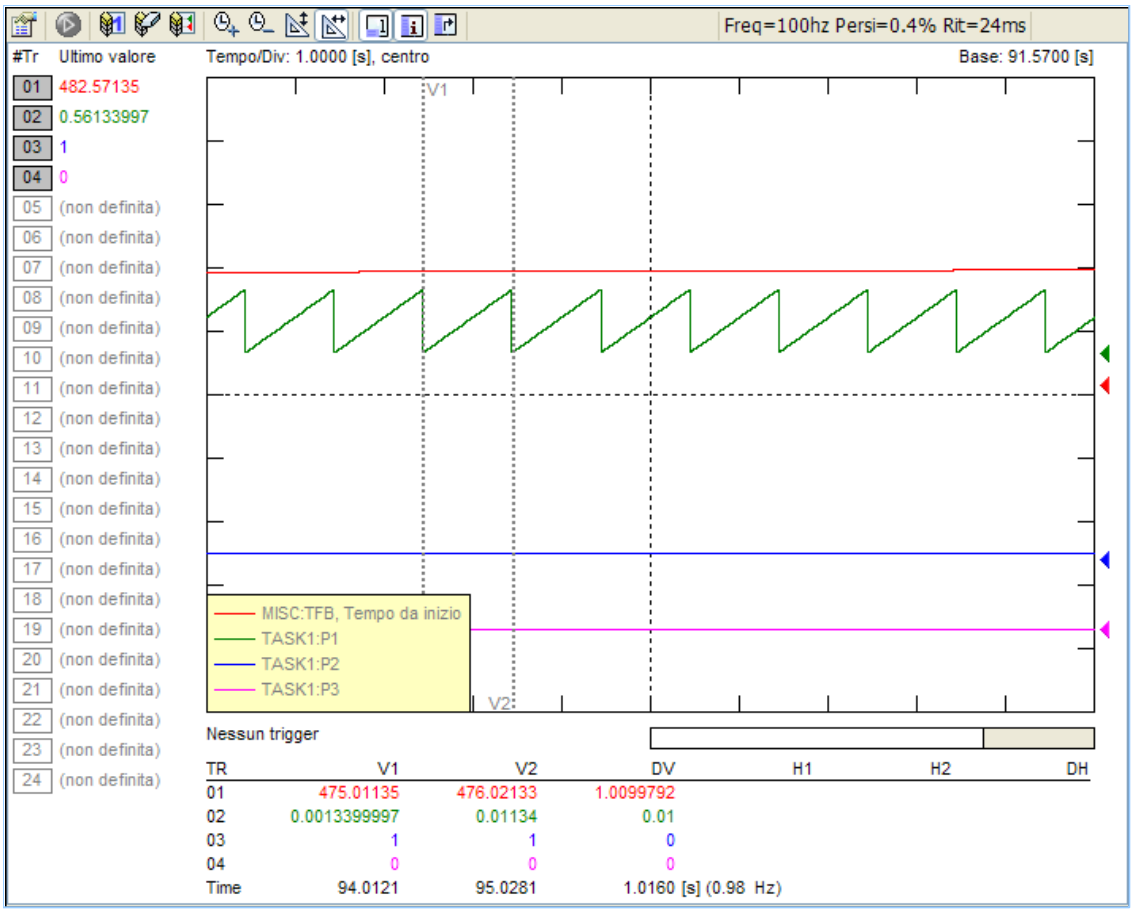
Sintassi	**real normalize (real value, real offset, real cycle)**
value	Espressione che fornisce il dato da normalizzare.
offset	Espressione che fornisce la posizione di inizio ciclo.
cycle	Espressione che fornisce la lunghezza del ciclo di riferimento.
Risultato	Rende un valore compreso fra 0.0 (incluso) e 1.0 (escluso), calcolato con la seguente formula: result = abs_mod 157 ((*value* - *offset*), *cycle*)
Validità	Rule, Task.
Note	Se il parametro entrante *value* coincide esattamente con l'inizio/fine di un ciclo, la funzione normalize() rende sempre 0.0. La funzione pertanto non restituisce mai il valore 1.0.

Esempi

Esempio di utilizzo:

```
p1 = normalize 513(tfb, p3, p2)   ; p3 = 0 e p2 = 1
```

© 2025 Robox SpA



Funzione onoff_filt()

Funzione che permette di cogliere sia il fronte di salita di una condizione che il fronte di discesa di una condizione impostando dei filtri in spazio o tempo.

Funzionamento

Al rilevamento del fronte di salita della condizione viene memorizzato il valore della variabile di filtro (*varFilt*) corrispondente.

Finché la differenza tra il valore attuale della variabile di filtro e quello memorizzato è inferiore alla soglia specificata (*thrTrue*) la funzione rende FALSE, mentre quando la differenza diventa maggiore della soglia specificata la funzione rende TRUE.

Solo quando la funzione sta rendendo TRUE, al rilevamento del fronte di discesa della condizione viene memorizzato il valore della variabile di filtro (*varFilt*) corrispondente.

Finché la differenza tra il valore attuale della variabile di filtro e quello memorizzato è inferiore alla soglia specificata (*thrFalse*) la funzione continua a rendere TRUE, mentre quando la differenza diventa maggiore della soglia specificata la funzione rende FALSE.

© 2025 Robox SpA

Sintassi	**bool onoff_filt (bool cond, real varFilt, real thrTrue [, real thrFalse=thrTrue])**
cond	Condizione da valutare per la ricerca del fronte.
varFilt	Grandezza evolvente (tipicamente tempo (tfb) o spazio ip(n)).
thrTrue	Quantità di *varFilt* in cui si deve vedere stabilmente la condizione vera.
thrFalse	Quantità di *varFilt* in cui si deve vedere stabilmente la condizione falsa. (opzionale, default *thrTrue*)
Risultato	Rende: Valore / Condizione FALSE / All'avvio se *cond* = FALSE. FALSE / *cond* = FALSE e si è superato la soglia *thrFalse*. FALSE / *cond* = TRUE ma non si è superato la soglia *thrTrue*. TRUE / *cond* = TRUE e si è superato la soglia *thrTrue*. TRUE / *cond* = FALSE ma non si è superata la soglia *thrFalse*.
Validità	Rule, Task.
Note	

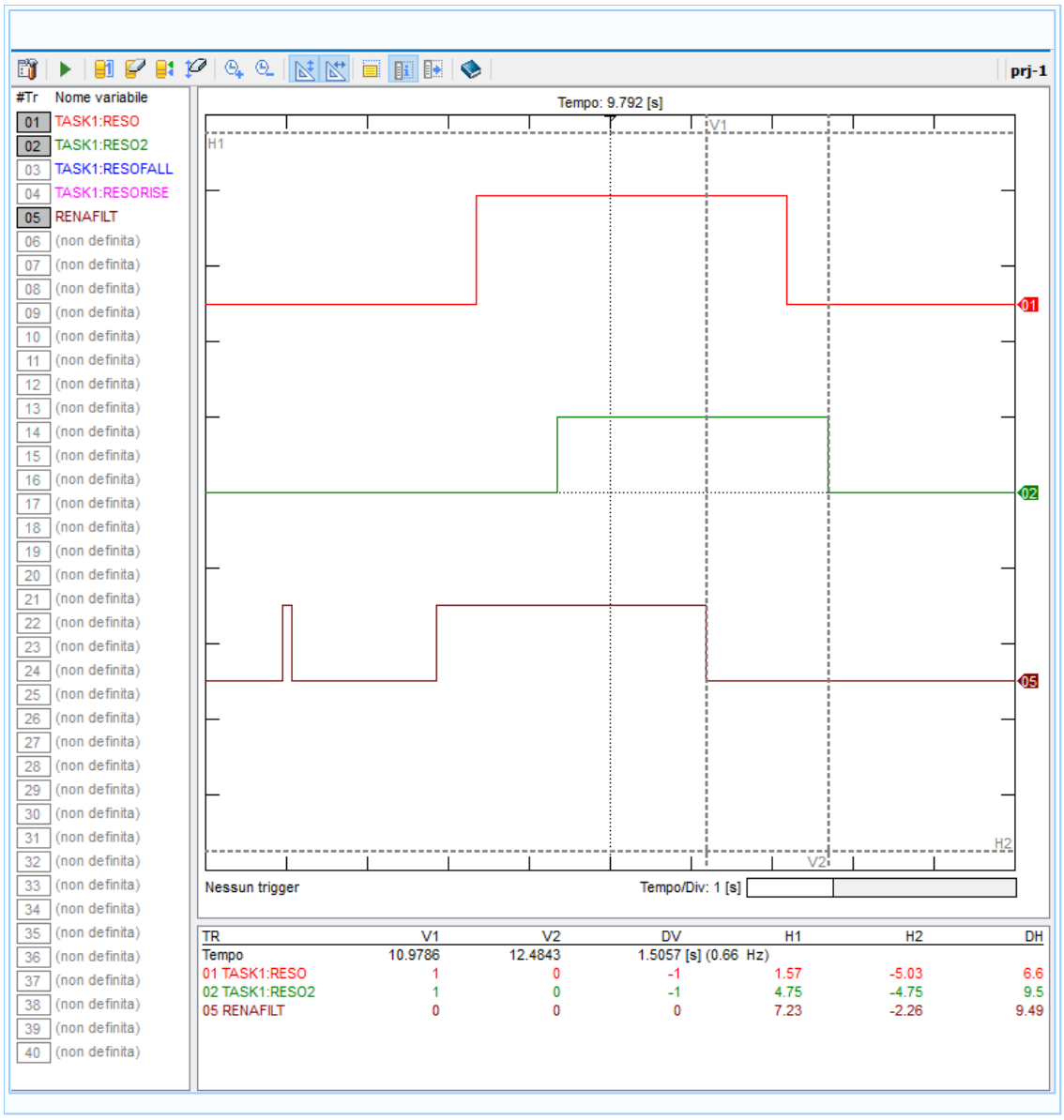
Fornendo come variabile di filtro ad esempio ip(n) si avrà un filtro in spazio, mentre fornendo come variabile di filtro tfb si avrà un filtro in tempo (unità secondi).

Esempi

Esempio di utilizzo:

```
bool reso = false
bool reso2 = false

; dopo 0.5s da quando si ha la condizione vera, setto l'uscita a TRUE,
; dopo 1s da quando la condizione diventa falsa, setto l'uscita a FALS
reso  = onoff_filt(rEnaFilt, tfb, 0.5, 1)
; imposto la stessa soglia di 1.5s sia per il fronte di salita che per
reso2 = onoff_filt(rEnaFilt, tfb, 1.5)
```

© 2025 Robox SpA



Funzione par_even()

Funzione che rende un valore TRUE se il numero di bit nello stato alto (1), nei byte indicati di una variabile, è pari.

Sintassi	**i32 par_even (i32 value [, i32 nByte=1])**
value	Valore di cui calcolare la parità.
nByte	Numero di byte. (opzionale, default 1)
Risultato	Rende:

© 2025 Robox SpA

Sintassi	**i32 par_even (i32 value [, i32 nByte=1])**
	Valore / Descrizione > 0 — Numero di bit a 1 pari. 0 — Altrimenti.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
if(par_even(r(100)))     ; 1 byte di default
    ; qui ci arriviamo se il numero di bit a 1 nel byte basso di R(100
    r(101) = r(101)+1
end_if


if(par_even(r(100), 4))     ; tutti e 4 i byte
    ; qui ci arriviamo se il numero di bit a 1 in R(100) è pari
    r(101) = r(101)+1
end_if
```

Funzione par_odd()

Funzione che rende un valore TRUE se il numero di bit nello stato alto (1), nei byte indicati di una variabile, è dispari.

Sintassi	**i32 par_odd (i32 value [, i32 nByte=1])**
value	Valore di cui calcolare la parità.
nByte	Numero di byte. (opzionale, default 1)
Risultato	Rende: Valore / Descrizione > 0 — Numero di bit a 1 dispari. 0 — Altrimenti.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

© 2025 Robox SpA

```
if(par_odd(r(100)))     ; 1 byte di default
    ; qui ci arriviamo se il numero di bit a 1 nel byte basso di R(100
    r(101) = r(101)+1
end_if


if(par_odd(r(100), 4))     ; tutti e 4 i byte
    ; qui ci arriviamo se il numero di bit a 1 in R(100) è dispari
    r(101) = r(101)+1
end_if
```

Funzione planar_comp_cp()

Funzione per applicare la compensazione inversa nella gestione della Planar Compensation Table (PCT).

Sintassi	**i32 planar_comp_cp (stru_aax posIn, stru_aax posOut)**
posIn	Struttura di tipo stru_aax contenente le posizioni CP degli assi.
posOut	Struttura di tipo stru_aax contenente le posizioni compensate CP_C degli assi.
Risultato	Rende sempre 0.
Validità	Rule.
Requisiti di sistema	RTE 34.16.0
Note	

Esempi

Esempio di utilizzo:

```
; rule prologue
; -------------
struct stru_aax
    real axp[1+32]     ; array per le posizioni
    ; l'indice 0 non si usa, si usano quindi axp[1]....axp[32]
end_struct

; Sample code

; Global Storage definition
; -------------------------
stru_aax    in_cp, out_cp
stru_aax    in_ip, out_ip

...
```

© 2025 Robox SpA

```
; usiamo solo 3 assi
in_ip.axp[1] = ip(1)
in_ip.axp[2] = ip(2)
in_ip.axp[3] = ip(3)
result = planar_comp_ip519(in_ip, out_ip)

rr(120+1) = out_ip.axp[1]
rr(120+2) = out_ip.axp[2]
rr(120+3) = out_ip.axp[3]

in_cp.axp[1] = cp(1)
in_cp.axp[2] = cp(2)
in_cp.axp[3] = cp(3)

esito = planar_comp_cp518(in_cp, out_cp)

rr(160+1) = out_cp.axp[1]
rr(160+2) = out_cp.axp[2]
rr(160+3) = out_cp.axp[3]
```

Funzione planar_comp_ip()

Funzione per applicare la compensazione diretta nella gestione della Planar Compensation Table (PCT).

Sintassi	**i32 planar_comp_ip (stru_aax posIn, stru_aax posOut)**
posIn	Struttura di tipo stru_aax822 contenente le posizioni IP degli assi.
posOut	Struttura di tipo stru_aax contenente le posizioni compensate IP_C degli assi.
Risultato	Rende sempre 0.
Validità	Rule.
Requisiti di sistema	RTE 34.16.0
Note	

Esempi

Esempio di utilizzo:

```
; rule_prologue759
; -------------
struct stru_aax
    real axp[1+32]     ; array per le posizioni
    ; l'indice 0 non si usa, si usano quindi axp[1]....axp[32]
end_struct
```

© 2025 Robox SpA

```
; Sample code

; Global Storage definition
; -------------------------
stru_aax    in_cp, out_cp
stru_aax    in_ip, out_ip

...

; usiamo solo 3 assi
in_ip.axp[1] = ip(1)
in_ip.axp[2] = ip(2)
in_ip.axp[3] = ip(3)
result = planar_comp_ip519(in_ip, out_ip)

rr(120+1) = out_ip.axp[1]
rr(120+2) = out_ip.axp[2]
rr(120+3) = out_ip.axp[3]

in_cp.axp[1] = cp(1)
in_cp.axp[2] = cp(2)
in_cp.axp[3] = cp(3)

esito = planar_comp_cp518(in_cp, out_cp)

rr(160+1) = out_cp.axp[1]
rr(160+2) = out_cp.axp[2]
rr(160+3) = out_cp.axp[3]
```

Funzione pos_capture_cmd()

Questa funzione permette di abilitare la cattura della foto della quota di un asse via hardware (precisione massima).

Sintassi	**i32 pos_capture_cmd (i32 nAx, i32 micro, i32 edge)**
nAx	Numero di asse (1-:-32).
micro	Tipo di trasduttore:

Trasduttore selezionato	Valore	Descrizione
ENCODER INCREMENTALE	< 0	Disabilita cattura.
	0	Solo canale di 0.
	1	Micro di 0 associato più canale di 0.

© 2025 Robox SpA

Sintassi	**i32 pos_capture_cmd (i32 nAx, i32 micro, i32 edge)**

Trasduttore selezionato	Valore	Descrizione
CANopen/EtherCAT Parker - SBC (tipo HID)	< 0	Disabilita cattura.
	0	Foto immediata dell'inizio del giro resolver o foto su canale.
	1	input0 su drive.
	2	input1 su drive.
	3	input2 su drive.
CANopen/EtherCAT Parker - SBC (tipo SLVD/TPDM)	< 0	Disabilita cattura.
	0	Foto immediata dell'inizio del giro resolver o foto su canale.
	1	input0 su drive. Su TPDM le foto possono essere effettuate su più input se tale prestazione è abilitata.
EtherCAT Robox IMD20	< 0	Disabilita cattura.
	0	Foto immediata dell'inizio del giro resolver o foto su canale.
	1	input1 su drive.
	2	input2 su drive.

© 2025 Robox SpA

Sintassi	**i32 pos_capture_cmd (i32 nAx, i32 micro, i32 edge)**

Trasduttore selezionato	Valore	Descrizione
CANopen/EtherCAT Phase AXM (necessita di programma PLC a bordo AXM)	< 0	Disabilita cattura.
	0	Foto immediata dell'inizio del giro resolver o foto su canale.
	1	input0 su drive.
	2	input1 su drive.
	3	input2 su drive.
	4	input3 su drive.
CANopen/EtherCAT Phase Ultract TW	< 0	Disabilita cattura. (non ancora implementato)
	0	Foto immediata dell'inizio del giro resolver o foto su canale.
	1	input0 su drive.
CANopen Sinamics RCB	-10 -11 -12 -13 -14 -15 -16	Foto hardware su input specificato.
	< 0 (diverso dai precedenti)	Disabilita cattura.
	0	Foto immediata dell'inizio del giro resolver o foto su canale.
	1 -:- 16	Foto software su input specificato. (latenza 1 ms)

edge	Flag a bit:

© 2025 Robox SpA

Sintassi	**i32 pos_capture_cmd (i32 nAx, i32 micro, i32 edge)**

Bit	Valore esadecimale	Stato bit	Descrizione
0	0x00000001	0	Fronte di discesa. (se previsto da hw)
		1	Fronte di salita. (se previsto da hw) (usato dove applicabile)
1	0x00000002	1	Cattura continua.
2	0x00000004	1	Disabilita esecuzione foto.
3	0x00000008	1	Abilita uso del micro di 0 più canale di 0 dell'encoder. (se l'hw dispone di ingresso lettura canale encoder)

Risultato

Rende:

Valore	Significato
> 0	Photo ID.
0	Cattura disabilitata.
< 0	Codice di errore.
-1	Foto hw non eseguibile o errore in fase di start foto o modo continuo non supportato.
-2	Foto già in corso.
-6	Numero di asse illegale.

© 2025 Robox SpA

Sintassi	**i32 pos_capture_cmd (i32 nAx, i32 micro, i32 edge)**
Validità	Rule, Task.
Requisiti di sistema	RTE 33.3.4
Note	

Da utilizzarsi insieme alla funzione pos_capture_sts 524().

È utilizzabile solo con i trasduttori che lo consentono ossia:

- Encoder incrementale
- CAN402 con tipo drive Parker S.B.C. (tipo HID 1) (necessita cot hidrive_ipmode_3)
- CAN402 con tipo drive Parker S.B.C. (tipo SLVD 1) (necessita cot slvdn_ipmode_3)
- EtherCAT con tipo drive Parker S.B.C. (necessita di XML Unico)
- EtherCAT/CANopen con tipo drive Robox IMD20
- CAN402 con tipo drive Sinamics (RCB)
- EtherCAT/CANopen con tipo drive Phase AXM (necessita programma PLC caricato su AXM)
- EtherCAT/CANopen con tipo drive Phase TW (non ancora implementata, non necessita di programma PLC caricato su TW)

Funzione pos_capture_sts()

Funzione di supporto a pos_capture_cmd 520() che serve per testare l'avvenuta cattura della foto.

Sintassi	**i32 pos_capture_sts (i32 nAx, i32 phId, real phDest [, i32 rawPh=NULL])**
nAx	Numero di asse (1-:-32).
phId	Valore reso da pos_capture_cmd 520().
phDest	Variabile in cui la funzione scrive il valore della quota catturata.
rawPh	Variabile in cui la funzione scrive il valore raw della quota. (opzionale, default NULL)
Risultato	Rende:

Valore	Significato
1	Cattura quota effettuata.
0	Sistema pronto per catturare la quota.
< 0	Codice di errore.
-1	Foto disattivata da istruzione o da

© 2025 Robox SpA

Sintassi	**i32 pos_capture_sts (i32 nAx, i32 phId, real phDest [, i32 rawPh=NULL])**

Valore	Significato
	override.
-2	Operazione abortita a causa di fallimento comandi.
-3	Override da ciclo di 0.
-4	Errore durante foto dello 0.
-5	Errore interno.
-6	Numero di asse illegale.
-10	ID non più gestito.

Validità	Rule, Task.
Requisiti di sistema	RTE 33.3.4, RC3E 33.1.9
Note	In caso di foto continua, per ogni foto fatta viene data una sola segnalazione.

Esempi

Esempio di utilizzo:

```
; nella RULE_INIT o nel task prima del LOOP
; abilita la cattura della quota sull'asse su In0 sul fronte di Salita
idFoto = pos_capture_cmd(asse, 1, 0x1)     ; idFoto assumerà un numei
; nella RULE
if(idFoto > 0)
    ; aspetta che la foto sia fatta e ne restituisce il valore nella v
    EsitoFoto = pos_capture_sts(asse, IdFoto, FotoQuota)
    if(EsitoFoto)
        idFoto = pos_capture_cmd(asse, 1, 0x1)     ; riabilita la cattu
        ; elaborazione fotografia di quota
    endif
endif
```

Funzione pow()

Funzione che calcola la potenza di un valore.

Sintassi	**real pow (real base, real exp)**
base	Base (deve essere > 0).

© 2025 Robox SpA

Sintassi	**real pow (real base, real exp)**
exp	Esponente.
Risultato	Rende il risultato di *base* elevato a *exp*.
Validità	Rule, Task.
Note	Se *base* è negativo e *exp* è intero allora esegue la potenza. Se *base* è negativo e *exp* frazionario rende 0 con errore. Se *exp* = 0 la funzione rende 1.0.

Esempi

Esempio di utilizzo:

```
result = pow(2,3)     ; reso = 8
result = pow(2,0)     ; reso = 1
result = pow(3,3)     ; reso = 27
result = pow(10,-2)   ; reso = 0.01
```

Funzione ps_channel_enable()

Funzione per comandare l'abilitazione dei canali del gruppo (PowerSet). Il comando può essere valido per tutti gli assi oppure solo per un sottoinsieme.

NOTA: Il riferimento agli assi logici o fisici dipende da quello definito in configuratore RTE.cfg.

Sintassi	**i32 ps_channel_enable (power_set psName, i32 enaMsk [, i32 axMsk=0xFFFFFFFF])**
psName	Nome del PowerSet (di tipo power_set).
enaMsk	Maschera abilitazione assi:

Bit	Valore esadecimale	Valore bit	Descrizione
0	0x00000001	0	Disabilita primo asse del gruppo.
		1	Abilita primo asse del gruppo.
1	0x00000002	0	Disabilita secondo asse del gruppo.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 ps_channel_enable (power_set psName, i32 enaMsk [, i32 axMsk=0xFFFFFFFF])</td></tr>
<tr><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Valore bit</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>1</td><td>Abilita secondo asse del gruppo.</td></tr>
<tr><td rowspan="2">2 -:- 30</td><td rowspan="2">0x00000004 -:- 0x40000000</td><td>0</td><td>Disabilita n-esimo asse del gruppo.</td></tr>
<tr><td>1</td><td>Abilita n-esimo asse del gruppo.</td></tr>
<tr><td rowspan="2">31</td><td rowspan="2">0x80000000</td><td>0</td><td>Disabilita trentaduesimo asse del gruppo.</td></tr>
<tr><td>1</td><td>Abilita trentaduesimo asse del gruppo.</td></tr>
</table>
NOTA: In caso di assi fisici usare la proprietà pam 926 del PowerSet.
</td></tr>
<tr><td>axMsk</td><td>Maschera assi interessati: (opzionale, default 0xFFFFFFFF)
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Valore bit</th><th>Descrizione</th></tr>
<tr><td rowspan="2">0</td><td rowspan="2">0x00000001</td><td>0</td><td>Non considera primo asse del gruppo.</td></tr>
<tr><td>1</td><td>Gestisci primo asse del gruppo.</td></tr>
<tr><td>1</td><td>0x00000002</td><td>0</td><td>Non considera secondo</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

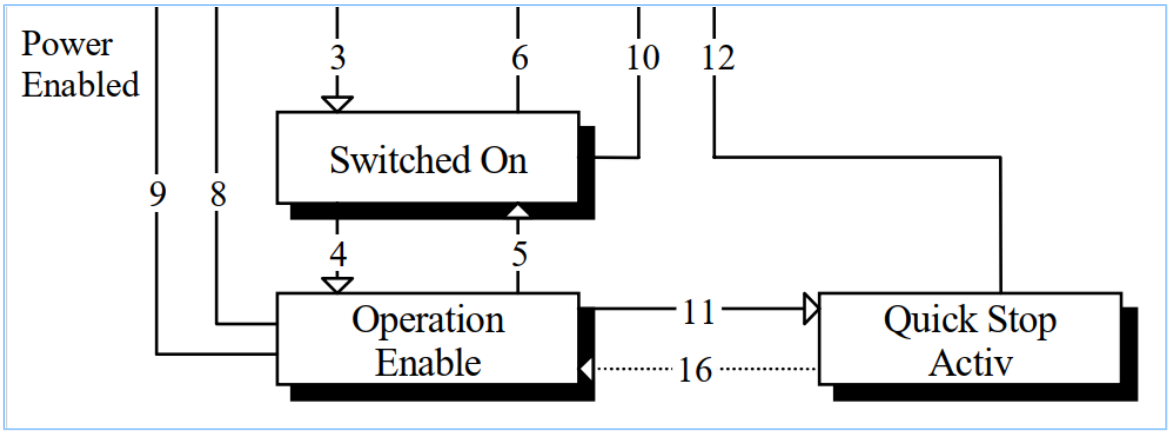
<table>
<tr><td>Sintassi</td><td>i32 ps_channel_enable (power_set psName, i32 enaMsk [, i32 axMsk=0xFFFFFFFF])</td></tr>
<tr><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Valore bit</th><th>Descrizione</th></tr>
<tr><td rowspan="2"></td><td rowspan="2"></td><td></td><td>asse del gruppo.</td></tr>
<tr><td>1</td><td>Gestisci secondo asse del gruppo.</td></tr>
<tr><td rowspan="2">2 -:- 30</td><td rowspan="2">0x00000004 -:- 0x40000000</td><td>0</td><td>Non considera n-esimo asse del gruppo.</td></tr>
<tr><td>1</td><td>Gestisci n-esimo asse del gruppo.</td></tr>
<tr><td rowspan="2">31</td><td rowspan="2">0x80000000</td><td>0</td><td>Non considera trentaduesimo asse del gruppo.</td></tr>
<tr><td>1</td><td>Gestisci trentaduesimo asse del gruppo.</td></tr>
</table>
</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>1</td><td>Operazione riuscita.</td></tr>
<tr><td>0</td><td>Altrimenti.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td>La funzione non ha effetto se il PowerSet dell'asse in questione non è stato precedentemente abilitato con l'istruzione ps_power_enable 540().

La variabile di comando utilizzata da RTE per effettuare il reset è specificata dall'utente nel file RTE.cfg (configuratore di progetto).</td></tr>
</table>

Nel caso di asse gestito in CAN402, viene comandato tramite la control word il passaggio ad operation enable (abilitato) o switched on (disabilitato).

© 2025 Robox SpA

Con riferimento all'immagine seguente, in caso venga abilitato l'asse, la transizione richiesta sarà la numero 4; se invece l'asse viene disabilitato la transizione richiesta sarà la 5.



Il comportamento di un azionamento CAN402, in caso venga richiesta la transizione 5, è determinato dall'oggetto del dizionario CANopen 0x605C "disable operation option code".

Esempi

Esempio di utilizzo:

```
power_set 926 robot1
result =  ps_channel_enable 526 (robot1, 0x4)   ; abilita il terzo asse
result =  ps_channel_enable (robot1, 0xF)   ; abilita i 4 assi del robo
```

Funzione ps_channel_enabled()

Funzione per sapere lo stato delle abilitazioni dei canali del gruppo (PowerSet).

NOTA: Il riferimento agli assi logici o fisici dipende da quello definito in configuratore RTE.cfg.

Sintassi	**i32 ps_channel_enabled (power_set psName)**
psName	Nome del PowerSet (di tipo power_set 926).
Risultato	Maschera assi che sono in stato di enable. Nel caso di asse gestito in CAN402 è la maschera dei drive in OPERATION ENABLE.

Bit	Valore esadecimale	Valore bit	Descrizione
0	0x00000001	0	Primo asse del gruppo disabilitato.
		1	Primo asse del gruppo abilitato.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 ps_channel_enabled (power_set psName)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Valore bit</th><th>Descrizione</th></tr>
<tr><td rowspan="2">1</td><td rowspan="2">0x00000002</td><td>0</td><td>Secondo asse del gruppo disabilitato.</td></tr>
<tr><td>1</td><td>Secondo asse del gruppo abilitato.</td></tr>
<tr><td rowspan="2">2 -:- 30</td><td rowspan="2">0x00000004 -:- 0x40000000</td><td>0</td><td>n-esimo asse del gruppo disabilitato.</td></tr>
<tr><td>1</td><td>n-esimo asse del gruppo abilitato.</td></tr>
<tr><td rowspan="2">31</td><td rowspan="2">0x80000000</td><td>0</td><td>Trentaduesimo asse del gruppo disabilitato.</td></tr>
<tr><td>1</td><td>Trentaduesimo asse del gruppo abilitato.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td></td></tr>
</table>

Esempi

Esempio di utilizzo:

```
power_set 926 robot1
result = ps_channel_enabled 529(robot1)
if(result r_and 0xF)
    sr(1) = "i primi 4 assi sono abilitati!"
```

© 2025 Robox SpA

Funzione ps_channel_fault()

Funzione per sapere lo stato di fault dei canali del gruppo.

NOTA: Il riferimento agli assi logici o fisici dipende da quello definito in configuratore RTE.cfg.

<table>
<tr><td>Sintassi</td><td>i32 ps_channel_fault (power_set psName)</td></tr>
<tr><td>psName</td><td>Nome del PowerSet (di tipo power_set 926).</td></tr>
<tr><td>Risultato</td><td>Maschera assi che sono in stato di fault. Nel caso di asse gestito in CAN402 è la maschera degli assi in stato di fault.
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Valore bit</th><th>Descrizione</th></tr>
<tr><td>0</td><td>0x00000001</td><td>0</td><td>Primo asse del gruppo non in stato di fault.</td></tr>
<tr><td></td><td></td><td>1</td><td>Primo asse del gruppo in fault.</td></tr>
<tr><td>1</td><td>0x00000002</td><td>0</td><td>Secondo asse del gruppo non in stato di fault.</td></tr>
<tr><td></td><td></td><td>1</td><td>Secondo asse del gruppo in fault.</td></tr>
<tr><td>2 -:- 30</td><td>0x00000004 -:- 0x40000000</td><td>0</td><td>n-esimo asse del gruppo non in stato di fault.</td></tr>
<tr><td></td><td></td><td>1</td><td>n-esimo asse del gruppo in fault.</td></tr>
<tr><td>31</td><td>0x80000000</td><td>0</td><td>Trentaduesimo asse del gruppo non in stato di fault.</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 ps_channel_fault (power_set psName)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Valore bit</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>1</td><td>Trentaduesimo asse del gruppo in fault.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td>La variabile di comando utilizzata da RTE per effettuare il reset è specificata dall'utente nel file RTE.cfg (configuratore di progetto).</td></tr>
</table>

Esempi

Esempio di utilizzo:

```
power_set 926 robot1
result = ps_channel_fault 531(robot1)
if(result r_and 0xF)
    sr(1) = "i primi 4 assi sono in fault!"
```

Funzione ps_channel_reset_fault()

Funzione per generare il comando di reset-fault a canali del gruppo (PowerSet). Il comando può essere valido per tutti gli assi oppure solo per un sottoinsieme.

NOTA: Il riferimento agli assi logici o fisici dipende da quello definito in configuratore RTE.cfg.

<table>
<tr><td>Sintassi</td><td>i32 ps_channel_reset_fault (power_set psName, i32 rfMsk [, i32 axMsk=0xFFFFFFFF])</td></tr>
<tr><td>psName</td><td>Nome del PowerSet (di tipo power_set 926).</td></tr>
<tr><td>rfMsk</td><td>Maschera dei comandi di reset fault:
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Valore bit</th><th>Descrizione</th></tr>
<tr><td>0</td><td>0x00000001</td><td>0</td><td>Primo asse nessun comando di reset fault.</td></tr>
<tr><td></td><td></td><td>1</td><td>Reset fault primo asse del gruppo.</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 ps_channel_reset_fault (power_set psName, i32 rfMsk [, i32 axMsk=0xFFFFFFFF])**

Bit	Valore esadecimale	Valore bit	Descrizione
1	0x00000002	0	Secondo asse nessun comando di reset fault.
		1	Reset fault secondo asse del gruppo.
2 -:- 30	0x00000004 -:- 0x40000000	0	n-esimo asse nessun comando di reset fault.
		1	Reset fault n-esimo asse del gruppo.
31	0x80000000	0	Trentaduesimo asse nessun comando di reset fault.
		1	Reset fault trentaduesimo asse del gruppo.

axMsk

Maschera assi interessati: (opzionale, default 0xFFFFFFFF)

Bit	Valore esadecimale	Valore bit	Descrizione
0	0x00000001	0	Non considerare primo asse del gruppo.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 ps_channel_reset_fault (power_set psName, i32 rfMsk [, i32 axMsk=0xFFFFFFFF])</td></tr>
<tr><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Valore bit</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>1</td><td>Gestisci primo asse del gruppo.</td></tr>
<tr><td rowspan="2">1</td><td rowspan="2">0x00000002</td><td>0</td><td>Non considerare secondo asse del gruppo.</td></tr>
<tr><td>1</td><td>Gestisci secondo asse del gruppo.</td></tr>
<tr><td rowspan="2">2 -:- 30</td><td rowspan="2">0x00000004 -:- 0x40000000</td><td>0</td><td>Non considerare n-esimo asse del gruppo.</td></tr>
<tr><td>1</td><td>Gestisci n-esimo asse del gruppo.</td></tr>
<tr><td rowspan="2">31</td><td rowspan="2">0x80000000</td><td>0</td><td>Non considerare trentaduesimo asse del gruppo.</td></tr>
<tr><td>1</td><td>Gestisci trentaduesimo asse del gruppo.</td></tr>
</table>
</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>1</td><td>Operazione riuscita.</td></tr>
<tr><td>0</td><td>Altrimenti.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 ps_channel_reset_fault (power_set psName, i32 rfMsk [, i32 axMsk=0xFFFFFFFF])**
Note	La variabile di comando utilizzata da RTE per effettuare il reset è specificata dall'utente nel file RTE.cfg (configuratore di progetto).

Esempi

Esempio di utilizzo:

```
power_set 926 robot1
result = ps_channel_reset_fault 532(robot1, 0xF)   ; diamo un comando di
```

Funzione ps_drive_reset()

Funzione per generare il comando di drive-reset ad assi appartenenti al gruppo (PowerSet). Il comando può essere valido per tutti gli assi oppure solo per un sottoinsieme.

NOTA: Il riferimento agli assi logici o fisici dipende da quello definito in configuratore RTE.cfg.

<table>
<tr><td>Sintassi</td><td>i32 ps_drive_reset (power_set psName, i32 dfMsk [, i32 axMsk=0xFFFFFFFF])</td></tr>
<tr><td>psName</td><td>Nome del PowerSet (di tipo power_set 926).</td></tr>
<tr><td>dfMsk</td><td>Maschera dei comandi di drive reset:
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Valore bit</th><th>Descrizione</th></tr>
<tr><td>0</td><td>0x00000001</td><td>0</td><td>Primo asse nessun comando di drive reset.</td></tr>
<tr><td></td><td></td><td>1</td><td>Drive reset primo asse del gruppo.</td></tr>
<tr><td>1</td><td>0x00000002</td><td>0</td><td>Secondo asse nessun comando di drive reset.</td></tr>
<tr><td></td><td></td><td>1</td><td>Drive reset secondo asse del gruppo.</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 ps_drive_reset (power_set psName, i32 dfMsk [, i32 axMsk=0xFFFFFFFF])**

Bit	Valore esadecimale	Valore bit	Descrizione
2 -:- 30	0x00000004 -:- 0x40000000	0	n-esimo asse nessun comando di drive reset.
		1	Drive reset n-esimo asse del gruppo.
31	0x80000000	0	Trentaduesimo asse nessun comando di drive reset.
		1	Drive reset trentaduesimo asse del gruppo.

axMsk — Maschera assi interessati: (opzionale, default 0xFFFFFFFF)

Bit	Valore esadecimale	Valore bit	Descrizione
0	0x00000001	0	Non considerare primo asse del gruppo.
		1	Gestisci primo asse del gruppo.
1	0x00000002	0	Non considerare secondo asse del gruppo.
		1	Gestisci secondo

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 ps_drive_reset (power_set psName, i32 dfMsk [, i32 axMsk=0xFFFFFFFF])</td></tr>
<tr><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Valore bit</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td></td><td>asse del gruppo.</td></tr>
<tr><td rowspan="2">2 -:- 30</td><td rowspan="2">0x00000004 -:- 0x40000000</td><td>0</td><td>Non considerare n-esimo asse del gruppo.</td></tr>
<tr><td>1</td><td>Gestisci n-esimo asse del gruppo.</td></tr>
<tr><td rowspan="2">31</td><td rowspan="2">0x80000000</td><td>0</td><td>Non considerare trentaduesimo asse del gruppo.</td></tr>
<tr><td>1</td><td>Gestisci trentaduesimo asse del gruppo.</td></tr>
</table>
</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>1</td><td>Operazione riuscita.</td></tr>
<tr><td>0</td><td>Altrimenti.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td>La variabile di comando utilizzata da RTE per effettuare il reset è specificata dall'utente nel file RTE.cfg (configuratore di progetto).</td></tr>
</table>

Esempi

Esempio di utilizzo:

```
power_set 926 robot1
result = ps_drive_reset 535(robot1, 0x4)   ; diamo un comando di drive re
```

© 2025 Robox SpA

Funzione ps_exclude_lax()

Funzione per escludere un asse logico dal gruppo (PowerSet).

Sintassi	**i32 ps_exclude_lax (power_set psName, i32 nAx)**
psName	Nome del PowerSet (di tipo power_set 926).
nAx	Numero di asse logico.
Risultato	Rende:

Valore	Significato
0	Operazione riuscita.
-1	PowerSet non esistente.
-2	Numero asse illegale.
-3	Asse non presente nel PowerSet specificato.
-4	PowerSer energizzato, operazione abortita.
-5	Funzione non supportata da RTE

Validità	Rule, Task.
Requisiti di sistema	RTE 34.24.9
Note	Le operazioni di esclusione e reinclusione sono permesse se e solo se il PowerSet non è energizzato. Gli assi FB esclusi dal PowerSet verranno mantenuti nello stato di SHUTDOWN ed eventuali loro diagnostiche non influenzeranno il comportamento del PowerSet stesso.

Esempi

Esempio di utilizzo:

```
; in cfgdefs.i3
...
; {7:0} Power set variables
...
power_set 926 sys
power_set ps_qua   ; powerset definition
...

; in task o rule
```

© 2025 Robox SpA

```
int result          ; result of the function
int ax2manage       ; axis number
...
result = ps_exclude_lax(ps_qua, ax2manage)
```

Funzione ps_exclude_pax()

Funzione per escludere un asse fisico dal gruppo (PowerSet).

Sintassi	**i32 ps_exclude_pax (power_set psName, i32 nAx)**
psName	Nome del PowerSet (di tipo power_set).
nAx	Numero di asse fisico.
Risultato	Rende: Valore / Significato 0 / Operazione riuscita. -1 / PowerSet non esistente. -2 / Numero asse illegale. -3 / Asse non presente nel PowerSet specificato. -4 / PowerSer energizzato, operazione abortita. -5 / Funzione non supportata da RTE
Validità	Rule, Task.
Requisiti di sistema	RTE 34.24.9
Note	Le operazioni di esclusione e reinclusione sono permesse se e solo se il PowerSet non è energizzato. Gli assi FB esclusi dal PowerSet verranno mantenuti nello stato di SHUTDOWN ed eventuali loro diagnostiche non influenzeranno il comportamento del PowerSet stesso.

Esempi

Esempio di utilizzo:

```
; in cfgdefs.i3
```

© 2025 Robox SpA

```
...
; {7:0} Power set variables
...
power_set 926 sys
power_set ps_qua   ; powerset definition
...

; in task o rule
int result         ; result of the function
int ax2manage      ; axis number
...
result = ps_exclude_pax 539(ps_qua, ax2manage)
```

Funzione ps_power_enable()

Funzione per abilitare l'energizzazione del gruppo (PowerSet).

Sintassi	**i32 ps_power_enable (power_set psName, i32 flag)**
psName	Nome del PowerSet (di tipo power_set 926).
flag	Flag per abilitare il gruppo: • 0, disabilita energizzazione • 1, abilita energizzazione
Risultato	Rende: Valore: 1 — Significato: Operazione riuscita. Valore: 0 — Significato: Altrimenti.
Validità	Rule, Task.
Note	Per dare il comando di abilitazione all'asse utilizzare l'istruzione ps_channel_enable 526(). La funzione ps_channel_enabled 529() non ha effetto se il PowerSet dell'asse in questione non è stato precedentemente abilitato con l'istruzione ps_power_enable(). La variabile di comando utilizzata da RTE per effettuare il reset è specificata dall'utente nel file RTE.cfg (configuratore di progetto). Nel caso di assi gestiti in CAN402 viene comandata la transizione a SWITCHED ON.

Esempi

Esempio di utilizzo:

© 2025 Robox SpA

```
power_set 926 robot1
if(robot1Present)
    result = ps_power_enable 540(robot1, 1)   ; abilita l'energizzazione
else
    result = ps_power_enable(robot1, 0)   ; disabilita l'energizzazione
endif
```

Funzione ps_reinclude_lax()

Funzione per reincludere un asse logico nel gruppo (PowerSet).

Sintassi	**i32 ps_reinclude_lax (power_set psName, i32 nAx)**
psName	Nome del PowerSet (di tipo power_set 926).
nAx	Numero di asse logico.
Risultato	Rende:

Valore	Significato
0	Operazione riuscita.
-1	PowerSet non esistente.
-2	Numero asse illegale.
-3	Asse non presente nel PowerSet specificato.
-4	PowerSer energizzato, operazione abortita.
-5	Funzione non supportata da RTE
-6	Asse già usato in un altro PowerSet. Viene generato l'allarme 9909.

Validità	Rule, Task.
Requisiti di sistema	RTE 34.24.9
Note	Le operazioni di esclusione e reinclusione sono permesse se e solo se il PowerSet non è energizzato. Gli assi FB esclusi dal PowerSet verranno mantenuti nello stato di SHUTDOWN ed eventuali loro diagnostiche non influenzeranno il comportamento del PowerSet stesso.

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

```
; in cfgdefs.i3
...
; {7:0} Power set variables
...
power_set 926 sys
power_set ps_qua     ; powerset definition
...

; in task o rule
int result           ; result of the function
int ax2manage        ; axis number
...
result = ps_reinclude_lax 541(ps_qua, ax2manage)
...
```

Funzione ps_reinclude_pax()

Funzione per reincludere un asse fisico nel gruppo (PowerSet).

Sintassi	**i32 ps_reinclude_pax (power_set psName, i32 nAx)**
psName	Nome del PowerSet (di tipo power_set 926).
nAx	Numero di asse fisico.
Risultato	Rende:

Valore	Significato
0	Operazione riuscita.
-1	PowerSet non esistente.
-2	Numero asse illegale.
-3	Asse non presente nel PowerSet specificato.
-4	PowerSer energizzato, operazione abortita.
-5	Funzione non supportata da RTE
-6	Asse già usato in un altro PowerSet. Viene generato l'allarme 9909.

© 2025 Robox SpA

Sintassi	**i32 ps_reinclude_pax (power_set psName, i32 nAx)**
Validità	Rule, Task.
Requisiti di sistema	RTE 34.24.9
Note	Le operazioni di esclusione e reinclusione sono permesse se e solo se il PowerSet non è energizzato. Gli assi FB esclusi dal PowerSet verranno mantenuti nello stato di SHUTDOWN ed eventuali loro diagnostiche non influenzeranno il comportamento del PowerSet stesso.

Esempi

Esempio di utilizzo:

```
; in cfgdefs.i3
...
; {7:0} Power set variables
...
power_set 926 sys
power_set ps_qua     ; powerset definition
...

; in task o rule
int result           ; result of the function
int ax2manage        ; axis number
...
result = ps_reinclude_pax 542(ps_qua, ax2manage)
...
```

Funzione ps_status()

Funzione per conoscere lo stato del gruppo (PowerSet).

Sintassi	**i32 ps_status (power_set psName)**
psName	Nome del PowerSet (di tipo power_set 926).
Risultato	Rende: Bit | Valore esadecimale | Descrizione 0 | 0x00000001 | Almeno un drive in fault. 1 | 0x00000002 | In potenza.

© 2025 Robox SpA

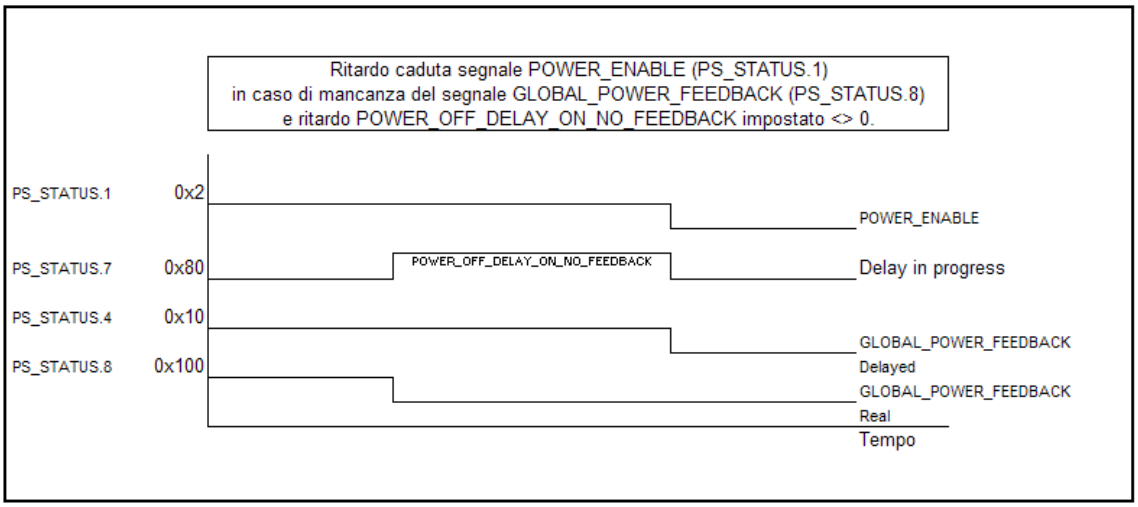
Sintassi	**i32 ps_status (power_set psName)**

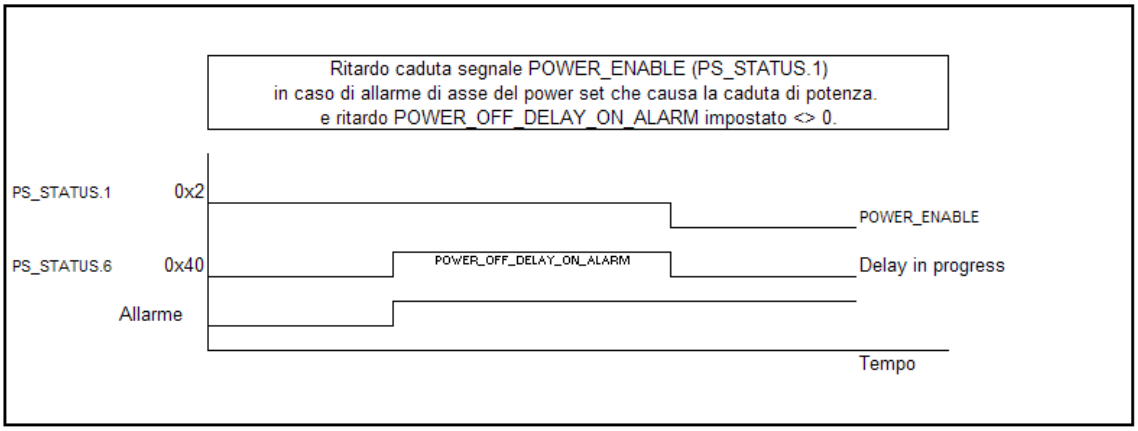
Bit	Valore esadecimale	Descrizione
2	0x00000004	Almeno un drive abilitato.
3	0x00000008	Tutti i drive sono abilitati.
4	0x00000010	Stato (eventualmente) ritardato del feedback.
5	0x00000020	Riservato.
6	0x00000040	Conteggio in corso in caso di delay su allarme di asse del PowerSet che causa la caduta di potenza (power_off_delay_on_alarm).
7	0x00000080	Conteggio in corso in caso di delay su mancanza feedback del PowerSet (power_off_delay_on_no_feedback).
8	0x00000100	Stato effettivo (non ritardato) del feedback.
9	0x00000200	Sono presenti i feedback di tutti i drive del PowerSet.
10	0x00000400	Se la versione RTE <= 34.27.01: sono presenti i feedback di tutti i drive del PowerSet ai quali è stata comandata l'abilitazione (ps_channel_enable 526()).

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 ps_status (power_set psName)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>Se la versione RTE >= 34.27.02: tutti gli assi abilitati hanno il relativo feedback e tutti gli assi abilitati hanno il bit di "drive follows command value", bit 12 di fb_stsw.</td></tr>
<tr><td>11</td><td>0x00000800</td><td>Sono presenti assi esclusi.</td></tr>
<tr><td>12</td><td>0x00001000</td><td>PowerSet deenergizzato per mancanza del feedback globale. (da RTE 34.26.0)</td></tr>
<tr><td>13</td><td>0x00002000</td><td>Tutti gli assi abilitati hanno il relativo feedback.</td></tr>
<tr><td>14</td><td>0x00004000</td><td>Tutti gli assi abilitati hanno il relativo bit di "drive follows command value", bit 12 di fb_stsw.</td></tr>
<tr><td>15 -:- 32</td><td>0x00008000 -:- 0x80000000</td><td>Riservati.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td>RTE, per diagnosticare se il drive è in fault, testa il segnale specificato nella keyword ps_channel_fault 531().</td></tr>
</table>

© 2025 Robox SpA





Esempi

Esempio di utilizzo:

```
power_set 926 robot1
int result
result =  ps_status 543 (robot1)
if(result r_and 806 0x8)
    sr(1) = "situazione ottima"
endif

if(result.6 or result.7)
    ; gestisco la frenata e l'attivazione dei freni
endif
```

© 2025 Robox SpA

Funzione qwbe2lef()

Converte big endian qword (x64) in little endian real (x64).

Sintassi	**real qwbe2lef (stru_large_int| stru_large_uint stru)**
stru	Struttura d'ingresso di tipo stru_large_int[872] o di tipo stru_large_uint[873].
Risultato	Valore di ritorno.
Validità	Rule, Task.
Requisiti di sistema	RTE 34.24.10
Note	

Esempi

Esempio di utilizzo:

```
stru_large_uint[873] input
real result

input.low  = 0xfb210940 ; Data memory: 40 09 21 fb hex.
input.high = 0x182d4454 ; Data memory: 54 44 2d 18 hex.
result = qwbe2lef[547](input)
; result = 3.141592653589793116 -> data memory: 18 2d 44 54 fb 21 09 4
```

Funzione qwbedwx2lef()

Converte dword exchange big endian qword (x64) in little endian real (x64).

Sintassi	**real qwbedwx2lef (stru_large_int| stru_large_uint stru)**
stru	Struttura d'ingresso di tipo stru_large_int[872] o di tipo stru_large_uint[873].
Risultato	Valore di ritorno.
Validità	Rule, Task.
Requisiti di sistema	RTE 34.24.10
Note	

Esempi

Esempio di utilizzo:

```
stru_large_uint[873] input
real result

input.low  = 0x182d4454 ; Data memory: 54 44 2d 18 hex.
```

© 2025 Robox SpA

```
input.high = 0xfb210940 ; Data memory: 40 09 21 fb hex.
result = qwbedwx2lef(input)
; result = 3.141592653589793116 -> data memory: 18 2d 44 54 fb 21 09 4
```

Funzione qwmbe2lef()

Converte middle big endian qword (x64) in little endian real (x64).

Sintassi	**real qwmbe2lef (stru_large_int| stru_large_uint stru)**
stru	Struttura d'ingresso di tipo stru_large_int o di tipo stru_large_uint.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Requisiti di sistema	RTE 34.24.10
Note	

Esempi

Esempio di utilizzo:

```
stru_large_uint input
real result

input.low  = 0x0940fb21 ; Data memory: 21 fb 40 09 hex.
input.high = 0x4454182d ; Data memory: 2d 18 54 44 hex.
result = qwmbe2lef(input)
; result = 3.141592653589793116 -> data memory: 18 2d 44 54 fb 21 09 4
```

Funzione qwmbedwx2lef()

Converte dword exchange middle big endian qword (x64) in little endian real (x64).

Sintassi	**real qwmbedwx2lef (stru_large_int| stru_large_uint stru)**
stru	Struttura d'ingresso di tipo stru_large_int o di tipo stru_large_uint.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Requisiti di sistema	RTE 34.24.10
Note	

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
stru_large_uint[873] input
real result

input.low  = 0x4454182d ; Data memory: 2d 18 54 44 hex.
input.high = 0x0940fb21 ; Data memory: 21 fb 40 09 hex.
result = qwmbedwx2lef[548](input)
; result = 3.141592653589793116 -> data memory: 18 2d 44 54 fb 21 09 4
```

Funzione r_rol()

Funzione che effettua una rotazione a sinistra del numero di bit specificato.

Sintassi	**i32 r_rol (i32 value, i32 nBit)**
value	Grandezza sulla quale eseguire la rotazione.
nBit	Numero di bit.
Risultato	Rende il risultato della rotazione.
Validità	Rule, Task.
Note	Nella rotazione il bit più pesante uscente a sinistra viene ricircolato nel bit più leggero entrante a destra.

Esempi

Esempio di utilizzo:

```
r(100) = r_rol[549](r(101), 3)
; Assumendo un valore di r(101) = 0x12481248, dopo l'esecuzione della
```

NOTA: Siccome il bit più pesante non esce a sinistra il comportamento è analogo alla r_shl[551]().

Esempio di utilizzo:

```
r(100) = r_rol(r(101), 4)
; Assumendo un valore di r(101) = 0x12481248, dopo l'esecuzione dell'i
```

NOTA: Il bit più pesante esce a sinistra e viene ricircolato a destra (comportamento diverso da r_shl()).

Esempio di utilizzo:

```
r(1) = r_rol(r(1), 1)
```

© 2025 Robox SpA

# esecuzione								
Valore iniziale	0000	0000	0000	0000	0000	0000	0000	000**1**
1	0000	0000	0000	0000	0000	0000	0000	00**1**0
2	0000	0000	0000	0000	0000	0000	0000	0**1**00
...								
31	**1**000	0000	0000	0000	0000	0000	0000	0000
32	0000	0000	0000	0000	0000	0000	0000	000**1**

Funzione r_ror()

Funzione che effettua una rotazione a destra del numero di bit specificato.

Sintassi	**i32 r_ror (i32 value, i32 nBit)**
value	Grandezza sulla quale eseguire la rotazione.
nBit	Numero di bit.
Risultato	Rende il risultato della rotazione.
Validità	Rule, Task.
Note	Nella rotazione il bit più leggero uscente a destra viene ricircolato nel bit più pesante entrante a sinistra.

Esempi

Esempio di utilizzo:

```
r(100) = r_ror[550](r(101), 3)
; Assumendo un valore di r(101) = 0x12481248, dopo l'esecuzione della
```

NOTA: Siccome il bit più leggero non esce a destra il comportamento è analogo alla r_shr[552]().

Esempio di utilizzo:

```
r(100) = r_ror(r(101), 4)
; Assumendo un valore di r(101) = 0x12481248, dopo l'esecuzione dell'i
```

NOTA: Il bit più leggero esce a destra e viene ricircolato a sinistra (comportamento diverso da r_shr()).

Esempio di utilizzo:

```
r(1) = r_ror(r(1), 1)
```

© 2025 Robox SpA

# esecuzione								
Valore iniziale	0000	0000	0000	0000	0000	0000	0000	000**1**
1	**1**000	0000	0000	0000	0000	0000	0000	0000
2	0**1**00	0000	0000	0000	0000	0000	0000	0000
...								
31	0000	0000	0000	0000	0000	0000	0000	00**1**0
32	0000	0000	0000	0000	0000	0000	0000	000**1**

Funzione r_shl()

Funzione che effettua uno shift aritmetico a sinistra del numero di bit specificato.

Sintassi	**i32 r_shl (i32 value, i32 nBit)**
value	Grandezza sulla quale eseguire lo shift.
nBit	Numero di bit.
Risultato	Rende il risultato dello shift.
Validità	Rule, Task.
Note	Nello shift a sinistra il bit più pesante uscente a sinistra viene perso. A destra entra uno 0.

Esempi

Esempio di utilizzo:

```
r(100) = r_shl(r(101), 3)
; Assumendo un valore di r(101) = 0x12481248, dopo l'esecuzione della
```

NOTA: Siccome il bit più pesante non esce a sinistra il comportamento è analogo alla r_rol().

Esempio di utilizzo:

```
r(100) = r_rshl(r(101), 4)
; Assumendo un valore di r(101) = 0x12481248, dopo l'esecuzione dell'i
```

NOTA: Il bit più pesante esce a sinistra e viene perso. A destra entra uno 0 (comportamento diverso da r_rol()).

Esempio di utilizzo:

© 2025 Robox SpA

```
r(1) = r_shl(r(1), 1)
```

# esecuzione								
Valore iniziale	0000	0000	0000	0000	0000	0000	0000	000**1**
1	0000	0000	0000	0000	0000	0000	0000	00**1**0
2	0000	0000	0000	0000	0000	0000	0000	0**1**00
...								
31	**1**000	0000	0000	0000	0000	0000	0000	0000
32	0000	0000	0000	0000	0000	0000	0000	0000

Funzione r_shr()

Funzione che effettua uno shift aritmetico a destra del numero di bit specificato.

Sintassi	**i32 r_shr (i32 value, i32 nBit)**
value	Grandezza sulla quale eseguire lo shift.
nBit	Numero di bit.
Risultato	Rende il risultato dello shift.
Validità	Rule, Task.
Note	Nello shift a destra il bit più leggero uscente a destra viene perso. Nel bit più pesante entra uno 0 se era in 0 entra un 1 se era in 1 (viene mantenuto il segno).

Esempi

Esempio di utilizzo:

```
r(100) = r_shr(r(101), 3)
; Assumendo un valore di r(101) = 0x12481248, dopo l'esecuzione della
```

NOTA: Siccome il bit più leggero non esce a destra il comportamento è analogo alla r_ror().

Esempio di utilizzo:

```
r(100) = r_rshr(r(101), 4)
; Assumendo un valore di r(101) = 0x12481248, dopo l'esecuzione dell'i
```

Esempio di utilizzo:

```
r(1) = r_shl(r(1), 1)
```

© 2025 Robox SpA

# esecuzione								
Valore iniziale	**1**000	0000	0000	0000	0000	0000	0000	000**1**
1	**11**00	0000	0000	0000	0100	0000	0000	0000
2	**111**0	0000	0000	0000	0010	0000	0000	0000
...								
31	**1111**	**1111**	**1111**	**1111**	**1111**	**1111**	**1111**	**111**0
32	**1111**	**1111**	**1111**	**1111**	**1111**	**1111**	**1111**	**1111**

Funzione r2m16()

Funzione utilizzata per manipolare un valore reale, restituendo a sua volta un valore reale, ma modificato in modo da riportarlo nel campo di valori validi per una grandezza di tipo unsigned int a 16 bit (equivalente ad assegnare a un unsigned int 16 il dato entrante e quindi copiare questo dato in un'altra variabile reale).

Sintassi	**real r2m16 (real value)**
value	È una generica espressione di tipo reale che fornisce alla funzione il dato da manipolare.
Risultato	Rende il valore convertito.
Validità	Rule, Task.
Note	

Le regole di conversione del dato da reale ad unsigned 16 comportano i seguenti risultati:

Valore entrante	Valore reso
0	0
17	17
65535	65535
65536	0
65537	1
-1	65535
-17	65519
-65535	1
-65536	0

© 2025 Robox SpA

Valore entrante	Valore reso
-65537	65535

Funzione r2m32()

Funzione utilizzata per manipolare un valore reale (double 8 byte), restituendo ancora un valore reale, ma modificato in modo da riportarlo nel campo di valori validi per una grandezza di tipo unsigned int a 32 bit (equivalente ad assegnare a un unsigned int 32 il dato entrante e quindi copiare questo dato in un'altra variabile reale).

Sintassi	**real r2m32 (real value)**
value	È una generica espressione di tipo reale che fornisce alla funzione il dato da manipolare.
Risultato	Rende il valore convertito.
Validità	Rule, Task.
Note	

Esegue il modulo di *value* con il valore 2^32 (4294967296):

- Se *value* è positivo, la funzione restituisce il suo modulo con il valore 2^32
- Se *value* è negativo, la funzione esegue il modulo con il valore 2^32 del suo valore assoluto. Successivamente ne fa il complemento a 2

Le regole di conversione del dato da reale a unsigned 32 comportano i seguenti risultati:

Valore entrante	Valore reso
0	0
17	17
4294967295	4294967295
4294967296	0
4294967297	1
-1	4294967295
-17	4294967279
-4294967295	1
-4294967296	0
-4294967297	4294967295

Funzione r2m8()

Funzione utilizzata per manipolare un valore reale, restituendo ancora un valore reale, ma modificato in modo da riportarlo nel campo di valori validi per una grandezza di tipo unsigned int a 8 bit (equivalente ad assegnare a un unsigned int 8 il dato entrante e quindi copiare questo dato in un'altra variabile reale).

© 2025 Robox SpA

Sintassi	**real r2m8 (real value)**
value	È una generica espressione di tipo reale che fornisce alla funzione il dato da manipolare.
Risultato	Rende il valore convertito.
Validità	Rule, Task.
Note	

Le regole di conversione del dato da reale ad unsigned 8 comportano i seguenti risultati:

Valore entrante	Valore reso
0	0
17	17
255	255
256	0
257	1
258	2
-1	255
-17	239
-255	1
-256	0
-257	255
-258	254

Funzione ramp()

Funzione che permette di incrementare linearmente (costruisce una rampa) una variabile fino al valore desiderato.

Per esempio, può essere utile per portare la velocità ideale di un asse a 0 con decelerazione costante.

Sintassi	**real ramp (real value, real target, real acc [, real dec=acc])**
value	Variabile da incrementare (può essere un registro, una variabile locale, una grandezza cinematica, ecc..).
target	Valore target desiderato per la variabile. La funzione si fermerà solo quando *value* sarà uguale a *target*.

© 2025 Robox SpA

Sintassi	**real ramp (real value, real target, real acc [, real dec=acc])**
acc	Valore di accelerazione per raggiungere il *target*.
dec	Valore di decelerazione per raggiungere il *target*. (opzionale, default *acc*)
Risultato	Rende il valore aggiornato della variabile entrante.
Validità	Rule.
Note	Accelerazione deve intendersi come valore per allontanarsi dallo 0. Decelerazione deve intendersi come valore per portarsi allo 0.

Esempi

Esempio di utilizzo:

```
rr(3)= ramp555(rr(3), 1000, 100)
; portiamo la variabile rr(3) al valore 1000 con acc e dec pari a 100.
; ricorrere alla relazione deltaVel/acc per sapere quanto tempo è impi

rr(2) = ramp(rr(2), 1000, 100)
; se al momento della chiamata rr(2)=0 avremo un tempo di accelerazion

iv(AX_1) = ramp(iv(AX_1), rr(1), nvrr(1))
; portiamo l'asse 1 alla velocità indicata in rr(1) con
acc/decelerazione indicate in nvrr(1)
```

Funzione rand32()

Funzione che rende un numero intero casuale compreso fra 0 e MAX_INT32 (0x7fffffff = +2147483647).

Sintassi	**i32 rand32 ()**
Risultato	Rende il numero casuale.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
r(3000) = rand32556()
```

© 2025 Robox SpA

Funzione random()

Funzione che rende un numero intero casuale compreso fra 0 e MAX_INT16 (0x7fff).

Sintassi	**i16 random ()**
Risultato	Rende il numero casuale.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
midVal = midVal + error * random()/max_int16
```

Funzione random_value()

Funzione che rende un numero reale casuale compreso tra un valore minimo e un valore massimo dati.

Sintassi	**real random_value (real min, real max)**
min	Valore limite inferiore.
max	Valore limite superiore.
Risultato	Rende il numero casuale.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
r(3000) = random_value(-100, 100)
```

Funzione range()

Funzione che testa se il valore della variabile argomento è all'interno dei limiti desiderati.

Sintassi	**i32 range (real value, real min, real max)**
value	È la grandezza da valutare (può essere un'espressione generica).
min	È il valore di limite inferiore (può essere un'espressione generica).

© 2025 Robox SpA

Sintassi	**i32 range (real value, real min, real max)**
max	È il valore di limite superiore (può essere un'espressione generica).
Risultato	Rende: Valore: 1 — Significato: Valore all'interno del range specificato. Valore: 0 — Significato: Valore esterno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
if(not range(rr(5), -12.5, +12.5))
    sr(1) = "il valore assegnato ad rr(5) non è all'interno dal range
endif

if(not range(pluto, -12.5, +12.5))
    pluto = 12.5 * sign(pluto)
endif
; Assumendo pluto = - 30 dopo l' esecuzione delle istruzioni pluto sar
```

Funzione range_adj()

Funzione che rende una scalatura del parametro d'ingresso. Normalizza una variabile che sta in un intervallo dato in ingresso portandola in un intervallo dato in uscita.

Sintassi	**real range_adj (real inValue, real inLow, real inHigh, real outLow, real outHigh)**
inValue	Valore d'ingresso.
inLow	Limite inferiore della variabile d'ingresso.
inHigh	Limite superiore della variabile d'ingresso.
outLow	Limite inferiore della variabile normalizzata.
outHigh	Limite superiore della variabile normalizzata.
Risultato	Rende il valore normalizzato calcolato come: ((*inValue* - *inLow*) / (*inHigh* - *inLow*)) * (*outHigh* - *outLow*) + *outLow*
Validità	Rule, Task.
Requisiti di sistema	RC3E 33.9.0.

© 2025 Robox SpA

Sintassi	**real range_adj (real inValue, real inLow, real inHigh, real outLow, real outHigh)**
Note	Nel caso in cui *inHigh - inLow* sia uguale a 0.0, la funzione restituisce *outLow*.

Esempi

Esempio di utilizzo:

```
lit nvrrMinSSI_bit nvrr(1000)
lit nvrrMaxSSI_bit nvrr(1001)
lit nvrrMaxSSI_mm  nvrr(1002)
lit nvrrMinSSI_mm  nvrr(1003)
lit rrRawSSI       rr(500)
lit rrNormField    rr(501)
...
nvrrMinSSI_bit = 3000
nvrrMaxSSI_bit = 6000
nvrrMinSSI_mm  = 500
nvrrMinSSI_mm  = 1000
...
rrRawSSI = 4500
rrNormField = range_adj(rrRawSSI, nvrrMinSSI_bit, &
                        nvrrMaxSSI_bit, nvrrMinSSI_mm, &
                        nvrrMinSSI_mm)
; ((4500-3000)/(6000-3000))*(1000-500)+500
; (1500/3000)*500+500
; 250+500
; rrNormField = 750
```

Funzione rbit()

Funzione che permette di resettare uno o più bit in un dato di tipo intero in funzione del verificarsi di una condizione.

Sintassi	**i32 rbit (i32 data, i32 bitMsk, bool cond)**
data	Grandezza sulla quale si desidera effettuare il reset.
bitMsk	Maschera dei bit da resettare.
cond	Condizione al verificarsi della quale i bit verranno resettati.
Risultato	Rende: Valore: 1 — La condizione è verificata. Valore: 0 — Altrimenti.

© 2025 Robox SpA

Sintassi	**i32 rbit (i32 data, i32 bitMsk, bool cond)**
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
rbit(r(1), 0x000000ff, r(100) = 34)
; mette a 0 gli 8 bit bassi quando r(100) è uguale a 34
```

Funzione release()

Funzione per rilasciare la forzatura dello stato degli I/O.

Sintassi	**i32 release (what)**
what	Oggetto da forzare. Può essere:

Tipo	Descrizione
inp(nCh)	Dove *nCh* è il canale input da rilasciare. Può essere un'espressione qualsiasi.
inp_w(nW)	Dove *nW* è l'indice della input word da rilasciare. Può essere un'espressione qualsiasi.
inp_w(nW).(iB)	Dove *nW* è l'indice della input word e *iB* è il bit della input word da rilasciare. Può essere un'espressione qualsiasi.
out(nCh)	Dove *nCh* è il canale output da rilasciare. Può essere un'espressione qualsiasi.
out_w(nW)	Dove *nW* è l'indice della output word da rilasciare. Può essere un'espressione qualsiasi.
out_w(nW).(iB)	Dove *nW* è l'indice della output word e *iB*

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 release (what)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td>è il bit della output word da rilasciare. Può essere un'espressione qualsiasi.</td></tr>
</table>
</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Descrizione</th></tr>
<tr><td>0</td><td>Operazione eseguita con successo.</td></tr>
<tr><td>-1</td><td>Versione di RTE non compatibile.</td></tr>
<tr><td>-2</td><td>Indice non valido.</td></tr>
<tr><td>-3</td><td>Bit non valito.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td></td></tr>
</table>

Esempi

Esempio di utilizzo:

```
$LIT iw_rem  401                ; indice input word remota
...
...
$LIT pbStart  inp_w(iw_rem).(4) ; start pushbutton
...
...
esitoF = release(pbStart, 0)
if(esitoF < 0)
    ...                  ; diagnostica per errore..
endif
```

Esempio di utilizzo:

```
$LIT iw_base_rem  401                    ; indice input word remota
...
...
$LIT pbStop  inp((iw_base_rem-1)*16+5)   ; stop pushbuttom
...
...
esitoF = release(pbStop)
if(esitoF < 0)
    ...                  ; diagnostica per errore..
endif
```

© 2025 Robox SpA

Esempio di utilizzo:

```
; rilascia un output
; r(1000) contiene il canale output
esitoF = release(out(r(1000)))
if(esitoF < 0)
    ...                 ; diagnostica per errore..
endif
```

Funzione rise()

Funzione che permette di cogliere il fronte di salita di una condizione.

Sintassi	**bool rise (cond)**
cond	Condizione di cui si vuole cogliere il fronte di salita.
Risultato	Rende TRUE solo quando la condizione è verificata ed era FALSE all'esecuzione precedente.
Validità	Rule, Task.
Note	La prima volta che viene eseguita, la funzione rende FALSE e memorizza lo stato attuale della condizione. Nel caso si utilizzi rise() in una funzione R3 utente, va evitato che i parametri della funzione stessa siano coinvolti nell'espressione di condizione. Il risultato non sarebbe quello sperato a causa della sovrapposizione in memoria dello stato di diversi variabili.

Ogni funzione rise() ha la propria memoria dello stato della condizione.

La valutazione del fronte è fatta tra lo stato attuale della condizione e la memoria relativa alla singola chiamata della funzione (es. se abbiamo allo step 1000 `if(rise(inp(1)))` e allo step 1100 `if(rise(inp(1)))`, benché facciano riferimento alla stessa condizione, la valutazione è fatta in modo distinto, ognuna in base allo stato memorizzato eseguendo step 1000 e poi step 1100).

Esempi

Esempio di utilizzo:

```
if(rise(inp(449)))
    sr(1) = "fronte di salita"
endif

if(rise(rr(1)>12))
    sr(1) = "sopra la soglia"
endif
```

© 2025 Robox SpA

Funzione rise_filt()

Funzione che permette di cogliere il fronte di salita di una condizione impostando un filtro in spazio o tempo.

Funzionamento

Al rilevamento del fronte di salita della condizione viene memorizzato il valore della variabile di filtro corrispondente.

Finché la differenza tra il valore attuale della variabile di filtro e quello memorizzato è inferiore alla soglia specificata la funzione rende 0, mentre quando la differenza diventa maggiore della soglia specificata la funzione rende il valore memorizzato sul fronte di salita.

<table>
<tr><td>Sintassi</td><td>real rise_filt (bool cond, real varFilt, real thrFilt)</td></tr>
<tr><td>cond</td><td>Condizione da valutare per la ricerca del fronte.</td></tr>
<tr><td>varFilt</td><td>Grandezza evolvente (tipicamente tempo (tfb) o spazio ip(n)).</td></tr>
<tr><td>thrFilt</td><td>Quantità di varFilt in cui si deve vedere stabilmente la condizione vera.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><td>Valore</td><td>Condizione</td></tr>
<tr><td rowspan="2">0</td><td>cond = FALSE</td></tr>
<tr><td>cond = TRUE ma non si è superato la soglia thrFilt</td></tr>
<tr><td><> 0 (viene reso il valore di varFilt registrato sul fronte di salita di cond)</td><td>cond = TRUE e si è superato thrFilt (valore della variabile di riferimento memorizzato su transizione).</td></tr>
</table></td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td>Dalla prima volta che viene eseguita la funzione fino a che la condizione si mantiene alta e non si supera la soglia il valore reso è 0. Superata tale soglia il valore reso è il primo valore fornito di varFilt.
Nel caso si utilizzi rise_filt() in una funzione R3 utente, va evitato che i parametri della funzione stessa siano coinvolti nell'espressione di condizione. Il risultato non sarebbe quello sperato a causa della sovrapposizione in memoria dello stato di diversi variabili.</td></tr>
</table>

© 2025 Robox SpA

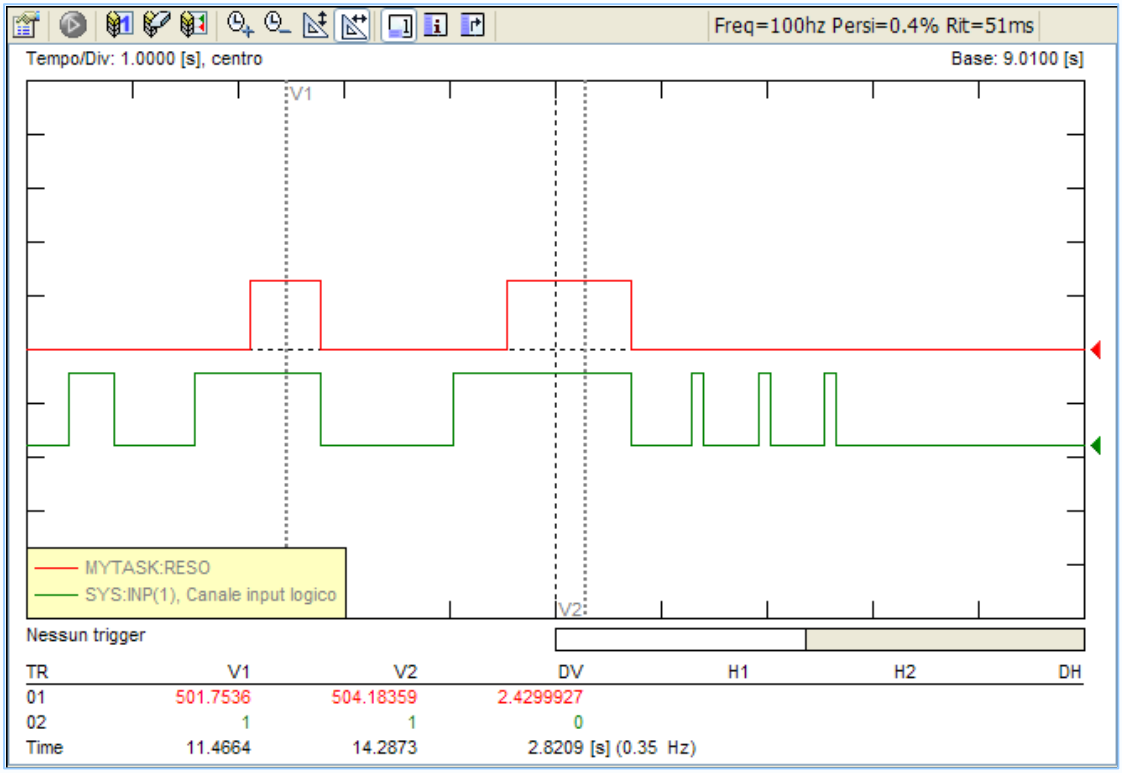
Fornendo come variabile di filtro ad esempio ip(n) si avrà un filtro in spazio, mentre fornendo come variabile di filtro tfb si avrà un filtro in tempo (unità secondi).

Ogni funzione di rise_filt() ha la propria memoria dello stato della condizione. La valutazione del fronte è fatta tra lo stato attuale della condizione e la memoria relativa alla singola chiamata della funzione. (es. se abbiamo allo step 1000 `if(rise_filt(inp(1), tfb, 0.5))` e allo step 1100 `if(rise_filt(inp(1), tfb, 0.5))`, benché facciano riferimento alla stessa condizione, la valutazione è fatta in modo distinto, ognuna in base allo stato memorizzato eseguendo step 1000 e poi step 1100).

Esempi

Esempio di utilizzo:

```
reso = rise_filt(563)(inp(1), tfb, 0.5)
```




```
; filtro temporale su inp(1)
inp1f = not(fall_filt(339)(rise_filt(inp(1), tfb, TimeFilter), tfb, TimeF

; calcolo della distanza tra la salita dell'input e la posizione attua
; per gestire una uscita tra due soglie
posUp = rise_filt(inp(1), ip(1), spaceFilter)
if(posUp)
    ; siamo in alto con input. vediamo lo spazio percorso
    deltaPos = abs(156)(ip(1) - posUp)
    ; apri la botola per eliminare pezzo
    out(oApriBotola) = (deltaPos > MaxDimensionItem)
else
```

© 2025 Robox SpA

```
    out(oApriBotola) = false
endif
```

Funzione round()

Funzione che rende il numero intero più vicino al numero indicato.

Sintassi	**i32 round (real value)**
value	Valore in ingresso (può essere negativo).
Risultato	Numero intero restituito.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
reso = round(12.3)     ; reso = 12
reso = round(12.499)   ; reso = 12
reso = round(12.5)     ; reso = 13
reso = round(12.9)     ; reso = 13
reso = round(-12.3)    ; reso = -12
reso = round(-12.499)  ; reso = -12
reso = round(-12.5)    ; reso = -12
reso = round(-12.501)  ; reso = -13
```

Funzione rtc_date()

Funzione che permette di settare la data del real time clock/calender presente sui controlli asse collegati.

Sintassi	**i32 rtc_date (i32 d, i32 m, i32 y)**
d	Variabile contenente il giorno (1-:-31).
m	Variabile contenente il mese (1-:-12).
y	Variabile contenente l'anno (2007-:-2099).
Risultato	Rende:

Valore	Significato
0	Data settata con successo.
-1	Giorno errato.
-2	Mese errato.

© 2025 Robox SpA

Sintassi	**i32 rtc_date (i32 d, i32 m, i32 y)**
	Valore: -3 — Significato: Anno errato.
Validità	Rule, Task.
Requisiti di sistema	RTE 33.8.1
Note	Il giorno, il mese, l'anno, ..., attuali del clock/calender si possono ottenere tramite le variabili predefinite specifiche come ad esempio day, month, year, ... La data si può visualizzare tramite comando shell date.

Esempi

Esempio di utilizzo:

```
int reso
r(1)=20
r(2)=10
r(3)=2020
if(setDate)
    setdate = 0
    reso = rtc_date(r(1), r(2), r(3))   ; imposto data 20 ottobre 2020
endif
```

Funzione rtc_time()

Funzione che permette di settare l'orario del real time clock/calender presente sui controlli asse collegati.

Sintassi	**i32 rtc_time (i32 h, i32 m, i32 s)**
h	Variabile contenente l'ora (0-:-23).
m	Variabile contenente i minuti (0-:-59).
s	Variabile contenente i secondi (0-:-59).
Risultato	Rende: Valore: 0 — Significato: Orario settato con successo. Valore: -1 — Significato: Ora errata. Valore: -2 — Significato: Minuti errati. Valore: -3 — Secondi errati.
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**i32 rtc_time (i32 h, i32 m, i32 s)**
Requisiti di sistema	RTE 33.8.1
Note	L'ora, i minuti, i secondi, ..., attuali del clock/calender si possono ottenere tramite le variabili predefinite specifiche come ad esempio hour, minute, second, ... L'ora si può visualizzare tramite comando shell time.

Esempi

Esempio di utilizzo:

```
int reso
r(1)=20
r(2)=10
r(3)=59
if(setTime)
    setTime = 0
    reso = rtc_time(r(1), r(2), r(3))   ; imposto ora 20:10:59
endif
```

Funzione safe_appl_ack()

Funzione che permette di settare il valore dell'Application Acknowledge di un'isola safety.

Sintassi	**i32 safe_appl_ack (i32 nIsle, i32 value [, i32 mask=0xFFFFFFFF])**
nIsle	Numero dell'isola (1-:-31).
value	Valore da settare.
mask	Maschera di *value.* (opzionale, default 0xFFFFFFFF)
Risultato	Rende: Valore / Significato 0 / Risultato positivo. -1 / Parametri illegali. -2 / *nIsle* illegale. -4 / Isola inesistente.
Validità	Rule, Task.
Requisiti di sistema	RTE 34.20.1
Note	

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
int esito
; imposto alto il secondo Appl. Ack dell'isola 3
esito = safe_appl_ack(3, 0x00000010)
if(esito = 0)
; settaggio dell'Appl. Ack riuscito
endif
```

Funzione safe_appl_diag()

Funzione che ritorna lo stato dei bit degli Application Diagnostic di un'isola safety.

Sintassi	**i32 safe_appl_diag (i32 nIsle [, i32 what=0])**
nIsle	Numero dell'isola (1-:-31).
what	Parametro per selezionare il gruppo di bit. (opzionale, default 0) Valore: 0 — Significato: Ricevo tutti e 32 i bit. Valore: 1 — Significato: Ricevo i primi 16 bit. Valore: 2 — Significato: Ricevo gli ultimi 16 bit.
Risultato	Rende: Condizione: *what* = 0 — Valore: > 0, Application Diagnostic (bit 0 -:- 31). — Significato: Risultato positivo. Condizione: *what* = 0 — Valore: 0 — Significato: Risultato negativo. Condizione: *what* = 1 — Valore: > 0, Application Diagnostic (bit 0 -:- 15). — Significato: Risultato positivo. Condizione: *what* = 1 — Valore: -1 — Significato: Parametri illegali. Condizione: *what* = 1 — Valore: -2 — Significato: *nIsle* illegale. Condizione: *what* = 1 — Valore: -4 — Significato: Isola inesistente. Condizione: *what* = 2 — Valore: > 0, Application — Significato: Risultato positivo.

© 2025 Robox SpA

Sintassi	**i32 safe_appl_diag (i32 nIsle [, i32 what=0])**

Condizione	Valore	Significato
	Diagnostic (bit 16 -:- 31).	
	-1	Parametri illegali.
	-2	*nIsle* illegale.
	-4	Isola inesistente.

Validità	Rule, Task.
Requisiti di sistema	RTE 34.20.1
Note	

Esempi

Esempio di utilizzo:

```
; Supponiamo che lo stato degli Application Diagnostic dell'isola 1 si
int esito
esito = safe_appl_diag(1)
; esito = 0xaaaabbbb
esito = safe_appl_diag(1, 1)
; esito = 0x0000aaaa
esito = safe_appl_diag(1, 2)
; esito = 0x0000bbbb
```

Funzione safe_diag()

Funzione che ritorna il registro di diagnostica del satellite desiderato.

Sintassi	**i32 safe_diag (i32 nIsle, i32 nSat)**
nIsle	Numero dell'isola (1-:-31).
nSat	Numero del satellite (0-:-16).
Risultato	Rende:

Valore	Modulo	Bit	Valore Hex	Significato
> 0	LPSDO8	16 -:- 31	0xFFFF0000	[Z] Zero.
		15	0x8000	[COK] Comunic

© 2025 Robox SpA

Sintassi	**i32 safe_diag (i32 nIsle, i32 nSat)**

Valore	Modulo	Bit	Valore Hex	Significato
				azione OK.
		14	0x4000	[SA] Indirizzo Safety scritto nei bit 0-:-10.
		13	0x2000	[E] Dispositivo in Errore (override bit 14) codice di errore scritto nei bit 0-:-10. (vedi documentazione "Phoenix Contact 106889 - Axioline F module with integrated safety logic and safe digital outputs")
		12	0x1000	[PUR] Richiesta Power Up.
		11	0x0800	[OAR] Richiesta Operator Acknowledge.
		0 -:- 10	0x07FF	Indirizzo safety o codice di errore.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 safe_diag (i32 nIsle, i32 nSat)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Modulo</th><th>Bit</th><th>Valore Hex</th><th>Significato</th></tr>
<tr><td rowspan="5"></td><td rowspan="5">SSDI8 - SSDO8</td><td>16 -:- 31</td><td></td><td>[Z] Zero.</td></tr>
<tr><td>15 - 14 - 13 (1 - 0 - 0)</td><td>0xE000 == 0x8XXX</td><td>[COK] Comunicazione OK.</td></tr>
<tr><td>15 - 14 - 13 (0 - 1 - 0)</td><td>0xE000 == 0x4XXX</td><td>[SA] Indirizzo Safety scritto nei bit 0-:-12.</td></tr>
<tr><td>15 - 14 - 13 (1 - 1 - 1)</td><td>0xE000 == 0xEXXX</td><td>[E] Dispositivo in Errore, codice di errore scritto nei bit 0-:-12.
(vedi documentazione modulo)</td></tr>
<tr><td>0 -:- 12</td><td>0x1FFF</td><td>Indirizzo safety o codice di errore.</td></tr>
<tr><td colspan="4">-1</td><td>Parametri illegali.</td></tr>
<tr><td colspan="4">-2</td><td>nIsle illegale.</td></tr>
<tr><td colspan="4">-3</td><td>nSat illegale.</td></tr>
<tr><td colspan="4">-4</td><td>Modulo inesistente.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Requisiti di sistema</td><td>RTE 34.20.1</td></tr>
<tr><td>Note</td><td></td></tr>
</table>

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
; Supponiamo l'isola 1 con un LPSDO8 come satellite 0 e un SSID8 come
int diag_0, diag_1
diag_0 = safe_diag(1, 0)
diag_1 = safe_diag(1, 1)

; Se tutto funziona correttamente otteniamo:
; diag_0 = 0x00008000 (comunicazione OK)
; diag_1 = 0x00008000 (comunicazione OK)

; Se il satellite 1 riceve un errore di Cross-circuit su IN0_Ch2:
; diag_0 = 0x0000A000 (0x8000 di comunicazione OK + 0x2000 di errore d
; diag_1 = 0x0000E128 (0xE000 di Device Error + 0x128 che indica il ti
```

Funzione safe_diag_reset()

Funzione che permette di inviare il Diagnostic Reset al satellite selezionato.

Sintassi	**i32 safe_diag_reset (i32 nIsle, i32 nSat)**
nIsle	Numero dell'isola (1-:-31).
nSat	Numero del satellite (0-:-16). Con -1 si intende tutti.
Risultato	Rende:

Valore	Significato
0	Risultato positivo.
-1	Parametri illegali.
-2	*nIsle* illegale.
-3	*nSat* illegale.
-4	Modulo inesistente.

Validità	Rule, Task.
Requisiti di sistema	RTE 34.20.1
Note	

© 2025 Robox SpA

Funzione safe_enab_out()

Funzione che permette di impostare il valore del registro di abilitazione degli output dell'isola selezionata.

Sintassi	**i32 safe_enab_out (i32 nIsle, i32 nSat, i32 value [, i32 mask=0xFFFFFFFF])**
nIsle	Numero dell'isola (1-:-31).
nSat	Numero del satellite (0-:-16).
value	Valore da imporre.
mask	Maschera su *value* (opzionale, default 0xFFFFFFFF)
Risultato	Rende: Valore / Significato 0 / Risultato positivo. -1 / Parametri illegali. -2 / *nIsle* illegale. -3 / *nSat* illegale. -4 / Modulo inesistente.
Validità	Rule, Task.
Requisiti di sistema	RTE 34.20.1
Note	

Esempi

Esempio di utilizzo:

```
; Supponiamo che nell'isola 1 satellite 3 ci sia un SSD08 con i primi
int esito
esito = safe_enab_out(1, 3, 0x1, 0xFF)
; solo il primo output è abilitato e quindi rimane alto, gli altri son
esito = safe_enab_out(1, 3, 0xFF, 0x0)
; mettendo mask = 0 gli output non impongono il loro stato abilitato/d
; tutte le uscite abilitate rimarranno abilitate e tutte quelle disabi
```

Funzione safe_oper_ack()

Funzione che permette di imporre un Operation Acknowledge sul satellite selezionato.

Sintassi	**i32 safe_oper_ack (i32 nIsle [, i32 nSat = 0])**
nIsle	Numero dell'isola (1-:-31).

© 2025 Robox SpA

Sintassi	**i32 safe_oper_ack (i32 nIsle [, i32 nSat = 0])**
nSat	Numero del satellite (0-:-16). (opzionale, default 0)
Risultato	Rende: Valore: 0 — Risultato positivo. -1 — Parametri illegali. -2 — *nIsle* illegale. -3 — *nSat* illegale. -4 — Modulo inesistente.
Validità	Rule, Task.
Requisiti di sistema	RTE 34.20.1
Note	

Esempi

Esempio di utilizzo:

```
int esito
; imposto un Oper. Ack sul satellite 3 dell'isola 2
esito = safe_oper_ack(2, 3)
if(esito < 0)
    sr(1) = "errore nell'impostazione dell'Oper. Ack"
endif
```

Funzione safe_prj_info()

Funzione che permette di avere informazioni relative al programma safety in esecuzione sull'isola specificata.

Sintassi	**i32 safe_prj_info (i32 nIsle, stru_safety_prjinfo stru)**
nIsle	Numero dell'isola (1-:-31).
stru	Struttura dati di tipo stru_safety_prjinfo che conterrà le informazioni del progetto safety.
Risultato	Rende: Valore: 0 — Risultato positivo.

© 2025 Robox SpA

Sintassi	**i32 safe_prj_info (i32 nIsle, stru_safety_prjinfo stru)**
	Valore / Significato -1 — *nIsle* illegale. -2 — *stru* = NULL. -3 — Isola inesistente. -4 — La versione di RTE non supporta questa funzione.
Validità	Task.
Requisiti di sistema	RTE 34.25.0
Note	

Esempi

Esempio di utilizzo:

```
int esito_spi
stru_safety_prjinfo spi

esito_spi = safe_prj_info(1, spi)
if(esito_spi < 0)
    ; write here your diagnostic
else
    ; set value in registers for hmi
    sr(10) = spi.prjname
    r(10)  = spi.hb_crc
    r(11)  = spi.lb_crc
    r(12)  = spi.ab_crc
endif
```

Funzione safe_sts()

Funzione che ritorna lo stato degli input/output safety del satellite selezionato.

Sintassi	**i32 safe_sts (i32 nIsle, i32 nSat [, i32 what=0])**
nIsle	Numero dell'isola (1-:-31).
nSat	Numero del satellite (0-:-16).
what	Parametro per selezionare il gruppo di IO. (opzionale, default 0)

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 safe_sts (i32 nIsle, i32 nSat [, i32 what=0])</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0</td><td>Qualsiasi.</td></tr>
<tr><td>1</td><td>Solo input.</td></tr>
<tr><td>2</td><td>Solo output.</td></tr>
</table>
</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Modulo</th><th>Significato</th></tr>
<tr><td rowspan="2">> 0</td><td>LPSDO8 - SSDO8</td><td>Stato degli output.</td></tr>
<tr><td>SSDI8</td><td>Stato degli input.</td></tr>
<tr><td colspan="2">-1</td><td>Parametri illegali.</td></tr>
<tr><td colspan="2">-2</td><td>nIsle illegale.</td></tr>
<tr><td colspan="2">-3</td><td>nSat illegale.</td></tr>
<tr><td colspan="2">-4</td><td>Modulo inesistente.</td></tr>
<tr><td colspan="2">-5</td><td>Satellite non coerente con il tipo richiesto.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Requisiti di sistema</td><td>RTE 34.20.1</td></tr>
<tr><td>Note</td><td></td></tr>
</table>

Esempi

Esempio di utilizzo:

```
; Supponiamo di avere un SSDO8 con i primi 2 output a 1, sull'isola 2
int esito
esito = safe_sts(2, 5)
; esito = 0x3
esito = safe_sts(2, 5, 1)
; esito = -5: sto chiedendo degli input a un SSDO8
```

© 2025 Robox SpA

Funzione sb0()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 0 dell'argomento con estensione del segno.

Lettura:

Sintassi	**i8 sb0 (i8|i16|i32|u8|u16|u32|real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto del primo byte dell'argomento *arg* con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**sb0 (i8|i16|i32|u8|u16|u32|real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Esempi

Esempio di utilizzo:

```
I32 pluto
; se pluto vale 0x8123
r(1) = sb0 577 (pluto) ; r(1) = 0x0023
r(2) = sb1 578 (pluto) ; r(2) = 0xff81
; paragone con b0 e b1
r(1) = b0 213 (pluto)  ; r(1) = 0x0023
r(2) = b1 215 (pluto)  ; r(2) = 0x0081
; se pluto vale 0x8234
r(1) = sb0(pluto) ; r(1) = 0x0034
r(2) = sb1(pluto) ; r(2) = 0xff82
; paragone con b0 e b1
r(1) = b0(pluto)  ; r(1) = 0x0034
r(2) = b1(pluto)  ; r(2) = 0x0082
; se pluto vale 0x1234
r(1) = sb0(pluto) ; r(1) = 0x0034
r(2) = sb1(pluto) ; r(2) = 0x0012
```

© 2025 Robox SpA

```
; paragone con b0 e b1
r(1) = b0(pluto)   ; r(1) = 0x0034
r(2) = b1(pluto)   ; r(2) = 0x0012

I16 appo,appo2,appo3
appo2 = 0x34
sb0(appo) = appo2   ; appo = 0x0034
sb1(appo3) = appo2 ; appo3 = 0x3400
; con RTE 33.9.0:
I32 pippo = 123
sb4(pippo) = 9      ; errore!
```

Funzione sb1()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 1 dell'argomento con estensione del segno.

Lettura:

Sintassi	**i8 sb1 (i16|i32|u16|u32|real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto del secondo byte dell'argomento *arg* con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**sb1 (i16|i32|u16|u32|real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	

Funzione sb2()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 2 dell'argomento con estensione del segno.

Lettura:

© 2025 Robox SpA

Sintassi	**i8 sb2 (i32|u32|real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto del terzo byte dell'argomento *arg* con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**sb2 (i32|u32|real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	

Funzione sb3()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 3 dell'argomento con estensione del segno.

Lettura:

Sintassi	**i8 sb3 (i32|u32|real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto del quarto byte dell'argomento *arg* con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**sb3 (i32|u32|real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.

© 2025 Robox SpA

Sintassi	**sb3 (i32|u32|real arg) = i8 value**
Validità	Rule, Task.
Note	

Funzione sb4()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 4 dell'argomento con estensione del segno.

Lettura:

Sintassi	**i8 sb4 (real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto del quinto byte dell'argomento *arg* con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**sb4 (real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	

Funzione sb5()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 5 dell'argomento con estensione del segno.

Lettura:

Sintassi	**i8 sb5 (real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.

© 2025 Robox SpA

Sintassi	**i8 sb5 (real arg)**
Risultato	Rende il contenuto del sesto byte dell'argomento *arg* con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**sb5 (real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	

Funzione sb6()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 6 dell'argomento con estensione del segno.

Lettura:

Sintassi	**i8 sb6 (real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto del settimo byte dell'argomento *arg* con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**sb6 (real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	

© 2025 Robox SpA

Funzione sb7()

Questa funzione si usa per leggere o scrivere (istruzione) il byte 7 dell'argomento con estensione del segno.

Lettura:

Sintassi	**i8 sb7 (real arg)**
arg	Variabile di cui si vuole conoscere il contenuto. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende il contenuto dell'ottavo byte dell'argomento *arg* con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Scrittura (istruzione):

Sintassi	**sb7 (real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	

Funzione sbit()

Funzione che permette di settare uno o più bit in un dato di tipo intero in funzione del verificarsi di una condizione.

Sintassi	**i32 sbit (i32 data, i32 bitMsk, bool cond)**
data	Grandezza sulla quale si desidera effettuare il set.
bitMsk	Maschera dei bit da settare.
cond	Condizione al verificarsi della quale i bit verranno settati.
Risultato	Rende: Valore: 1 — Significato: La condizione è verificata. Valore: 0 — Significato: Altrimenti.

© 2025 Robox SpA

Sintassi	**i32 sbit (i32 data, i32 bitMsk, bool cond)**
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
sbit(r(1), 0x000000ff, r(100) = 34)
; mette a 1 gli 8 bit bassi quando r(100) è uguale a 34
; equivale a:
if(r(100) = 34))
    r(1) = r(1) r_or 0x000000ff
endif
```

Funzione sdw0()

Questa funzione si usa per leggere o scrivere (istruzione) la dword 0 dell'argomento con estensione del segno.

Lettura:

Sintassi	**i32 sdw0 (i32|u32|real arg)**
arg	Argomento della Dword. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende la Dword dell'argomento con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una dword non esistente ritorna un errore.

Scrittura (istruzione):

Sintassi	**sdw0 (i32|u32|real arg) = i32 value**
arg	Argomento della Dword.
value	Valore da imporre in arg
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una dword non esistente ritorna un errore.

Esempi

Esempio di utilizzo:

```
REAL pippo1, pippo2, pippo3, pippo4
```

© 2025 Robox SpA

```
REAL pippo5, pippo6, pippo7, pippo8
INT prova[4]
prova[0] = 0x1122
prova[1] = 0x8001
prova[2] = 0x5566
prova[3] = 0x7788

pippo1 = sdw0(prova[0]) ; pippo1 = -2147413726
pippo2 = sdw1(prova[0]) ; pippo2 = 2005423462
pippo3 = sdw0(prova[2]) ; pippo3 = 2005423462
pippo4 = sdw1(prova[2]) ; pippo4 = 0
; confronto con l'istruzione DW
pippo5 = dw0(prova[0])  ; pippo1 = 2147553570
pippo6 = dw1(prova[0])  ; pippo2 = 2005423462
pippo7 = dw0(prova[2])  ; pippo3 = 2005423462
pippo8 = dw1(prova[2])  ; pippo4 = 0

REAL pluto1, pluto2, pluto3, pluto4
REAL pluto5, pluto6, pluto7, pluto8
INT prova2[4]
prova2[0] = 0x1122
prova2[1] = 0x2233
prova2[2] = 0x5566
prova2[3] = 0x7788

pluto1 = sdw0(prova2[0]) ; pluto1 = 573772066
pluto2 = sdw1(prova2[0]) ; pluto2 = 2005423462
pluto3 = sdw0(prova2[2]) ; pluto3 = 2005423462
pluto4 = sdw1(prova2[2]) ; pluto4 = 0
; confronto con l'istruzione DW (analoghi)
pluto5 = dw0(prova2[0])  ; pluto1 = 573772066
pluto6 = dw1(prova2[0])  ; pluto2 = 2005423462
pluto7 = dw0(prova2[2])  ; pluto3 = 2005423462
pluto8 = dw1(prova2[2])  ; pluto4 = 0

; con RTE 33.9.0:
I16 pippo = 123
sdw1(pippo) = 9  ; errore!
```

Funzione sdw1()

Questa funzione si usa per leggere o scrivere (istruzione) la dword 1 dell'argomento con estensione del segno.

Lettura:

Sintassi	**i32 sdw1 (real arg)**
arg	Argomento della Dword. Da RTE 33.7.1 può essere un'espressione qualunque.

© 2025 Robox SpA

Sintassi	**i32 sdw1 (real arg)**
Risultato	Rende la Dword dell'argomento con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una dword non esistente ritorna un errore.

Scrittura (istruzione):

Sintassi	**sdw1 (real arg) = i32 value**
arg	Argomento della Dword.
value	Valore da imporre in arg
Validità	Rule, Task.
Note	

Funzione ser_485_tx_sts()

Funzione che rende lo stato dell'ultima trasmissione fatta sul dispositivo specificato.

Sintassi	**i32 ser_485_tx_sts (i32 nSer)**
nSer	Numero del canale seriale.
Risultato	Rende il codice di stato. Il significato dei bit che compongono il codice di stato è:

Bit	Valore esadecimale	Significato
0	0x00000001	Trasmissione 485 in corso.
1	0x00000002	Trasmissione finita (inviato ultimo carattere).
2	0x00000004	Esito positivo - trasmissione finita e ricevuto anche ultimo carattere trasmesso.
3	0x00000008	Esito negativo - trasmissione abortita (causa bit 8).

© 2025 Robox SpA

Sintassi	**i32 ser_485_tx_sts (i32 nSer)**

Bit	Valore esadecimale	Significato
4	0x00000010	Esito negativo - trasmissione abortita, timeout ricezione.
5	0x00000020	Libero.
6	0x00000040	Libero.
7	0x00000080	Errori di trasmissione/ricezione.
8	0x00000100	Esito negativo - seriale non in modo 485 (12).
9	0x00000200	Esito negativo - seriale non gestibile in modo 485.

Validità	Rule, Task.
Note	

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se *nSer* illegale
- Genera 9901 ill. Funct. T:x ST:xxx se la seriale non è in modo 485 (vedi ser_mod 592)

Esempi

Esempio di utilizzo:

```
function SendCommand()
    ; Seriale 4
    ; Invio sulla seriale NUMSER lungh caratteri
    lungh = str_len 631(StrTx)
    ser_bin_out 588(NUMBER, StrTx, lungh)
    FotoTempo = tfb
    loopTx:
    if((ser_485_tx_sts 585(NUMSER) r_and 806 0x31C))
        ErroriTx = (ser_485_tx_sts(NUMSER) r_and 0x318)
        return
    endif
    if((tfb - FotoTempo) > 1.0)
        ; Timeout
```

© 2025 Robox SpA

```
        ErroriTx = 0x2000
        return
    endif
    goto loopTx
    return
end_fun
```

Funzione ser_bcc_h_init()

Funzione che esegue l'impostazione dei dati che servono per inviare su seriale un messaggio BCC (destinato ad un altro controllo Robox).

Sintassi	**i32 ser_bcc_h_init (bcc_msguser stru, i32 nSer, i32 nChDst, i32 nChSrc, i32 msgCode, i32 len)**
stru	Struttura di tipo bcc_msguser 820 usata per i dati da inviare. La struttura dei dati da spedire può essere la stessa dei dati ricevuti.
nSer	Numero del canale seriale (1, 2, 3, 4).
nChDst	Numero del canale utente da utilizzare per destinazione (190 -:- 199).
nChSrc	Numero del canale utente sorgente da utilizzare per eventuali risposte (190 -:- 199).
msgCode	Codice del messaggio (10000 -:- 19999).
len	Lunghezza dei dati dei messaggi (max 255 byte).
Risultato	Rende:

Valore	Significato
<> 0	Operazione eseguita correttamente.
0	Altrimenti.

Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; prepara seriale 2 per trasmettere su canale 190 la struttura blocco
; variabile di tipo struttura bcc_msguser (è una struttura predefinita
reso = ser_bcc_h_init 587(blocco, 3, 190, 190, 10000, 5*4)  ; 5 * 4 byte
; blocco = struttura
```

© 2025 Robox SpA

```
; 3 = usa seriale 3
; 190 = trasmetti sul canale utente 190
; 190 = se lo slave risponde con bcc_h_setup() up usa il canale 190
; 10000 = codice di questo messaggio 10000 (a discrezione utente nel r
; 5 * 4 = 20 byte = 5 registri interi
```

Funzione ser_bin_in()

Funzione che permette di leggere un numero specifico di caratteri dal dispositivo specificato.

Sintassi	**i32 ser_bin_in (i32 nSer, string dst, i32 nChar)**
nSer	Numero del canale seriale.
dst	Buffer in cui scaricare le informazioni.
nChar	Numero di byte da leggere.
Risultato	Rende: Valore: <> 0 — Operazione eseguita correttamente. Valore: 0 — Altrimenti.
Validità	Rule, Task.
Note	

La funzione non è bloccante. Se vengono richiesti in lettura n caratteri e nel buffer di ricezione sono presenti r caratteri, il comportamento è il seguente:

- r < n, nessun carattere viene letto e la funzione rende 0
- r >= n, vengono letti esattamente n caratteri e la funzione rende un valore TRUE

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se *nSer* illegale
- Genera 9901 ill. Funct. T:x ST:xxx se la seriale non è in modo 485 o singolo carattere (vedi ser_mod 592)

Funzione ser_bin_out()

Funzione che permette di inviare un numero specifico di caratteri sul dispositivo specificato.

Sintassi	**i32 ser_bin_out (i32 nSer, string src, i32 nChar)**
nSer	Numero del canale seriale.

© 2025 Robox SpA

Sintassi	**i32 ser_bin_out (i32 nSer, string src, i32 nChar)**
src	Buffer in cui scaricare le informazioni.
nChar	Numero di byte da inviare.
Risultato	Rende: Valore: <> 0 — Significato: Operazione eseguita correttamente. Valore: 0 — Significato: Altrimenti.
Validità	Rule, Task.
Note	

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se *nSer* illegale
- Genera 9901 ill. Funct. T:x ST:xxx se la seriale non è in modo 485 o singolo carattere (vedi ser_mod 592)

Funzione ser_bin_ready()

Funzione che rende il numero di caratteri presenti sul canale seriale specificato.

Sintassi	**i32 ser_bin_ready (i32 nSer)**
nSer	Numero del canale seriale.
Risultato	Rende il numero di caratteri.
Validità	Rule, Task.
Note	

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se *nSer* illegale
- Genera 9901 ill. Funct. T:x ST:xxx se la seriale non è in modo 485 o singolo carattere (vedi ser_mod 592)

© 2025 Robox SpA

Funzione ser_buff_free()

Funzione che rende il numero di byte liberi nel buffer di trasmissione della seriale specificata.

Sintassi	**i32 ser_buff_free (i32 nSer)**
nSer	Numero del canale seriale.
Risultato	Rende: Valore: Significato >= 0: Numero di byte liberi. < 0: *nSer* illegale.
Validità	Rule, Task.
Note	

Funzione ser_buff_queue()

Funzione che rende il numero di byte occupati nel buffer di trasmissione della seriale specificata.

Sintassi	**i32 ser_buff_queue (i32 nSer)**
nSer	Numero del canale seriale.
Risultato	Rende: Valore: Significato >= 0: Numero di byte occupati. < 0: *nSer* illegale.
Validità	Rule, Task.
Note	

Funzione ser_flush()

Funzione che pulisce tutti i dati nei buffer di tx e rx della seriale specificata.

Sintassi	**i32 ser_flush (i32 nSer)**
nSer	Numero del canale seriale.
Risultato	Rende:

© 2025 Robox SpA

Sintassi	**i32 ser_flush (i32 nSer)**
	Valore / Significato 0 — Operazione eseguita con successo. -2 — *nSer* illegale.
Validità	Rule, Task.
Note	

Funzione ser_in()

Funzione che sposta una stringa dal buffer di ricezione del dispositivo specificato alla stringa di destinazione.

Sintassi	**i32 ser_in (i32 nSer, string dst)**
nSer	Numero del canale seriale.
dst	Buffer in cui scaricare le informazioni.
Risultato	Rende: Valore / Significato <> 0 — Operazione eseguita correttamente. 0 — Altrimenti.
Validità	Rule, Task.
Note	Il canale deve necessariamente essere inizializzato come ascii (PSER_ASCII).

La stringa deve essere terminata dal terminatore attivo (terminator 778) o dal terminatore di default LF (10 dec. - 0AH) o dal raggiungimento del numero di caratteri massimo (83).

Se nel buffer specificato la stringa non è terminata allora la funzione rimane in attesa.

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se *nSer* illegale
- Genera 9901 ill. Funct. T:x ST:xxx se la seriale non è in modo 485 o singolo carattere (vedi ser_mod 592)

© 2025 Robox SpA

Funzione ser_mod()

Questa funzione è usata per definire le modalità di lavoro dei dispositivi seriali.

<table>
<tr><td>Sintassi</td><td>i32 ser_mod (i32 nSer, i32 mod)</td></tr>
<tr><td>nSer</td><td>Numero del canale seriale (1, 2, 3, 4).</td></tr>
<tr><td>mod</td><td>Rappresenta la modalità di lavoro prescelta per il canale seriale:
<table>
<tr><td>Valore</td><td>Modalità</td></tr>
<tr><td>0</td><td>Protocollo Robox (DLE BCC) - con handshake (DTR-DSR) se supportato da hardware.</td></tr>
<tr><td>1</td><td>Comunicazione seriale pura - con handshake (DTR-DSR).</td></tr>
<tr><td>2</td><td>Comunicazione DLE-ASCII - senza interblocchi.</td></tr>
<tr><td>12</td><td>RS485.</td></tr>
<tr><td>17</td><td>Comunicazione seriale pura - senza handshake.</td></tr>
<tr><td>18</td><td>Comunicazione LFASCII - senza handshake.</td></tr>
</table></td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><td>Valore</td><td>Significato</td></tr>
<tr><td><> 0</td><td>La porta è stata configurata correttamente.</td></tr>
<tr><td>0</td><td>Altrimenti.</td></tr>
</table></td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td>Da RTE 34.22.01 se la seriale del dispositivo Robox non ha DTR e DSR, la loro gestione viene esclusa (es. su SER2 di RP2).</td></tr>
</table>

Diagnostica

- 9900 ill. Arg A:1 T:x ST:xxx se numero di seriale illegale oppure se la modalità scelta non è compatibile con il canale seriale

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
int esito
; programmazione del canale seriale 3 in modalità 485
esito = ser_mod(3, 12)
if(esito)
    sr(1) = "configurazione eseguita correttamente"
endif
```

Funzione ser_out()

Funzione che scrive una stringa nel buffer del dispositivo specificato.

Sintassi	**i32 ser_out (i32 nSer, string src)**
nSer	Numero del canale seriale.
src	Stringa contenente i dati da inviare.
Risultato	Rende: Valore / Significato <> 0 / Operazione eseguita correttamente. 0 / Altrimenti.
Validità	Rule, Task.
Note	Il canale deve necessariamente essere inizializzato come ascii (PSER_ASCII).

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se *nSer* illegale
- Genera 9901 ill. Funct. T:x ST:xxx se la seriale non è in modo stringa terminata (vedi ser_mod)

Funzione ser_r_sts()

Funzione che permette di leggere la parola di stato del canale seriale specificato.

Sintassi	**i32 ser_r_sts (i32 nSer)**
nSer	Numero del canale seriale.
Risultato	Contiene la parola di stato del canale seriale come segue:

© 2025 Robox SpA

Sintassi	**i32 ser_r_sts (i32 nSer)**
	<table><tr><th>Valore</th><th>Significato</th></tr><tr><td>1</td><td>DTR.</td></tr><tr><td>2</td><td>RTS.</td></tr><tr><td>4</td><td>DSR.</td></tr><tr><td>8</td><td>CTS.</td></tr><tr><td>16</td><td>Tutto inviato.</td></tr><tr><td>32</td><td>CRC/framing error.</td></tr><tr><td>64</td><td>RX overrun.</td></tr><tr><td>128</td><td>Parity error.</td></tr><tr><td>256</td><td>RX carattere disponibile.</td></tr><tr><td>512</td><td>TX buffer vuoto.</td></tr></table>
Validità	Rule, Task.
Note	È possibile leggere lo stato di questi segnali solamente se è stata preventivamente selezionata la modalità MODEM per il canale seriale specificato (vedi ser_mod 592).

Funzione ser_ready()

Funzione che controlla se nel buffer del dispositivo specificato è presente una stringa terminata.

Sintassi	**i32 ser_ready (i32 nSer)**
nSer	Numero del canale seriale.
Risultato	Rende:<table><tr><th>Valore</th><th>Significato</th></tr><tr><td><> 0</td><td>Nel buffer del dispositivo specificato è presente una stringa terminata.</td></tr><tr><td>0</td><td>Altrimenti.</td></tr></table>
Validità	Rule, Task.
Note	Il canale deve necessariamente essere inizializzato come ascii (PSER_ASCII).

© 2025 Robox SpA

La stringa deve essere terminata dal terminatore attivo (terminator 778) o dal terminatore di default LF (10 dec. - 0AH) o dal raggiungimento del numero di caratteri massimo (83).

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se *nSer* illegale
- Genera 9901 ill. Funct. T:x ST:xxx se la seriale non è inizializzata come ascii

Funzione set_kinematics()

Funzione che impone contemporaneamente le 3 grandezze cinematiche.

In caso di limitazione, vengono ricalcolati i valori.

Sintassi	**i32 set_kinematics (i32 nAx, real ipDes, real ivDes [, real iaDes=0])**
nAx	Numero dell'asse (1-:-32).
ipDes	Posizione asse desiderata [unit].
ivDes	Velocità asse desiderata [unit/s].
iaDes	Accelerazione asse desiderata [unit/s^2]. (opzionale, default 0)
Risultato	Rende: Valore / Significato -1 / *nAx* illegale. 0 / Operazione ok. 1 / Limitato a causa di IP. 2 / Limitato a causa di IV. 3 / Limitato a causa di IA.
Validità	Rule.
Requisiti di sistema	RTE 33.11.3
Note	Da utilizzarsi in caso di CAN impostato in modalità interpolazione cubica (invio target di posizione e velocità).

Funzione set_virt_iw()

Funzione che imposta il valore reale in un input word virtuale.

Per stato reale di un ingresso si intende lo stato fisico. L'applicativo leggerà poi lo stato fisico modificato da eventuale forzamento.

© 2025 Robox SpA

Sintassi	**i32 set_virt_iw (i32 nIW, i32 sts)**
nIW	Numero di inp_w a cui viene impostato lo stato reale.
sts	Valore da impostare.
Risultato	Rende: Valore: <> 0 — Significato: Operazione eseguita con successo. Valore: 0 — Significato: L'indirizzo specificato non appartiene al range di IO virtuali definiti.
Validità	Rule, Task.
Note	

Funzione shift_left()

Funzione che esegue uno shift logico a sinistra di un bit, in una finestra specificata, di un array u32, considerato di fatto come array di bit.

Sintassi	**i32 shift_left (u32 array, i32 idStart, i32 nBit, i32 stsBit)**
array	Array di riferimento. **NOTA**: Anche se di un solo elemento deve essere dichiarato come variabile.
idStart	Indice del bit di inizio della finestra. Il numero deve essere compreso tra 0 e (arrayLength*32 - 1). **ATTENZIONE**: Il controllo di non sbordare è a carico dell'utilizzatore.
nBit	Numero di bit di dimensione della finestra.
stsBit	Valore booleano del bit entrante.
Risultato	Rende il valore (0 o 1) del bit uscente (il più alto) dalla finestra in cui è eseguito lo shift.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

© 2025 Robox SpA

```
u32 arr[2]
reso = shift_left(arr, 32, 16, 1)
; ad ogni esecuzione sulla seconda dword dell'array arr verrà eseguita
```

		arr[1]								arr[0]							
Valore iniziale		1010	1010	1010	1010	0000	0000	0000	0000	1010	1010	1010	1010	1010	1010	1010	1010
# esecuzione	Reso					Finestra (dal bit di indice 32 per 16 bit)											
1	0	1010	1010	1010	1010	**0000**	**0000**	**0000**	**0001**	1010	1010	1010	1010	1010	1010	1010	1010
2	0	1010	1010	1010	1010	**0000**	**0000**	**0000**	**0011**	1010	1010	1010	1010	1010	1010	1010	1010
...	0	1010	1010	1010	1010	**...**	**...**	**...**	**...**	1010	1010	1010	1010	1010	1010	1010	1010
15	0	1010	1010	1010	1010	**0111**	**1111**	**1111**	**1111**	1010	1010	1010	1010	1010	1010	1010	1010
16	0	1010	1010	1010	1010	**1111**	**1111**	**1111**	**1111**	1010	1010	1010	1010	1010	1010	1010	1010
17	1	1010	1010	1010	1010	**1111**	**1111**	**1111**	**1111**	1010	1010	1010	1010	1010	1010	1010	1010

Funzione shift_right()

Funzione che esegue uno shift logico a destra di un bit, in una finestra specificata, di un array u32, considerato di fatto come array di bit.

Sintassi	**i32 shift_right (u32 array, i32 idStart, i32 nBit, i32 stsBit)**
array	Array di riferimento. **NOTA**: Anche se di un solo elemento deve essere dichiarato come variabile.
idStart	Indice del bit di inizio della finestra. Il numero deve essere compreso tra 0 e (arrayLength*32 - 1). **ATTENZIONE**: Il controllo di non sbordare è a carico dell'utilizzatore.

© 2025 Robox SpA

Sintassi	**i32 shift_right (u32 array, i32 idStart, i32 nBit, i32 stsBit)**
nBit	Numero di bit di dimensione della finestra.
stsBit	Valore booleano del bit entrante.
Risultato	Rende il valore (0 o 1) del bit uscente (il più basso) dalla finestra in cui è eseguito lo shift.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
u32 arr[2]
reso = shift_right(arr, 16, 16, 1)
; ad ogni esecuzione sulla seconda word dell'array arr verrà eseguita
```

		arr[1]								arr[0]							
Valore iniziale:		1010	1010	1010	1010	0000	0000	0000	0000	1010	1010	1010	1010	1010	1010	1010	1010
# esecuzione	Reso									Finestra (dal bit di indice 16 per 16 bit)							
1	0	1010	1010	1010	1010	1010	1010	1010	1010	**1000**	**0000**	**0000**	**0000**	1010	1010	1010	1010
2	0	1010	1010	1010	1010	1010	1010	1010	1010	**1100**	**0000**	**0000**	**0000**	1010	1010	1010	1010
...	0	1010	1010	1010	1010	1010	1010	1010	1010	**...**	**...**	**...**	**...**	1010	1010	1010	1010
15	0	1010	1010	1010	1010	1010	1010	1010	1010	**1111**	**1111**	**1111**	**1110**	1010	1010	1010	1010
16	0	1010	1010	1010	1010	1010	1010	1010	1010	**1111**	**1111**	**1111**	**1111**	1010	1010	1010	1010
17	1	1010	1010	1010	1010	1010	1010	1010	1010	**1111**	**1111**	**1111**	**1111**	1010	1010	1010	1010

© 2025 Robox SpA

Funzione shl()

Funzione che esegue uno shift logico a sinistra del numero di bit specificato.

Sintassi	**i32 shl (i32 value, i32 nBit)**
value	Variabile su cui eseguire lo shift.
nBit	Numero di bit da traslare.
Risultato	Rende il risultato dello shift.
Validità	Rule, Task.
Note	Nello shift a sinistra il bit più pesante uscente viene perso, nel bit più leggero entra sempre 0. Si comporta in modo analogo a r_shl551().

Esempi

Esempio di utilizzo:

```
r(1) = shl599(r(1), 1)
```

# esecuzione								
Valore iniziale	0000	0000	0000	0000	0000	0000	0000	000**1**
1	0000	0000	0000	0000	0000	0000	0000	00**1**0
2	0000	0000	0000	0000	0000	0000	0000	0**1**00
...								
31	**1**000	0000	0000	0000	0000	0000	0000	0000
32	0000	0000	0000	0000	0000	0000	0000	0000

Funzione shr()

Funzione che esegue uno shift logico a destra del numero di bit specificato.

Sintassi	**i32 shr (i32 value, i32 nBit)**
value	Variabile su cui eseguire lo shift.
nBit	Numero di bit da traslare.
Risultato	Rende il risultato dello shift.
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**i32 shr (i32 value, i32 nBit)**
Note	Nello shift a destra il bit più leggero uscente viene perso, nel bit più pesante entra sempre 0 (viene perso il segno). Per mantenere il segno usare r_shr552().

Esempi

Esempio di utilizzo:

```
r(1) = shr599(r(1), 1)
```

# esecuzione								
Valore iniziale	**1**000	0000	0000	0000	**1**000	0000	0000	000**1**
1	0**1**00	0000	0000	0000	0**1**00	0000	0000	0000
2	00**1**0	0000	0000	0000	00**1**0	0000	0000	0000
...								
31	0000	0000	0000	0000	0000	0000	0000	000**1**
32	0000	0000	0000	0000	0000	0000	0000	0000

Funzione sign()

Funzione che rende 0 oppure rende 1 con il segno del valore specificato.

Sintassi	**i32 sign (i32|real value)**
value	Variabile real o i32 dalla quale prendere il segno.
Risultato	Rende: Condizione: Valore *value* < 0: -1 *value* = 0: 0 *value* > 0: +1
Validità	Rule, Task.
Note	

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
if(abs(rr(5)) > 12.5)
    rr(5) = 12.5 * sign(rr(5))
end_if
; Assumendo rr(5)= -30, dopo l'esecuzione delle istruzioni rr(5) sarà
```

Funzione similar()

Funzione che controlla se due variabili sono uguale a meno di K_EPSILON.

Sintassi	**i32 similar (real var1, real var2 [, real epsThr=K_EPSILON])**
var1	Prima variabile.
var2	Seconda variabile.
epsThr	Soglia di controllo. (opzionale, default K_EPSILON)
Risultato	Rende:

Condizione	Valore	Significato
|*var1* - *var2*| < *epsThr*	<> 0	Le variabili sono simili a meno di *epsThr*.
|*var1* - *var2*| > *epsThr*	0	Le variabili differiscono di un valore superiore a *epsThr.*

Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
result = similar(rr(1), rr(2))
if(abs((rr(1) - rr(2)) < K_EPSILON)
    result = TRUE
else
    result = FALSE
endif
```

© 2025 Robox SpA

Funzione sin()

Funzione che esegue il seno dell'angolo in radianti specificato.

Sintassi	**real sin (real value)**
value	Argomento del seno. **NOTA**: *value* deve essere in radianti.
Risultato	Rende il seno di *value*.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
rr(5) = 0

rr(10) = sin(rr(5))              ; = 0
rr(10) = sin(K_PI/4)             ; = 0.7071...
rr(101) = sin(torad(rr(100)))  ; rr (100) --> angolo in gradi
                                 ; rr (101) --> seno dell'angolo contenuto
```

Funzione sizeof()

Funzione che ritorna la dimensione della variabile indicata.

Sintassi	**i32 sizeof (real arg)**
arg	Variabile di cui si vuole sapere la dimensione.
Risultato	Rende la dimensione di *var*.
Validità	Rule, Task.
Note	*var* non può essere di tipo sr o nvsr.

Esempi

Esempio di utilizzo:

```
; dichiarazione variabili
string str1
string str2[1000]
int dim

STRUCT record
    int p1
    int p2
    real p3
    real p4
```

© 2025 Robox SpA

```
    real p5
END_STRUCT
record arrow[2]

dim = sizeof(str1)       ; dim = 128
dim = sizeof(str2)       ; dim = 128*1000
dim = sizeof("1234567")  ; dim = 8
dim = sizeof(record)     ; dim = 32
dim = sizeof(arrow)      ; dim = 32*2
dim = sizeof(1)          ; dim = 4
dim = sizeof(1.1)        ; dim = 8
dim = sizeof(r(1))       ; dim = 4
dim = sizeof(rr(1))      ; dim = 8
```

Funzione soe_ridn()

Funzione che permette di leggere un parametro SoE (lettura del parametro *stru*.idn sul dispositivo identificato da *stru*.rid e informazioni ricevute memorizzate in *destination*).

La lettura può avvenire in modo passante o in modo bloccante (utilizzabile solo nei task 1-:-8).

Per modo passante si intende che la funzione deve essere continuamente chiamata e questa rende lo stato attuale dell'operazione.

Per modo bloccante si intende che la funzione viene invocata e ritorna solo quando ha completato l'operazione di lettura.

La funzione rende un esito sullo stato dell'esecuzione.

Sintassi	**i32 soe_ridn (stru_soe_idn stru, destination)**
stru	Struttura di tipo stru_soe_idn contenente le informazione per la lettura di un parametro SoE.
destination	Contiene le informazioni ricevute. *destination* deve essere un oggetto compatibile con i dati che si stanno leggendo (se si legge una VISIBLE STRING, *destination* sarà di tipo string, se si legge un OCTET_STRING sarà un array di byte o una struttura). In caso di lettura di array verranno gestite solo informazioni la cui lunghezza è <= a 256 byte. Dovendo leggere informazioni di maggior dimensione utilizzare come tipo dati OCTET_STRING e fornire un buffer di destinazione sufficientemente grande (ovviamente l'interpretazione dei dati è a carico dell'applicativo).
Risultato	Variabile contenente il risultato. Vedi gestione passante e/o bloccante.
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**i32 soe_ridn (stru_soe_idn stru, destination)**
Requisiti di sistema	RTE 34.18.3
Note	

Gestione passante

- Imporre ***stru*.phase = 1** per inizializzare la lettura.
- Se non specificato il tipo di dato (*stru*.dataType=0), viene eseguita la lettura del tipo di dato su dispositivo remoto. La variabile *stru*.phase viene cambiata.
  La funzione rende 0.
- Il comando di lettura viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.
  La funzione rende 0.
- Attesa della risposta.
  La funzione continua a rendere 0.
- Ricezione risposta con esito **positivo**. Il valore letto viene scritto in *destination*. RTE cambia *stru*.phase = 4.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -2, gestore comunicazione occupato
  - -3, crash gestore comunicazione (riavvio)
  - -5, comando a un dispositivo non presente (*stru*.rid)
  - -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata
  - -12, valore illegale di idn (allarme 9900 arg 12)
  - -13, valore illegale di element/cmd (allarme 9900 arg 13)
  - -14, valore illegale di datatype (allarme 9900 arg 4/arg 14)
  - -15, valore illegale di datalen (allarme 9900 arg 5/arg 15)
  - -16, valore illegale di phase (allarme 9900 arg 6)
  - -21, destinazione non specificata (allarme 9900 arg 16)
  - -22, destinazione cambiata (allarme 9900 arg 32)
  - -23, errore software
  - 2, comando di cui si è già ricevuto risposta positiva, valore disponibile in *destination*
  - >2, codice di errore SDO, vedi tabella sotto

Gestione bloccante

- Imporre ***stru*.phase = -1** per eseguire la lettura.
- Se non specificato il tipo di dato (*stru*.dataType=0), viene eseguita la lettura del tipo di dato su dispositivo remoto. La variabile *stru*.phase viene cambiata.
- Il comando di lettura viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.

© 2025 Robox SpA

- Attesa della risposta. Se la risposta non arriva entro un massimo di 3 secondi la funzione esce con un errore (timeout, rende -2).
- Ricezione risposta con esito **positivo**. Il valore letto viene scritto in *destination*. RTE cambia *stru*.phase = 4.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -2, gestore comunicazione occupato
  - -3, crash gestore comunicazione (riavvio)
  - -4, gestione bloccante non permessa nel task corrente (*stru*.phase = 1) (rule, ladder). Comando non accettato
  - -5, comando a un dispositivo non presente (*stru*.rid)
  - -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata
  - -12, valore illegale di idn (allarme 9900 arg 12)
  - -13, valore illegale di element/cmd (allarme 9900 arg 13)
  - -14, valore illegale di datatype (allarme 9900 arg 4/arg 14)
  - -15, valore illegale di datalen (allarme 9900 arg 5/arg 15)
  - -16, valore illegale di phase (allarme 9900 arg 6)
  - -21, destinazione non specificata (allarme 9900 arg 16)
  - -22, destinazione cambiata (allarme 9900 arg 32)
  - -23, errore software
  - 2, comando di cui si è già ricevuto risposta positiva, valore disponibile in *destination*
  - >2, codice di errore SDO, vedi tabella sotto

Codici di errore SDO SoE

SDO error code	Descrizione
0x0009	Accesso non valido all'elemento 0.
0x1001	Non esiste.
0x1009	Accesso non valido all'elemento 1.
0x2001	Nome inesistente.
0x2002	Nome sottodimensionato in trasmissione.
0x2003	Nome sovradimensionato in trasmissione.
0x2004	Nome non si può cambiare.
0x2005	Nome attualmente protetto da scrittura.
0x3002	Attributo sottodimensionato in trasmissione.

© 2025 Robox SpA

SDO error code	Descrizione
0x3003	Attributo sovradimensionato in trasmissione.
0x3004	Attributo non si può cambiare.
0x3005	Attributo attualmente protetto da scrittura.
0x4001	Unità inesistente.
0x4002	Unità sottodimensionato in trasmissione.
0x4003	Unità sovradimensionato in trasmissione.
0x4004	Unità non si può cambiare.
0x4005	Unità attualmente protetto da scrittura.
0x5001	Valore minimo in input inesistente.
0x5002	Valore minimo in input sottodimensionato in trasmissione.
0x5003	Valore minimo in input sovradimensionato in trasmissione.
0x5004	Valore minimo in input non si può cambiare.
0x5005	Valore minimo in input attualmente protetto da scrittura.
0x6001	Valore massimo in input inesistente.
0x6002	Valore massimo in input sottodimensionato in trasmissione.
0x6003	Valore massimo in input sovradimensionato in trasmissione.
0x6004	Valore massimo in input non si può cambiare.
0x6005	Valore massimo in input attualmente protetto da scrittura.
0x7001	Elemento inesistente.
0x7002	Elemento sottodimensionato in trasmissione.
0x7003	Elemento sovradimensionato in trasmissione.
0x7004	Elemento non si può cambiare.
0x7005	Elemento attualmente protetto da scrittura.
0x7006	Elemento inferiore al limite di *minimum input value.*

© 2025 Robox SpA

SDO error code	Descrizione
0x7007	Elemento superiore al limite di *maximum input value.*
0x7008	Elemento non corretto.
0x7009	Elemento protetto da password.
0x700A	Elemento temporaneamente non modificabile (AT o MDT).
0x700B	Indirettamente non valido.
0x700C	Elemento temporaneamente non modificabile (parametro o modo operativo).
0x7010	Comando già attivo.
0x7011	Comando non interrompibile.
0x7012	Comando non disponibile (in questa fase).
0x7013	Comando non disponibile (parametro non valido).

Codici di errore SDO SoE/CoE

SDO error code	Descrizione
0x0500 0000	Trovato errore generale protocollo SDO.
0x0503 0000	Toggle bit non alternato.
0x0504 0000	Timeout del protocollo SDO.
0x0504 0001	Identificatore del comando client/server non valido o sconosciuto.
0x0504 0002	Dimensione del blocco non valida (solo gestione bloccante).
0x0504 0003	Numero di sequenza non valido (solo gestione bloccante).
0x0504 0004	Errore CRC (solo gestione bloccante).
0x0504 0005	Eccesso di memoria.
0x0601 0000	Accesso non supportato a un oggetto.
0x0601 0001	Tentativo di leggere un oggetto a sola scrittura.
0x0601 0002	Tentativo di scrivere un oggetto a sola lettura.
0x0602 0000	L'oggetto non esiste nel dizionario oggetti.

© 2025 Robox SpA

SDO error code	Descrizione
0x0604 0041	Impossibile mappare l'oggetto sul PDO.
0x0604 0042	Il numero e la lunghezza degli oggetti da mappare supererebbero la lunghezza del PDO.
0x0604 0043	Incompatibilità dei parametri.
0x0604 0047	Incompatibilità interna al dispositivo.
0x0606 0000	Accesso non riuscito a causa di un errore hardware.
0x0607 0010	Il tipo di dati non corrisponde, la lunghezza del parametro di servizio non corrisponde.
0x0607 0012	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo alto.
0x0607 0013	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo basso.
0x0609 0011	Il sottoindice non esiste.
0x0609 0030	Superamento del campo di valori del parametro (solo per accesso in scrittura).
0x0609 0031	Valore del parametro scritto troppo alto.
0x0609 0032	Valore del parametro scritto troppo basso.
0x0609 0036	Il valore massimo è inferiore al valore minimo.
0x060A 0023	Risorsa non disponibile: connessione SDO.
0x0800 0000	Errore generale.
0x0800 0020	I dati non possono essere trasferiti o archiviati nell'applicazione.
0x0800 0021	I dati non possono essere trasferiti o archiviati nell'applicazione a causa del controllo locale.
0x0800 0022	I dati non possono essere trasferiti o archiviati nell'applicazione a causa dello stato attuale del dispositivo.
0x0800 0023	La generazione dinamica del dizionario oggetti non riesce o non è presente alcun dizionario oggetti (ad esempio, il dizionario oggetti viene generato da un file e la generazione non riesce a causa di un errore di file).
0x0800 0024	Nessun dato disponibile.

© 2025 Robox SpA

SDO error code	Descrizione
Altri valori	Codice non valido - non è presente CiA DS301 v4.1.

Esempi

Esempio di utilizzo:

```
$task 2
INT      idn, pIdn
INT      rid, pRid
INT      idn_sts
STRING   idn_name
U32      idn_attr
STRING   idn_unit
INT      idn_min
INT      idn_max
INT      idn_data
INT      idn_default
U32      flgData
LIT      isBlob  0x1
LIT      isString 0x2
U8       idn_data_blob[260]
STRING   idn_data_string
I16      idn_data_I16[200]
I32      idn_data_I32[200]
stru_soe_idn 909 soe
INT esitoSoe
INT      wflgData
INT      widn_data
U8       widn_data_blob[260]
STRING   widn_data_string
I16      widn_data_I16[200]
I32      widn_data_I32[200]
stru_soe_idn wsoe
INT wesitoSoe
INT stopExec , blocca
REAL tdwell


pIdn = -1
pRid = -1
stopExec= FALSE
flgData = 0
rid = 2
idn = 51
tdwell = 0.1
blocca = FALSE

__MAIN_LOOP__
```

© 2025 Robox SpA

```
if((rid <> pRid) or (idn <> pIdn))
    pRid = rid
    pIdn = idn

    ; dati comuni
    soe.rid          = pRid
    soe.idn          = pIdn

    wSoe.rid         = pRid
    wSoe.idn         = pIdn
    wSoe.element     = 0x40       ; DATA

    ; lettura status
    soe.element      = 0x01              ; STATUS
    soe.datatype     = 0                 ; se lo ricerca da solo
    soe.datalen      = sizeof(idn_sts) ; dimensione destinazione in b
    soe.phase        = -1                ; bloccante
    esitoSoe = soe_ridn(soe, idn_sts)
    if((esitoSoe < 0) or blocca)
        stopExec = TRUE
        wait(NOT stopExec)
    endif

    ; lettura NAME
    soe.element      = 0x02              ; NAME
    soe.datatype     = 0                 ; se lo ricerca da solo
    soe.datalen      = sizeof(idn_name); dimensione destinazione in byt
    soe.phase        = -1                ; bloccante
    esitoSoe = soe_ridn(soe, idn_name)
    if((esitosoe < 0) or blocca)
        stopExec = TRUE
        wait(NOT stopExec)
    endif

    ; lettura ATTR
    soe.element      = 0x04              ; ATTR
    soe.datatype     = 0                 ; se lo ricerca da solo
    soe.datalen      = sizeof(idn_attr); dimensione destinazione in byt
    soe.phase        = -1                ; bloccante
   esitoSoe = soe_ridn(soe, idn_attr)
    if((esitosoe < 0) or blocca)
        stopExec = TRUE
        wait(NOT stopExec)
    endif

    ; lettura UNIT
    soe.element      = 0x08              ; UNIT
    soe.datatype     = 0                 ; se lo ricerca da solo
    soe.datalen      = sizeof(idn_unit); dimensione destinazione in byt
    soe.phase        = -1                ; bloccante
    esitoSoe = soe_ridn(soe, idn_unit)
```

© 2025 Robox SpA

```
    if((esitosoe < 0) or blocca)
        stopExec = TRUE
        wait(NOT stopExec)
    endif

    ; lettura MIN
    soe.element     = 0x10              ; MIN
    soe.datatype    = 0                 ; se lo ricerca da solo
    soe.datalen     = sizeof(idn_min)   ; dimensione destinazione in byt
    soe.phase       = -1                ; bloccante
    esitoSoe = soe_ridn(soe, idn_min)
    if((esitosoe < 0) or blocca)
        stopExec = TRUE
        wait(NOT stopExec)
    endif

    ; lettura MAX
    soe.element     = 0x20              ; MAX
    soe.datatype    = 0                 ; se lo ricerca da solo
    soe.datalen     = sizeof(idn_max)   ; dimensione destinazione in byt
    soe.phase       = -1                ; bloccante
    esitoSoe = soe_ridn(soe, idn_max)
    if((esitosoe < 0) or blocca)
        stopExec = TRUE
        wait(NOT stopExec)
    endif

    ; lettura DEFAULT
    soe.element     = 0x80                ; MAX
    soe.datatype    = 0                   ; se lo ricerca da solo
    soe.datalen     = sizeof(idn_default) ; dimensione destinazione in
    soe.phase       = -1                  ; bloccante
    esitoSoe = soe_ridn(soe, idn_default)
    if((esitosoe < 0) or blocca)
        stopExec = TRUE
        wait(NOT stopExec)
    endif
endif

select(flgData)
    CASE 0      ; lettura DATA
    CASE 1      ; lettura DATA
        soe.element     = 0x40              ; DATA
        soe.datatype    = 0                 ; se lo ricerca da solo
        soe.datalen     = sizeof(idn_data)  ; dimensione destinazione in
        soe.phase       = -1                ; bloccante

        esitoSoe = soe_ridn(soe, idn_data)
        if((esitosoe < 0) or (esitosoe > 2) or blocca)
            stopExec = TRUE
            wait(NOT stopExec)
```

© 2025 Robox SpA

```
        endif
        break

    CASE 2      ; lettura data come octet_string
        soe.element     = 0x40                  ; DATA
        soe.datatype    = 10                    ; DT_OCTET_STRING
        soe.datalen     = sizeof(idn_data_blob); dimensione destinazio
        soe.phase       = -1                    ; bloccante

        esitoSoe = soe_ridn(soe, idn_data_blob)
        if((esitosoe < 0) or (esitosoe > 2) or blocca)
            stopExec = TRUE
            wait(NOT stopExec)
        endif
        break

    CASE 3      ; lettura data come -VISIBLE_STRING
        soe.element     = 0x40                      ; DATA
        soe.datatype    = -9                        ; -VISIBLE_STRING
        soe.datalen     = sizeof(idn_data_string); dimensione destinaz
        soe.phase       = -1                        ; bloccante

        esitoSoe = soe_ridn(soe, idn_data_string)
        if((esitosoe < 0) or (esitosoe > 2) or blocca)
            stopExec = TRUE
            wait(NOT stopExec)
        endif
        break

    CASE 4      ; lettura data come array di word
        soe.element     = 0x40                  ; DATA
        soe.datatype    = -3                    ; -Integer16
        soe.datalen     = sizeof(idn_data_I16); dimensione destinazion
        soe.phase       = -1                    ; bloccante

        esitoSoe = soe_ridn(soe, idn_data_I16)
        if((esitosoe < 0) or (esitosoe > 2) or blocca)
            stopExec = TRUE
            wait(NOT stopExec)
        endif
        break

    CASE 5      ; lettura data come array di dword
        soe.element     = 0x40                  ; DATA
        soe.datatype    = -4                    ; -Integer32
        soe.datalen     = sizeof(idn_data_I32); dimensione destinazion
        soe.phase       = -1                    ; bloccante

        esitoSoe = soe_ridn(soe, idn_data_I32)
        if((esitosoe < 0) or (esitosoe > 2) or blocca)
            stopExec = TRUE
```

© 2025 Robox SpA

```
            wait(NOT stopExec)
        endif
        break

end_select

select(wFlgData)
    CASE 1      ; scrittura da wIdn_data
        wSoe.datatype   = 0         ; se lo ricerca da solo

    CASE -1     ; scrittura da wIdn_data con data type  imposta da she
        ; wSoe.datalen  = sizeof(idn_sts); dimensione destinazione in
        wSoe.cmd = 0
        wSoe.phase      = -1        ; bloccante
        wEsitoSoe = soe_widn(wSoe, wIdn_data)
        if((wesitosoe < 0) OR blocca)
            stopExec = TRUE
            wait(NOT stopExec)
        endif
        wFlgData = 0
        break

    CASE 2      ; scrittura da octet string
        wSoe.datatype   = 0         ; se lo ricerca da solo

    CASE -2     ; scrittura da wIdn_data con data type  imposta da she
        ; wSoe.datalen  = sizeof(idn_sts); dimensione destinazione in
        wSoe.phase      = -1        ; bloccante
        wSoe.cmd = 0
   ;   wEsitoSoe = SOE_WIDN (wSoe, wIdn_data_blob))
        if((wesitosoe < 0) OR blocca)
            stopExec = TRUE
            wait(NOT stopExec)
        endif
        wFlgData = 0
        break

    CASE 3      ; scrittura da -visible_string
        wSoe.datatype   = 0         ; se lo ricerca da solo

    CASE -3     ; scrittura da wIdn_data con data type  imposta da she
        ; wSoe.datalen  = SIZEOF (idn_sts); dimensione destinazione in
        wSoe.cmd = 0
        wSoe.phase      = -1        ; bloccante
        wEsitoSoe = soe_widn(wSoe, wIdn_data_string))
        if((wesitosoe < 0) OR blocca)
            stopExec = TRUE
            wait(NOT stopExec)
        endif
        wFlgData = 0
        break
```

© 2025 Robox SpA

```
    CASE 4      ; scrittura da array di word
        wSoe.datatype   = 0         ; se lo ricerca da solo

    CASE -4     ; scrittura da wIdn_data con data type  imposta da she
       ; wSoe.datalen  = SIZEOF (idn_sts); dimensione destinazione in
        wSoe.cmd = 0
        wSoe.phase      = -1        ; bloccante
        wEsitoSoe = soe_widn(wSoe, wIdn_data_I16))
        if((wesitosoe < 0) OR blocca)
            stopExec = TRUE
            wait(NOT stopExec)
        endif
        wFlgData = 0
        break

    CASE 5      ; scrittura da array di dword
        wFlgData = 0

    CASE -5     ; scrittura da wIdn_data con data type  imposta da she
        ; wSoe.datalen  = SIZEOF (idn_sts); dimensione destinazione in
        wSoe.cmd = 0
        wSoe.phase      = -1        ; bloccante
        wEsitoSoe = soe_widn(wSoe, wIdn_data_I32))
        if((wesitosoe < 0) OR blocca)
            stopExec = TRUE
            wait(NOT stopExec)
        endif
        wFlgData = 0
        break

    CASE 6      ; scrittura di comando
        wSoe.datatype = 2       ; data type comando dt_interger16
        wSoe.datalen  = 2
        wSoe.phase  = -1        ; bloccante
    ;   wSoe.cmd   scritto da shell
        wEsitoSoe = soe_widn(wSoe, wIdn_data)
        if((wesitosoe < 0) OR blocca)
            stopExec = TRUE
            wait(NOT stopExec)
        endif
        wFlgData = 0
        break

end_select

dwell 704(tdwell)

END_MAIN_LOOP
```

© 2025 Robox SpA

Funzione soe_widn()

Funzione che permette di scrivere un parametro SoE (scrittura del parametro *stru*.idn sul dispositivo identificato da *stru*.rid e informazioni prese in *source*).

La scrittura può avvenire in modo passante o in modo bloccante (utilizzabile solo nei task 1-:-8).

Per modo passante si intende che la funzione deve essere continuamente chiamata e questa rende lo stato attuale dell'operazione.

Per modo bloccante si intende che la funzione viene invocata e ritorna solo quando ha completato l'operazione di scrittura.

La funzione rende un esito sullo stato dell'esecuzione.

NOTA: Per scrivere comandi alle procedure SoE, impostare *stru*.cmd al valore interessato (vedi direttiva soe_widn).

Sintassi	**i32 soe_widn (stru_soe_idn stru, source)**
stru	Struttura di tipo stru_soe_idn 909 contenente le informazione per la scrittura di un parametro SoE.
source	Contiene il valore da scrivere. *source* deve essere un oggetto compatibile con i dati che si stanno scrivendo (se si scrive una VISIBLE STRING, *source* sarà di tipo string, se si scrive un OCTET_STRING sarà un array di byte o una struttura). In caso di scrittura di array verranno gestite solo informazioni la cui lunghezza è <= a 256 byte. Dovendo scrivere informazioni di maggior dimensione utilizzare come tipo dati OCTET_STRING e fornire un buffer di *souce* sufficientemente grande (ovviamente l'interpretazione dei dati è a carico dell'applicativo).
Risultato	Variabile contenente il risultato. Vedi gestione passante e/o bloccante.
Validità	Rule, Task.
Requisiti di sistema	RTE 34.18.3
Note	

Gestione passante

- Imporre ***stru*.phase = 1** per inizializzare la scrittura.
- Se non specificato il tipo di dato (*stru*.dataType=0), viene eseguita la lettura del tipo di dato su dispositivo remoto. La variabile *stru*.phase viene cambiata.
  La funzione rende 0.
- Il comando di scrittura viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.
  La funzione rende 0.

© 2025 Robox SpA

- Attesa della risposta.
  La funzione continua a rendere 0.
- Ricezione risposta con esito **positivo**. RTE cambia *stru*.phase = 4.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -2, gestore comunicazione occupato
  - -3, crash gestore comunicazione (riavvio)
  - -5, comando a un dispositivo non presente (*stru*.rid)
  - -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata
  - -12, valore illegale di idn (allarme 9900 arg 12)
  - -13, valore illegale di element/cmd (allarme 9900 arg 13)
  - -14, valore illegale di datatype (allarme 9900 arg 4/arg 14)
  - -15, valore illegale di datalen (allarme 9900 arg 5/arg 15)
  - -16, valore illegale di phase (allarme 9900 arg 6)
  - -21, destinazione non specificata (allarme 9900 arg 16)
  - -22, destinazione cambiata (allarme 9900 arg 32)
  - -23, errore software
  - 2, comando di cui si è già ricevuto risposta positiva, valore disponibile in *destination*
  - >2, codice di errore SDO, vedi tabella sotto

Gestione bloccante

- Imporre ***stru*.phase = -1** per eseguire la scrittura.
- Se non specificato il tipo di dato (*stru*.dataType=0), viene eseguita la lettura del tipo di dato su dispositivo remoto. La variabile *stru*.phase viene cambiata.
- Il comando di scrittura viene inviato al dispositivo remoto e si attende risposta. La variabile *stru*.phase viene cambiata.
- Attesa della risposta. Se la risposta non arriva entro un massimo di 3 secondi la funzione esce con un errore (timeout, rende -2).
- Ricezione risposta con esito **positivo**. RTE cambia *stru*.phase = 4.
  La funzione rende 1.
- Ricezione risposta con esito **negativo**. RTE cambia *stru*.phase = 0.
  La funzione rende codice di errore:
  - -2, gestore comunicazione occupato
  - -3, crash gestore comunicazione (riavvio)
  - -4, gestione bloccante non permessa nel task corrente (*stru*.phase = 1) (rule, ladder). Comando non accettato
  - -5, comando a un dispositivo non presente (*stru*.rid)
  - -6, nessuna operazione in esecuzione. *stru*.phase non inizializzata
  - -12, valore illegale di idn (allarme 9900 arg 12)

© 2025 Robox SpA

- -13, valore illegale di element/cmd (allarme 9900 arg 13)
- -14, valore illegale di datatype (allarme 9900 arg 4/arg 14)
- -15, valore illegale di datalen (allarme 9900 arg 5/arg 15)
- -16, valore illegale di phase (allarme 9900 arg 6)
- -21, destinazione non specificata (allarme 9900 arg 16)
- -22, destinazione cambiata (allarme 9900 arg 32)
- -23, errore software
- 2, comando di cui si è già ricevuto risposta positiva, valore disponibile in *destination*
- >2, codice di errore SDO, vedi tabella sotto

Codici di errore SDO SoE

SDO error code	Descrizione
0x0009	Accesso non valido all'elemento 0.
0x1001	Non esiste.
0x1009	Accesso non valido all'elemento 1.
0x2001	Nome inesistente.
0x2002	Nome sottodimensionato in trasmissione.
0x2003	Nome sovradimensionato in trasmissione.
0x2004	Nome non si può cambiare.
0x2005	Nome attualmente protetto da scrittura.
0x3002	Attributo sottodimensionato in trasmissione.
0x3003	Attributo sovradimensionato in trasmissione.
0x3004	Attributo non si può cambiare.
0x3005	Attributo attualmente protetto da scrittura.
0x4001	Unità inesistente.
0x4002	Unità sottodimensionato in trasmissione.
0x4003	Unità sovradimensionato in trasmissione.
0x4004	Unità non si può cambiare.
0x4005	Unità attualmente protetto da scrittura.
0x5001	Valore minimo in input inesistente.
0x5002	Valore minimo in input sottodimensionato in trasmissione.

© 2025 Robox SpA

SDO error code	Descrizione
0x5003	Valore minimo in input sovradimensionato in trasmissione.
0x5004	Valore minimo in input non si può cambiare.
0x5005	Valore minimo in input attualmente protetto da scrittura.
0x6001	Valore massimo in input inesistente.
0x6002	Valore massimo in input sottodimensionato in trasmissione.
0x6003	Valore massimo in input sovradimensionato in trasmissione.
0x6004	Valore massimo in input non si può cambiare.
0x6005	Valore massimo in input attualmente protetto da scrittura.
0x7001	Elemento inesistente.
0x7002	Elemento sottodimensionato in trasmissione.
0x7003	Elemento sovradimensionato in trasmissione.
0x7004	Elemento non si può cambiare.
0x7005	Elemento attualmente protetto da scrittura.
0x7006	Elemento inferiore al limite di *minimum input value.*
0x7007	Elemento superiore al limite di *maximum input value.*
0x7008	Elemento non corretto.
0x7009	Elemento protetto da password.
0x700A	Elemento temporaneamente non modificabile (AT o MDT).
0x700B	Indirettamente non valido.
0x700C	Elemento temporaneamente non modificabile (parametro o modo operativo).
0x7010	Comando già attivo.
0x7011	Comando non interrompibile.
0x7012	Comando non disponibile (in questa fase).

© 2025 Robox SpA

SDO error code	Descrizione
0x7013	Comando non disponibile (parametro non valido).

Codici di errore SDO SoE/CoE

SDO error code	Descrizione
0x0500 0000	Trovato errore generale protocollo SDO.
0x0503 0000	Toggle bit non alternato.
0x0504 0000	Timeout del protocollo SDO.
0x0504 0001	Identificatore del comando client/server non valido o sconosciuto.
0x0504 0002	Dimensione del blocco non valida (solo gestione bloccante).
0x0504 0003	Numero di sequenza non valido (solo gestione bloccante).
0x0504 0004	Errore CRC (solo gestione bloccante).
0x0504 0005	Eccesso di memoria.
0x0601 0000	Accesso non supportato a un oggetto.
0x0601 0001	Tentativo di leggere un oggetto a sola scrittura.
0x0601 0002	Tentativo di scrivere un oggetto a sola lettura.
0x0602 0000	L'oggetto non esiste nel dizionario oggetti.
0x0604 0041	Impossibile mappare l'oggetto sul PDO.
0x0604 0042	Il numero e la lunghezza degli oggetti da mappare supererebbero la lunghezza del PDO.
0x0604 0043	Incompatibilità dei parametri.
0x0604 0047	Incompatibilità interna al dispositivo.
0x0606 0000	Accesso non riuscito a causa di un errore hardware.
0x0607 0010	Il tipo di dati non corrisponde, la lunghezza del parametro di servizio non corrisponde.
0x0607 0012	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo alto.

© 2025 Robox SpA

SDO error code	Descrizione
0x0607 0013	Il tipo di dati non corrisponde, il parametro della durata del servizio è troppo basso.
0x0609 0011	Il sottoindice non esiste.
0x0609 0030	Superamento del campo di valori del parametro (solo per accesso in scrittura).
0x0609 0031	Valore del parametro scritto troppo alto.
0x0609 0032	Valore del parametro scritto troppo basso.
0x0609 0036	Il valore massimo è inferiore al valore minimo.
0x060A 0023	Risorsa non disponibile: connessione SDO.
0x0800 0000	Errore generale.
0x0800 0020	I dati non possono essere trasferiti o archiviati nell'applicazione.
0x0800 0021	I dati non possono essere trasferiti o archiviati nell'applicazione a causa del controllo locale.
0x0800 0022	I dati non possono essere trasferiti o archiviati nell'applicazione a causa dello stato attuale del dispositivo.
0x0800 0023	La generazione dinamica del dizionario oggetti non riesce o non è presente alcun dizionario oggetti (ad esempio, il dizionario oggetti viene generato da un file e la generazione non riesce a causa di un errore di file).
0x0800 0024	Nessun dato disponibile.
Altri valori	Codice non valido - non è presente CiA DS301 v4.1.

Esempi

Vedi esempio per funzione soe_ridn603().

Funzione sqrt()

Funzione che calcola la radice quadrata del valore specificato.

Sintassi	**real sqrt (real arg)**
arg	Valore sul quale eseguire l'operazione di radice.

© 2025 Robox SpA

Sintassi	**real sqrt (real arg)**
Risultato	Rende la radice quadrata di *arg*.
Validità	Rule, Task.
Note	

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se argomento < 0 (dove x =numero di task e xxx numero di step)

Esempi

Esempio di utilizzo:

```
ipotenusa = sqrt(cateto1*cateto1 + cateto2*cateto2)
```

Funzione statistics()

Funzione che permette di avere informazioni statistiche relative ad una variabile.

Sintassi	**real statistics (real value, stru_stat stru)**
value	Variabile sulla quale eseguire le statistiche.
stru	Struttura di tipo stru_stat dove verranno restituite le statistiche.
Risultato	Rende un valore contenente la media.
Validità	Rule, Task.
Note	Bisogna inizializzare ncount con il numero di campioni sul quale eseguire la media.

Esempi

Esempio di utilizzo:

```
real reso, T1
stru_stat statT1
statT1.ncount = 1000    ; esegui la media ogni 1000 cicli


__main_loop__
    ...
    T1 = loop_time()
    reso = statistics(T1, statT1)
end_main_loop
```

© 2025 Robox SpA

Funzione store_nvr_nopf()

Funzione che permette di aggiornare i dati di registri nvr anche in memoria statica.

Questa funzione serve solo per ovviare a possibili problemi di mancata esecuzione del salvataggio di tutti gli nvr allo spegnimento.

È quindi possibile aggiornare in memoria ritentiva runtime e avere i dati aggiornati all'accensione con allarme 995 anzichè avere i dati relativi all'ultimo spegnimento valido.

ATTENZIONE: Deve essere usata con parsimonia (spazio e frequenza) e solo in applicazioni particolari.

Sintassi	**i32 store_nvr_nopf (i32 idNVR, i32 nNVR)**
idNVR	Indice del primo registro nvr da salvare.
nNVR	Numero di registri nvr da salvare a partire da *idNVR*.
Risultato	Rende:

Valore	Significato
0	Operazione eseguita con successo.
-1	Parametri illegali (*idNVR* o *nNVR* non validi)
-2	Memoria statica non inizializzata.
-3	Errore interno.

Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; salvo i registri nvr dal 10 al 26 ogni 10 secondi
real time_1
int result_store_nvr
string msg

time_1 = tfb

__main_loop__
    ...
    if(tfb - time_1 > 10)
        time_1 = tfb
        result_store_nvr = stor_nvr_nopf(10, 16)
        if(result_store_nvr < 0)
            str_format(msg, "Store NVR failed with code %i", result_
            inf_report(0x100, msg)
```

© 2025 Robox SpA

```
        endif
    endif
end_main_loop
```

Funzione store_nvrr_nopf()

Funzione che permette di aggiornare i dati di registri nvrr anche in memoria statica.

Questa funzione serve solo per ovviare a possibili problemi di mancata esecuzione del salvataggio di tutti gli nvrr allo spegnimento.

È quindi possibile aggiornare in memoria ritentiva runtime e avere i dati aggiornati all'accensione con allarme 995 anzichè avere i dati relativi all'ultimo spegnimento valido.

ATTENZIONE: Deve essere usata con parsimonia (spazio e frequenza) e solo in applicazioni particolari.

<table>
<tr><td>Sintassi</td><td>i32 store_nvrr_nopf (i32 idNVRR, i32 nNVRR)</td></tr>
<tr><td>idNVRR</td><td>Indice del primo registro nvrr da salvare.</td></tr>
<tr><td>nNVRR</td><td>Numero di registri nvrr da salvare a partire da idNVRR.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>0</td><td>Operazione eseguita con successo.</td></tr>
<tr><td>-1</td><td>Parametri illegali (idNVRR o nNVRR non validi)</td></tr>
<tr><td>-2</td><td>Memoria statica non inizializzata.</td></tr>
<tr><td>-3</td><td>Errore interno.</td></tr>
</table></td></tr>
<tr><td>Validità</td><td>Rule, Task.</td></tr>
<tr><td>Note</td><td></td></tr>
</table>

Esempi

Esempio di utilizzo:

```
; salvo i registri nvrr dal 10 al 26 ogni 10 secondi
real time_1
int result_store_nvrr
string msg

time_1 = tfb
```

© 2025 Robox SpA

```
__main_loop__
    ...
    if(tfb - time_1 > 10)
        time_1 = tfb
        result_store_nvrr = stor_nvrr_nopf(10, 16)
        if(result_store_nvrr < 0)
            str_format(msg, "Store NVRR failed with code %i", result
            inf_report(0x100, msg)
        endif
    endif
end_main_loop
```

Funzione str_asc()

Funzione che permette di estrapolare un singolo carattere da una variabile stringa.

Il corrispondente codice ascii viene reso come valore intero.

Sintassi	**i32 str_asc (string str, i32 nPos)**
str	Individua la stringa interessata.
nPos	Individua la posizione del carattere all'interno della stringa.
Risultato	Rende il codice ascii corrispondente al carattere estrapolato.
Validità	Rule, Task.
Note	Se *nPos* punta ad un carattere fuori dalla stringa (o la stringa è vuota), il valore reso sarà 0.

Esempi

Esempio di utilizzo:

```
int val
string esa_digit = "0123456789ABCDEF"

val = str_asc(esa_digit, 2)
; il contenuto di val sarà 31H (49 dec.)
val = str_asc(esa_digit, 11)
; il contenuto di val sarà 41H (65 dec.)
```

© 2025 Robox SpA

Funzione str_chr()

Funzione che converte un valore numerico intero in un carattere ascii.

Sintassi	**string str_chr (i32 value)**
value	È un'espressione di tipo intero rappresentante il valore da convertire.
Risultato	Rende la stringa contenente il carattere corrispondente al codice ascii *value*.
Validità	Rule, Task.
Note	La conversione avviene rispettando le seguenti regole: • la variabile viene trattata in modo conforme al trattamento delle espressioni tipo intero • la variabile intera ottenuta viene convertita dal formato decimale al formato esadecimale • solamente le ultime due cifre esadecimali vengono prese in considerazione

ATTENZIONE: Occorre prestare attenzione al fatto che questa funzione può anche restituire il carattere ascii NULL (00), che viene interpretato come terminatore di stringa dal linguaggio R3.

Esempi

Esempio di utilizzo:

```
string1 = str_chr(31H)
; dopo l'esecuzione dell'istruzione il contenuto della variabile local
sr(4) = str_chr(65)
; dopo l'esecuzione dell'istruzione il contenuto di sr (4) sarà : "A"
sr(4) = str_chr(97)
; dopo l'esecuzione dell'istruzione il contenuto di sr (4) sarà : "a"
sr(5) = str_chr(1074.5)
; dopo l'esecuzione dell'istruzione il contenuto di sr (5) sarà : "2"
```

Funzione str_compact()

Funzione che sostituisce tutte le sequenze di spazi in una stringa con un solo carattere spazio.

Sintassi	**string str_compact (string str)**
str	È la stringa da compattare.
Risultato	Rende la stringa compattata.
Validità	Rule, Task.
Note	

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

```
sr(4) = str_compact(" questa è          una stringa          lunga ")
; dopo l'esecuzione dell'istruzione il contenuto di sr(4) sarà :
; " questa è una stringa lunga "
sr(5) = str_compact(sr(4))
; se la stringa non ha sequenze di spazi multipli, la funzione non int
; alcuna modifica, per cui si avrà nell'esempio sopra riportato che:
sr(5) = sr(4)
```

Funzione str_equal()

Funzione che esegue un confronto di uguaglianza tra stringhe.

Sintassi	**i32 str_equal (string str1, string str2)**
str1	Prima stringa da confrontare.
str2	Seconda stringa da confrontare.
Risultato	Rende: Valore: <> 0 — Significato: Le due stringhe sono identiche (tiene conto sia di caratteri, che di spaziature e lunghezze). Valore: 0 — Significato: Le stringhe sono diverse.
Validità	Rule, Task.
Note	La funzione non è case sensitive.

Esempi

Esempio di utilizzo:

```
sr(4) = "esempio"
if(str_equal(sr(4), "EsEmPiO"))
    ; l'esito del confronto sarà TRUE

if(str_equal(sr(4), "ESEmpio "))
    ; l'esito del confronto sarà FALSE in quanto la stringa sr(4) è pi
```

© 2025 Robox SpA

Funzione str_find()

Funzione che rende la posizione all'interno di una stringa dove comincia la sottostringa da cercare.

Sintassi	**i32 str_find (string strSrc, string strKey)**
strSrc	Stringa in cui cercare.
strKey	Stringa da cercare.
Risultato	Rende la posizione del carattere a cui comincia *strKey*.
Validità	Rule, Task.
Nota	Se la sottostringa comincia al primo carattere rende 1. Se la sottostringa non esiste rende 0. La ricerca è case sensitive.

Esempi

Esempio di utilizzo:

```
sr(1) = "va tutto bene"
r(10) = str_find(sr(1), "BENE")
; in questo caso r(10) = 0
sr(1) = "va tutto bene"
r(10) = str_find(sr(1), "bene")
; in questo caso r(10) = 10
```

Funzione str_format()

Funzione che esegue la formattazione di una stringa in base alla descrizione fornita dall'utente.

Sintassi	**i32 str_format (string buff, desc [, val1, val2, val3, ...])**
buff	Variabile contenente la stringa formattata.
desc	Costante stringa contenente la descrizione del formato da utilizzare durante la composizione di *buff*. **NOTA**: Si faccia riferimento alla descrizione della funzione C (standard ANSI) **sprintf** per i dettagli e il formalismo utilizzabile.

Codice	Descrizione
%d	Variabile intera signed - Rappresentazione decimale.

© 2025 Robox SpA

Sintassi	**i32 str_format (string buff, desc [, val1, val2, val3, ...])**
	<table><tr><th>Codice</th><th>Descrizione</th></tr><tr><td>%u</td><td>Variabile intera unsigned - Rappresentazione decimale.</td></tr><tr><td>%x</td><td>Variabile intera - Rappresentazione esadecimale.</td></tr><tr><td>%f</td><td>Variabile reale.</td></tr><tr><td>%g</td><td>Variabile reale.</td></tr><tr><td>%s</td><td>Variabile stringa - Solo per allarmi di sistema.</td></tr></table>
val1, val2, val3, ...	Parametri opzionali, costituiti da generiche espressioni intere, reali o stringa. Devono corrispondere esattamente al formalismo e al numero di parametri presenti nella stringa *desc*.
Risultato	Rende il numero di byte scritti nella stringa *buff*.
Validità	Rule, Task.
Note	È responsabilità dell'utilizzatore la corretta corrispondenza, in numero e tipo, dei parametri opzionali passati alla funzione str_format(). Occorre inoltre assicurarsi che la formattazione della stringa *buff* non richieda la scrittura di un numero di caratteri superiore alla dimensione dichiarata. Il compilatore non è in grado di identificare e segnalare eventuali incoerenze. Questa funzione va pertanto utilizzata con attenzione particolare, potendo causare blocchi di sistema o altri malfunzionamenti runtime.

Esempi

Esempio di utilizzo:

```
int scritti
string data[60]
string mese = "febbraio"  ; dichiaro variabile mese di tipo STRING e l
int anno = 2005    ; dichiaro variabile anno di tipo intero e la inizi

r(20) = 5
scritti = str_format(data, "Oggi è il %d %s %4d", r(20), mese, anno)
; la stringa data contiene: "Oggi è il 5 febbraio 2005"
```

© 2025 Robox SpA

Funzione str_i()

Funzione che converte un valore numerico intero nella sua rappresentazione equivalente ascii.

Sintassi	**string str_i (i32 value, i32 minLen, i32 form)**
value	Valore da convertire.
minLen	Lunghezza minima della stringa resa. È tipicamente un numero, ma può anche essere un'espressione. Se il numero di caratteri che rappresentano la stringa è inferiore a *minLen*, essa verrà inizializzata con un numero opportuno di caratteri spazio. Nel caso che *minLen* non sia sufficiente a rappresentare il numero, la stringa conterrà un numero di caratteri maggiore di *minLen*, cioè i caratteri strettamente necessari a rappresentare il numero *value*. Non si ha segnalazione di errore.
form	Parametro usato per specificare se si desidera rappresentare il numero *value* come quantità intera espressa in formato decimale oppure in formato esadecimale. I valori da attribuire a *form* possono essere: • 10, rappresentazione decimale • 16, rappresentazione esadecimale con indicazione della base • -16, rappresentazione esadecimale senza indicazione della base Qualsiasi valore di *form* diverso da 10 o 16 implica l'utilizzo della rappresentazione decimale, senza segnalazione di errore.
Risultato	Rende la rappresentazione ascii del numero *value*.
Validità	Rule, Task.
Note	

Se il numero da convertire è reale, esso viene preventivamente trasformato e poi convertito.

Esempi

Esempio di utilizzo:

```
sr(4) = str_i(123, 3, 10)
; dopo l'esecuzione dell'istruzione il contenuto di sr (4) sarà : "123
string1 = str_i(12, 3, 16)
```

© 2025 Robox SpA

```
; dopo l'esecuzione dell'istruzione il contenuto della variabile local
sr(5) = str_i(rr(12), 4, 10)
; se nel registro reale rr(12) è contenuto K_PI, il contenuto del re
```

Funzione str_insert()

Funzione che inserisce in una stringa di destinazione la sottostringa passata, a partire dalla posizione specificata.

Sintassi	**i32 str_insert (string strDst, string strKey, i32 pos)**
strDst	Stringa in cui viene inserita *strKey*.
strKey	Stringa che viene inserita in *strDst*.
pos	Indice della posizione di inserimento. Se *pos* = 1, inserisce nella prima posizione. Se *pos* è fuori dalle dimensioni di *strDst*, l'inserimento viene limitato al primo o ultimo carattere.
Risultato	Rende: Valore: <> 0 — Significato: Operazione eseguita con successo. Valore: 0 — Significato: Altrimenti.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
int reso
sr(1)= "Questo è l'esempio numero"
reso = str_insert(sr(1), " uno!!!!", 26)
; in sr(1) avrò: "Questo è l'esempio numero uno!!!!"

int reso2
sr(1)= "Questo è l'esempio numero"
reso2 = str_insert(sr(2), " due!!!!", 8)
; in sr(2) avrò: "Questo  due!!!è l'esempio numero"

if(str_insert(sr(1), "Esempio Nuovo", 25)
    write_dis("Come sono contento! Ho inserito")
endif
```

© 2025 Robox SpA

Funzione str_len()

Funzione che restituisce il numero di caratteri che compongono la stringa specificata.

Sintassi	**i32 str_len (string vs)**
str	È la stringa di cui viene calcolata la lunghezza.
Risultato	Rende il numero di byte che compongono la stringa.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
r(128) = str_len("questa è una stringa")
; dopo l'esecuzione dell'istruzione il contenuto di r(128) sarà : 20

string1 = "esempio di stringa"
if(str_len(string1) = 18)
    ; in questo caso l'esito del confronto è TRUE
endif
```

Funzione str_mid()

Funzione che restituisce la porzione specificata di una stringa.

Sintassi	**string str_mid (string strSrc, i32 pos, i32 nChar)**
strSrc	Stringa da cui deve essere prelevata la sottostringa.
pos	Posizione del carattere da cui inizia il prelievo. *pos* = 1 indica che il prelievo deve iniziare dal primo carattere. Se *pos* < 1 viene emesso un allarme e viene forzato il valore 1. Se *pos* è maggiore del numero di caratteri costituenti la stringa, viene generato un allarme e viene restituita una stringa vuota.
nChar	Numero di caratteri che devono essere prelevati. Se viene impostato un valore superiore alla lunghezza corrente della stringa, la sottostringa risultante sarà composta dai caratteri compresi fra *pos* e la fine della stringa.

© 2025 Robox SpA

Sintassi	**string str_mid (string strSrc, i32 pos, i32 nChar)**
Risultato	Rende una sottostringa contenente una porzione della stringa *strSrc*.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
sr(3) = str_mid("ciao a tutti ", 8, 5)
; dopo l'esecuzione dell'istruzione il contenuto di sr(3) sarà : "tutt
sr(2) = str_mid(sr(3), 3, 2)
; dopo l'esecuzione dell'istruzione il contenuto di sr(2) sarà : "tt"
```

Funzione str_r()

Funzione che converte un'espressione di tipo reale nella sua rappresentazione equivalente ascii.

Segno e punto decimale vengono trattati adeguatamente.

Sintassi	**string str_r (real value, i32 minLen, i32 nDec)**
value	Valore da convertire.
minLen	Lunghezza minima della stringa resa. È tipicamente un numero, ma può anche essere un'espressione. Se il numero di caratteri che rappresentano la stringa è inferiore a *minLen*, essa verrà inizializzata con un numero opportuno di caratteri spazio. Nel caso che *minLen* non sia sufficiente a rappresentare il numero, la stringa conterrà un numero di caratteri maggiore di *minLen*, cioè i caratteri strettamente necessari a rappresentare il numero *value*. Non si ha segnalazione di errore. Con *minLen* = 0 si impone il formato di visualizzazione a virgola mobile.
nDec	Rappresenta il numero di cifre decimali desiderato.
Risultato	Rende la rappresentazione equivalente ascii del numero *value*.
Validità	Rule, Task.
Note	Nel formato in virgola mobile non vengono aggiunti alla stringa gli zeri non significativi e

© 2025 Robox SpA

Sintassi	**string str_r (real value, i32 minLen, i32 nDec)**
	gli spazi di riempimento. Quindi la lunghezza della stringa può variare tra 1 e (*nDec* + 2). I parametri *minLen* e *nDec* devono essere specificati. Se *minLen* = 0, si assume una lunghezza minima di stringa pari a 10.

Esempi

Esempio di utilizzo:

```
sr(4) = str_r(123, 5, 0)
; dopo l'esecuzione dell'istruzione il contenuto di sr(4) sarà :" 123"

STRING1 = str_r(12/5, 6, 0)
; dopo l'esecuzione dell'istruzione il contenuto della variabile local
; " 2" (il numero viene arrotondato per difetto)

sr(5) = str_r(rr(12), 10, 5)
; se nel registro reale rr(12) è contenuto - K_PI, il contenuto del
; " -3.14160" (il numero viene arrotondato per eccesso)

sr(1) = str_r(1/3, 0, 5)
; dopo l'esecuzione sr(1) sarà: "0,33333"

sr(2) = str_r(1+4, 0, 8)
; dopo l'esecuzione sr(2) sarà: "5"
```

Funzione str_replace()

Funzione che effettua, all'interno di una stringa, la sostituzione di una sottostringa (se trovata) con un'altra specificata.

Sintassi	i32 str_replace (string strSrc, string strOld, string strNew)
strSrc	Variabile stringa su cui la funzione lavora.
strOld	Sottostringa da cercare.
strNew	Sottostringa da sostituire.
Risultato	Rende l'indice all'interno di *strSrc* del primo carattere dove è avvenuta la sostituzione. Se la sottostringa comincia dal primo carattere rende 1, se non esiste rende 0 e non effettua alcuna sostituzione.
Validità	Rule, Task.
Note	La ricerca tiene conto della differenza fra caratteri maiuscoli o minuscoli.

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

```
sr(1) = "Uno due quattro cinque"
if(str_relace633(sr(1), "quattro", "tre quattro")
    write_dis676 = "Successione sistemata"
endif
```

Funzione str_to_ipaddr()

Questa funzione converte da una stringa contenente un indirizzo IP a un indirizzo rappresentato come intero.

Sintassi	**i32 str_to_ipaddr (string sAddr, u32 addr)**
sAddr	Variabile contenente la stringa da convertire.
addr	Variabile in cui la funzione scrive il risultato.
Risultato	Rende: Valore: <> 0 — Significato: Operazione conclusa con successo. Valore: 0 — Significato: *sAddr* non è un indirizzo valido.
Validità	Rule, Task.
Note	

La stringa ha formato: "xxx.xxx.xxx.xxx".

L'intero è di tipo u32 con formato: u32 *addr* = 0xaabbccdd; aa, bb, cc e dd rappresentano le 4 componenti dell'indirizzo IP espresse in formato esadecimale.

Esempi

Esempio di utilizzo:

```
int reso
string indiSr
u32 indiNum
reso = str_to_ipaddr634(indiSr, indiNum)
; se indiSr = "192.168.1.10" -->indiNum= 0xc0a8010a
; se indiSr = "127.0.0.18" -->indiNum= 0x7f000012
```

© 2025 Robox SpA

Funzione str_val()

Funzione che rende, in formato numerico, un valore contenuto in una stringa.

Sintassi	**real str_val (string strSrc, i32 nVal [, i32 base=10] [, i32 code=NULL])**
strSrc	Stringa in cui cercare il valore numerico.
nVal	Specifica quale dato numerico, fra quelli che la stringa contiene, deve essere convertito (se *nVal*=1 sarà il primo dato numerico, se *nVal*=2 ci si riferirà al secondo, ...).
base	Specifica il formato desiderato per il valore convertito. (opzionale, default 10) Valore: 10 — Formato: Decimale Valore: 16 — Formato: Esadecimale
code	Rende un codice di esito per la funzione: (opzionale, default NULL) Valore: 0 — Descrizione: Operazione effettuata con successo. Valore: 1 — Descrizione: *nVal* illegale (<= 0). Valore: 2 — Descrizione: Dato non trovato in *strSrc*. Valore: 3 — Descrizione: Errore durante la conversione.
Risultato	Rende il valore numerico (base 10) specificato contenuto nella stringa.
Validità	Rule, Task.
Note	Qualora il valore specificato da *nVal* non corrisponda a nessun numero all'interno della stringa, il valore reso sarà 0 e si avrà una segnalazione di errore. Lo stesso comportamento si avrà se *nVal* < 1. Se, nella ricerca del numero da convertire verrà trovato un segno "+" oppure "-" non seguito da un numero, verrà generato *code* = 3 e ed il valore restituito sarà 0.

Esempi

Esempio di utilizzo:

```
sr(4) = "posizione asse 3 = 3000"
```

© 2025 Robox SpA

```
rr(32) = str_val(sr(4), 1)
rr(200) = str_val(sr(4), 2)
rr(48) = str_val(sr(4), 3)
; Dopo l' esecuzione delle istruzioni rr(32)=3, rr(200)=3000, rr(48)=0
```

Altri esempi:

Risultato	*nVal*	*base*	*code*	*strSrc*
16	1	10	0	0x10
16	1	16	0	0x10
10	1	10	0	0xA
10	1	16	0	0xA
0	1	16	2	Ah
0	1	16	2	hA
0	1	10	2	Ah
0	1	10	2	hA
123	1	17	0	123
123	1	5	0	123
291	1	16	0	123
2	1	10	0	Sono arrivati 2 atleti su 23 partecipanti. In media l' 8.69 %
23	2	10	0	Sono arrivati 2 atleti su 23 partecipanti. In media l' 8.69 %
8.69	3	10	0	Sono arrivati 2 atleti su 23 partecipanti. In media l' 8.69 %
0	4	10	2	Sono arrivati 2 atleti su 23 partecipanti. In media l' 8.69 %
3	1	10	0	3,45 ; errata la virgola
3.45	1	10	0	3.45
3	1	16	0	3.45 ; errata la base

© 2025 Robox SpA

Risultato	*nVal*	*base*	*code*	*strSrc*
3	1	16	0	3,45 ; errata la virgola e la base
0	1	10	3	+
0	1	10	3	-
0	1	16	0	+
0	1	16	0	-

Funzione sum8_mem()

Funzione che restituisce la somma di *n* elementi di un array.

Sintassi	**i32 sum8_mem (u8 buff, i32 nEl)**
buff	Array di u8.
nEl	Numero di elementi dell'array che si vogliono sommare a partire dalla prima posizione.
Risultato	Rende la somma dei primi *nEl* elementi dell'array.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
int reso
u8 array[5]
; array = [1, 2, 3, 4, 5]

reso = sum8_mem(array, 0)   ; reso = 0
reso = sum8_mem(array, 1)   ; reso = 1
reso = sum8_mem(array, 2)   ; reso = 3
reso = sum8_mem(array, 3)   ; reso = 6
reso = sum8_mem(array, 4)   ; reso = 10
reso = sum8_mem(array, 5)   ; reso = 15
reso = sum8_mem(array, 12)  ; reso = 15
```

© 2025 Robox SpA

Funzione sum16_mem()

Funzione che restituisce la somma di *n* elementi di un array.

Sintassi	**i32 sum16_mem (u16 buff, i32 nEl)**
buff	Array di u16.
nEl	Numero di elementi dell'array che si vogliono sommare a partire dalla prima posizione.
Risultato	Rende la somma dei primi *nEl* elementi dell'array.
Validità	Rule, Task.
Note	

Funzione sum32_mem()

Funzione che restituisce la somma di *n* elementi di un array.

Sintassi	**i32 sum32_mem (u32 buff, i32 nEl)**
buff	Array di u32.
nEl	Numero di elementi dell'array che si vogliono sommare a partire dalla prima posizione.
Risultato	Rende la somma dei primi *nEl* elementi dell'array.
Validità	Rule, Task.
Note	

Funzione sumr_mem()

Funzione che restituisce la somma di *n* elementi di un array.

Sintassi	**i32 sumr_mem (real buff, i32 nEl)**
buff	Array di real.
nEl	Numero di elementi dell'array che si vogliono sommare a partire dalla prima posizione.
Risultato	Rende la somma dei primi *nEl* elementi dell'array.
Validità	Rule, Task.
Note	

© 2025 Robox SpA

Funzione sw0()

Questa funzione si usa per leggere o scrivere (istruzione) la prima word (a partire da quella meno significativa) dell'argomento con estensione del segno.

Lettura:

Sintassi	**i16 sw0 (i16|i32|u16|u32|real arg)**
arg	Argomento della word. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende la prima word dell'argomento con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una word non esistente ritorna un errore.

Scrittura (istruzione):

Sintassi	**sw0 (i16|i32|u16|u32|real arg) = i16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
I32 pluto, ris1, ris2
pluto = 0x12345678
; se pluto vale 0x12345678
r(1) = sw0(pluto) ; r(1)= 0x5678
r(1) = sw1(pluto) ; r(1)= 0x1234
; se pluto vale 0x80121234
r(1) = sw0(pluto) ; r(1)= 0x00001234
r(1) = sw1(pluto) ; r(1)= 0xffff8012

; con RTE 33.9.0:
I32 pippo = 123
sw4(pippo) = 9  ; errore!
```

Funzione sw1()

Questa funzione si usa per leggere o scrivere (istruzione) la seconda word (a partire da quella meno significativa) dell'argomento con estensione del segno.

Lettura:

© 2025 Robox SpA

Sintassi	**i16 sw1 (i32|u32|real arg)**
arg	Argomento della word. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende la seconda word dell'argomento con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una word non esistente ritorna un errore.

Scrittura (istruzione):

Sintassi	**sw1 (i32|u32|real arg) = i16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Note	

Funzione sw2()

Questa funzione si usa per leggere o scrivere (istruzione) la terza word (a partire da quella meno significativa) dell'argomento con estensione del segno.

Lettura:

Sintassi	**i16 sw2 (real arg)**
arg	Argomento della word. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende la terza word dell'argomento con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una word non esistente ritorna un errore.

Scrittura (istruzione):

Sintassi	**sw2 (real arg) = i16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**sw2 (real arg) = i16 value**
Note	

Funzione sw3()

Questa funzione si usa per leggere o scrivere (istruzione) la quarta word (a partire da quella meno significativa) dell'argomento con estensione del segno.

Lettura:

Sintassi	**i16 sw3 (real arg)**
arg	Argomento della word. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende la quarta word dell'argomento con l'aggiunta del segno.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una word non esistente ritorna un errore.

Scrittura (istruzione):

Sintassi	**sw3 (real arg) = i16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Note	

Funzione tan()

Funzione che esegue la tangente di un valore specificato.

Sintassi	**real tan (real value)**
value	Argomento della tangente. **NOTA**: *value* deve essere in radianti.
Risultato	Rende la tangente di *value*.
Validità	Rule, Task.
Note	Se l'argomento è pari a K_PIMEZZI oppure a -K_PIMEZZI, il risultato sarà ± 1E308 e non verrà emessa una segnalazione di errore.

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

```
rr(5) = 0
rr(10) = tan(rr(5))                ; = 0
rr(10) = tan(K_PI/4)               ; = 1
```

Funzione time_micro()

Funzione che rende il valore del tempo, in microsecondi, trascorso dall'accensione del controllo assi Robox.

Sintassi	**real time_micro ()**
Risultato	Rende il tempo trascorso [us].
Validità	Rule, Task.
Note	Può essere utilizzata per misure di durata di porzioni di software, ...

Esempi

Esempio di utilizzo:

```
real calcTime

rr(100) = time_micro()
calcTime = time_micro() - rr(100)
```

Funzione tm_et()

Questa funzione si usa per leggere o impostare (istruzione) il valore corrente (Elapsed Time) del timer.

Lettura:

ATTENZIONE: Il valore è leggibile solo dopo che venga fatto partire il timer (es. dopo aver chiamato la funzione tm_int() per la prima volta), altrimenti viene generato l'allarme 9901.

Sintassi	**real tm_et (timer name)**
name	Nome del timer di tipo timer.
Risultato	Rende il valore, in millisecondi, di tempo corrente del timer.
Validità	Rule, Task.
Note	Tutti i timer, ad eccezione di quello integrale (tm_int()), la prima volta che vengono chiamati resettano il loro valore attuale.

Scrittura (istruzione):

© 2025 Robox SpA

ATTENZIONE: Il valore è impostabile solo prima che venga fatto partire il timer (es. prima di chiamare la funzione tm_int779() per la prima volta), altrimenti viene generato l'allarme 9901.

Sintassi	**tm_et (timer name, real valueEt)**
name	Nome del timer di tipo timer930.
valueEt	Valore del tempo, in millisecondi, da impostare.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; Task che utilizza il timer tm_int per misurare per quanto tempo il c
; Il tempo trascorso viene continuamente scritto in un registro non vo
; Ad ogni riaccensione, il timer ricarica il valore presente nell'nvrr

$task 1

timer930 timer1
tm_et642(timer1, nvrr(1))      ; nvrr(1) -> timer1.et
tm_int779(timer1, 1)           ; attivo il timer integrale

__MAIN_LOOP__
   nvrr(1) = tm_et(timer1) ; timer1.et -> nvrr(1)
END_MAIN_LOOP
```

Funzione tm_pt()

Questa funzione si usa per leggere o impostare (istruzione) il valore di preset del timer.

Lettura:

Sintassi	**real tm_pt (timer name)**
name	Nome del timer di tipo timer930.
Risultato	Rende il valore, in millisecondi, di preset del timer.
Validità	Rule, Task.
Note	

Scrittura (istruzione):

Sintassi	**tm_pt (timer name, real presetTm)**
name	Nome del timer di tipo timer930.

© 2025 Robox SpA

Sintassi	**tm_pt (timer name, real presetTm)**
presetTm	Valore del preset, in millisecondi, da impostare.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
tm_pt(T1, rr(1000))          ; scrivo il valore di preset contenuto nel
tm_pt(tm[idx], 1000)       ; scrivo il valore 1000 di preset nel timer
valorePreset = tm_tp(T1)   ; scrivo il valore del preset nella variabil
```

Funzione tm_pt2()

Questa funzione si usa per leggere o impostare (istruzione) il valore del secondo preset del timer quadrato.

Lettura:

Sintassi	**real tm_pt (timer name)**
name	Nome del timer di tipo timer.
Risultato	Rende il valore, in millisecondi, di preset del timer.
Validità	Rule, Task.
Note	

Scrittura (istruzione):

Sintassi	**tm_pt2 (timer name, real presetTm)**
name	Nome del timer di tipo timer.
presetTm	Valore del preset, in millisecondi, da impostare.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
tm_pt2(T1, rr(1000))          ; scrivo il valore di preset contenuto
valorePreset2 = tm_tp2(T1)   ; scrivo il valore del preset 2 nella vari
```

© 2025 Robox SpA

Funzione tm_q()

Funzione per leggere il valore di stato dell'uscita del timer.

Sintassi	**bool tm_q (timer name)**
name	Nome del timer di tipo timer 930.
Risultato	Rende il valore di stato dell'uscita del timer.
Validità	Rule, Task.
Note	

Funzione tobcd()

Funzione che converte un valore numerico intero, espresso in formato binario, in formato BCD (Binary-Coded Decimal).

Sintassi	**i32 tobcd (i32 value)**
value	Variabile rappresentante il carattere (0-:-255) o EOF (-1).
Risultato	Rende il valore convertito in formato BCD.
Validità	Rule, Task.
Note	Poiché il valore di uscita è codificato in 4 byte, il campo di variabilità del valore in uscita è: 0-:-99999999 sempre positivo.

La funzione opera come segue:

- L'espressione di tipo reale in entrata viene convertita in una quantità intera
- Viene eseguito il valore assoluto della quantità intera
- Se il valore è maggiore di 99999999, esso viene limitato a 99999999
- Il valore così ottenuto viene convertito in BCD

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se argomento illegale (argomento > 99999999)

Esempi

Esempio di utilizzo:

```
; Assumendo di voler scrivere sulla porta output_word 1 il numero 1234
out_w(1) = tobcd 645(1234)
```

© 2025 Robox SpA

Funzione tobin()

Funzione che converte un valore numerico intero, espresso in formato BCD (Binary-Coded Decimal), nel formato binario.

Sintassi	**i32 tobin (i32 value)**
value	Valore BCD che si vuole convertire.
Risultato	Rende il valore convertito in formato binario.
Validità	Rule, Task.
Note	Il formato binario è usato dagli operatori 803 del linguaggio R3.

Diagnostica

- Genera 9900 ill. Arg A:1 T:x ST:xxx se argomento illegale (una delle cifre è > 9)

Esempi

Esempio di utilizzo:

```
r(33) = tobin 646 (inp_w(1))
; I 16 bit della porta input_word 1 vengono interpretati come 4 numeri
; Nell'esempio sotto riportato, nel quale si è immaginato di voler leg
; codificata BCD il cui range è 0 - 999 (12 bit) e si immagini altresì
r(33) = inp_w(1)
; a valle di questa istruzione i 16 bit della porta di entrata 1 vengo
if(r(33) r_and 806 0X8000)
    ; verifica DATA_VALID: se DATA_VALID presente leggi codice e conve
    r(34) = r(33) r_and 806 0X0FFFH
    ; estrai i 12 bit meno significativi
    r(35) = tobin(r(34))
    ; converti in binario puro
endif
; dopo l'esecuzione delle istruzioni, supponendo di aver letto il valo
```

Registro	**R(33)**	**R(34)**	**R(35)**
Esadecimale	8634	634	27A
Decimale	-31180	1588	634

Funzione todeg()

Funzione che converte un valore numerico da radianti a gradi.

Sintassi	**real todeg (real value)**
value	Rappresenta il valore, espresso in radianti, che si vuole convertire.

© 2025 Robox SpA

Sintassi	**real todeg (real value)**
Risultato	Rende il valore convertito in gradi.
Validità	Rule, Task.
Note	Tutte le funzioni trigonometriche operano su argomenti espressi in radianti.

Esempi

Esempio di utilizzo:

```
angle = todeg(K_PI / 6)
; dopo l'esecuzione dell'istruzione, il contenuto della variabile loca
```

Funzione tolower()

Funzione che converte un carattere in minuscolo.

Sintassi	**i32 tolower (i32 ch)**
ch	Carattere da convertire.
Risultato	Rende il carattere convertito.
Validità	Rule, Task.
Note	

Funzione torad()

Funzione che converte un valore numerico da gradi a radianti.

Sintassi	**real torad (real value)**
value	Rappresenta il valore, espresso in gradi, che si vuole convertire.
Risultato	Rende il valore convertito in radianti.
Validità	Rule, Task.
Note	Tutte le funzioni trigonometriche operano su argomenti espressi in radianti.

Esempi

Esempio di utilizzo:

```
sinTeta = sin(torad(30))
; dopo l'esecuzione dell'istruzione, il contenuto della variabile loca
```

© 2025 Robox SpA

Funzione toupper()

Funzione che converte un carattere in maiuscolo.

Sintassi	**i32 toupper (i32 ch)**
ch	Carattere da convertire.
Risultato	Rende il carattere convertito.
Validità	Rule, Task.
Note	

Funzione tracking()

Funzione che permette di portare la variabile controllata (tipicamente un asse) al target specificato con la velocità specificata.

La funzione, runtime, controlla che non siano superate velocità e accelerazione massime specificate per la variabile controllata.

La funzione controlla inoltre, sempre runtime, che la variabile controllata non esca fuori dai finecorsa specificati eseguendo un arresto controllato su di essi.

Per escludere il controllo dei paracarri impostarli allo stesso valore.

Sintassi	**i32 tracking (stru_track stru)**
stru	Struttura di tipo stru_track 916 per lo scambio dei dati.
Risultato	Rende:

Valore	Significato
-9	Parametro *delta* <= 0.
-6 [M_IN_ALARM]	Parametro *acc* nullo.
-5 [M_ILLEGAL_ARGS]	Parametro *maxspeed* nullo.
5 [M_FOLLOWING]	Target non raggiunto - inseguimento in corso.
6 [M_REACHED]	Target raggiunto - raggiunto uno dei due limiti di posizione.
8 [M_SYNCHRONIZED]	Target raggiunto - movimentazione in coppia.
9 [M_ACCEL_LIMITED]	Target non raggiunto - inseguimento in corso - limitazione dell'accelerazione.

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 tracking (stru_track stru)</td></tr>
<tr><td></td><td>
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>10 [M_SPEED_LIMITED]</td><td>Target non raggiunto - inseguimento in corso - limitazione della velocità.</td></tr>
<tr><td>17 [M_DECEL]</td><td>Target non raggiunto - decelerazione per raggiungimento di uno dei due limiti di posizione.</td></tr>
<tr><td>19 [M_OUT_OF_LIMITS]</td><td>Target non raggiunto - tentativo di uscire ulteriormente dai limiti di posizione.</td></tr>
</table>
</td></tr>
<tr><td>Validità</td><td>Rule.</td></tr>
<tr><td>Note</td><td>Qualora si volesse disabilitare il controllo dei limiti, imporli uguali.</td></tr>
</table>

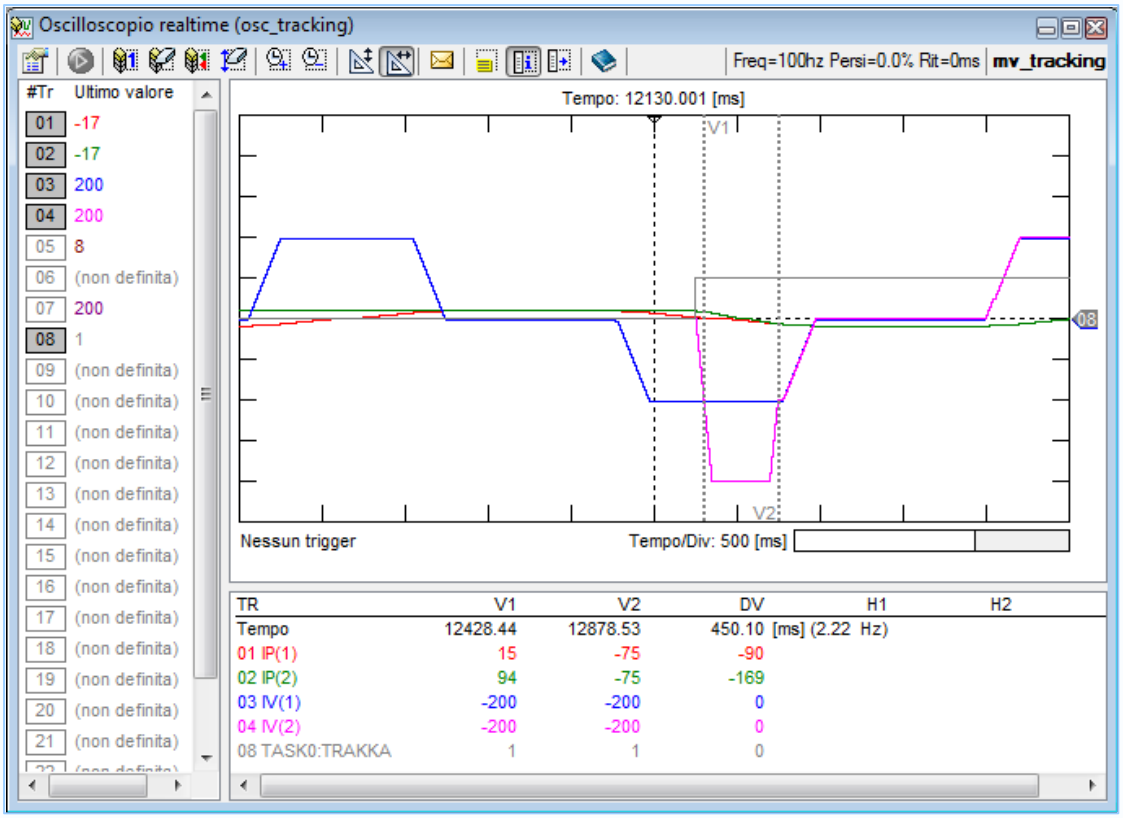
Esempi

Esempio di utilizzo:

```
RULE R_AUTO_2
axes(AX_2)
motion
    if(first_time 346())
    endif
    if(trakka)
        ; Let's track AXES 1 considering  AXES 2 limitations
        trk.delta = SI
        trk.desspeed = IV(AX_1)
        trk.despos = IP(AX_1)
        trk.actpos = IP(AX_2)
        trk.actspeed = IV(AX_2)
        trk.maxspeed = MAX_SPE(AX_2)
        trk.acc = MAX_ACC(AX_2)
        trk.posmin = MIN_STR(AX_2)
        trk.posmax = MAX_STR(AX_2)
        ; call the function
        reso = tracking 648(trk)
        ; link to the axe
        IP(AX_2) = trk.actpos
    else
        ; stop the axe
        IV(AX_2) = ramp 555(IV(AX_2), 0, MAX_ACC(AX_2))
    endif
end_motion
```

© 2025 Robox SpA

```
END_RULE
```



Funzione trunc()

Funzione che rende la parte intera del suo argomento.

Sintassi	**i32 trunc (real value)**
value	Valore o espressione da cui ricavare la parte intera.
Risultato	Rende la parte intera dell'argomento.
Validità	Rule, Task.
Note	Rende sempre un valore approssimato all'intero minore o uguale al valore entrante (es. 5.3 -> 5, 5.8 -> 5, -5.3 -> -6, ..).

Esempi

Esempio di utilizzo:

```
r(100) = trunc(650)(3.141598)     ; r(100) = 3
r(101) = trunc(3.141598 * 2 / 6.0 * sin(602)(torad(647)(90)))     ; r(101) = 1
```

© 2025 Robox SpA

Funzione udp_close()

Funzione per la chiusura di una comunicazione UDP.

Sintassi	**i32 udp_close (i32 idx)**
idx	Numero dell'handler della comunicazione UDP da chiudere.
Risultato	Rende: Valore / Significato 0 / Operazione eseguita con successo. -1 / Valore *idx* illegale. -2 / Socket già chiuso. -1000 -:- -2000 / Codice di errore della funzione di S.O. close(), con offset di 1000. Per vedere più nello specifico alcuni valori indicante errore, vedi Tabella errori resi 651.
Validità	Rule, Task.
Note	

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Tabella errore resi

Valore	Significato
-1004	[EINTR] Funzione interrotta da una chiamata di sistema.
-1009	[EBADF] Numero di file errato.

NOTA: Per la tabella completa dei codici errore vedi Codici di errore funzioni 59 tenendo conto dell'offset.

Funzione udp_open_client()

Funzione per l'apertura di una comunicazione UDP di tipo client.

Il client invia messaggi verso una stazione di destinazione specificata ed eventualmente legge la risposta proveniente dalla stessa stazione.

© 2025 Robox SpA

Sintassi	**i32 udp_open_client (u32 nPort, u32 nAddr)**
nPort	Numero della porta del server (remota) verso la quale il client invierà i suoi messaggi.
nAddr	Indirizzo IP (remoto) verso il quale il client invierà i suoi messaggi.
Risultato	Rende: Per vedere più nello specifico alcuni errori resi, vedi Tabella errori resi 652.
Validità	Rule, Task.
Note	Se il risultato è negativo non deve essere utilizzato nelle altre funzioni di gestione della comunicazione UDP.

Valore	Significato
0-:-99	Operazione eseguita con successo. Viene reso l'ID della comunicazione.
-1	*nPort* illegale.
-2	*nAddr* illegale. Deve essere diverso da 0.
-3	Tutte le entry della tabella connessioni del task sono già impegnate, impossibile stabilire una nuova connessione (max 10 connessioni).
-1000 -:- -2000	Apertura socket fallita: rende il codice fornito dalla funzione di S.O. socket(), con offset 1000.
-2000 -:- -3000	Funzione di S.O. connect() fallita: rende il codice fornito dalla connect(), con offset 2000.

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Tabella errore resi

© 2025 Robox SpA

Valore	Significato
-1013	[EACCES] Permesso per la creazione del socket del tipo e del protocollo specificato negato.
-1023	[ENFILE] La tabella di file di sistema è piena.
-1024	[EMFILE] La tabella di descrizione pre-process è piena.
-1091	[EPROTOTYPE] Il tipo di socket non è supportato dal protocollo.
-1093	[EPROTONOSUPPORT] La famiglia di protocollo non è supportata oppure il protocollo stesso non è supportato.
-1097	[EAFNOSUPPORT] Il dominio non è supportato.
-1105	[ENOBUFS] Spazio buffer insufficiente.
-2002	[ENOENT] Il socket non esiste.
-2004	[EINTR] Il tentativo di connessione è stato interrotto.
-2009	[EBADF] Id non valido.
-2013	[EACCES] Si è tentato di eseguire una connessione ad un indirizzo broadcast senza che il socket fosse stato abilitato per il broadcast.
-2014	[EFAULT] Ha un indirizzo non valido.
-2022	[EINVAL] Indirizzo socket fuori dal range ammesso.
-2040	[ELOOP] Troppi link simbolici incontrati durante la traduzione del percorso.
-2088	[ENOTSOCK] Id non valido, non si riferisce ad un socket.
-2097	[EAFNOSUPPORT] L'indirizzo non ha una famiglia valida di indirizzi corretta nel relativo campo.
-2098	[EADDRINUSE] Un altro socket è già in ascolto sulla stessa porta.
-2101	[ENETUNREACH] La rete non è raggiungibile.
-2106	[EISCONN] Il socket è già connesso.

© 2025 Robox SpA

Valore	Significato
-2110	[ETIMEDOUT] Si è avuto timeout durante il tentativo di connessione. Su richiesta di SYN (n tentativi ogni x secondi) non si è avuto risposta (server occupato).
-2111	[ECONNREFUSED] Non c'è nessuno in ascolto sull'indirizzo remoto (su richiesta di SYN il client ha ricevuto un RST inciato direttamente dalla gestione TCP del server quando qualcosa non va) oppure il server ha raggiunto il numero di connessioni massime possibili.
-2114	[EALREADY] Il socket non è bloccante e un tentativo precedente di connessione non si è ancora concluso.
-2115	[EINPROGRESS] Il socket non è bloccante e la connessione non può essere conclusa immediatamente.

NOTA: Per la tabella completa dei codici errore vedi Codici di errore funzioni 59 tenendo conto dell'offset.

Funzione udp_open_server()

Funzione per l'apertura di una comunicazione UDP di tipo server.

Il server rimane in ascolto di messaggi provenienti da diverse stazioni ed eventualmente risponde a ciascuna di esse. L'ascolto avviene sulla porta locale specificata da *nPort* e può essere limitato all'interfaccia locale specificata da *nAddr* oppure esteso a tutte le interfacce locali.

Sintassi	**i32 udp_open_server (u32 nPort [, i32 nAddr=ALL])**
nPort	Numero della porta (locale) sulla quale il server è in ascolto.
nAddr	Indirizzo IP (locale) dell'interfaccia sulla quale il server è in ascolto. (opzionale, default tutte le interfacce locali)
Risultato	Rende:

Valore	Significato
0-:-99	Operazione eseguita con successo. Viene reso l'ID della comunicazione.
-1	*nPort* illegale.
-3	Tutte le entry della tabella connessioni

© 2025 Robox SpA

Sintassi	**i32 udp_open_server (u32 nPort [, i32 nAddr=ALL])**
	<table><tr><th>Valore</th><th>Significato</th></tr><tr><td></td><td>del task sono già impegnate, impossibile stabilire una nuova connessione (max 10 connessioni).</td></tr><tr><td>-1000 -:- -2000</td><td>Apertura socket fallita: rende il codice fornito dalla funzione di S.O. socket(), con offset 1000.</td></tr><tr><td>-2000 -:- -3000</td><td>Funzione di S.O. bind() fallita: rende il codice fornito dalla bind(), con offset 2000.</td></tr></table> Per vedere più nello specifico alcuni errori resi, vedi Tabella errori resi 655.
Validità	Rule, Task.
Note	Se il risultato è negativo non deve essere utilizzato nelle altre funzioni di gestione della comunicazione UDP.

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Tabella errore resi

Valore	Significato
-1013	[EACCES] Permesso per la creazione del socket del tipo e del protocollo specificato negato.
-1023	[ENFILE] La tabella di file di sistema è piena.
-1024	[EMFILE] La tabella di descrizione pre-process è piena.
-1091	[EPROTOTYPE] Il tipo di socket non è supportato dal protocollo.
-1093	[EPROTONOSUPPORT] La famiglia di protocollo non è supportata oppure il protocollo stesso non è supportato.
-1097	[EAFNOSUPPORT] Il dominio non è supportato.

© 2025 Robox SpA

Valore	Significato
-1105	[ENOBUFS] Spazio buffer insufficiente.
-2002	[ENOENT] Il socket non esiste.
-2004	[EINTR] Il tentativo di connessione è stato interrotto.
-2009	[EBADF] Id non valido.
-2013	[EACCES] Si è tentato di eseguire una connessione ad un indirizzo broadcast senza che il socket fosse stato abilitato per il broadcast.
-2014	[EFAULT] Ha un indirizzo non valido.
-2022	[EINVAL] Indirizzo socket fuori dal range ammesso.
-2040	[ELOOP] Troppi link simbolici incontrati durante la traduzione del percorso.
-2088	[ENOTSOCK] Id non valido, non si riferisce ad un socket.
-2097	[EAFNOSUPPORT] L'indirizzo non ha una famiglia valida di indirizzi corretta nel relativo campo.
-2098	[EADDRINUSE] Un altro socket è già in ascolto sulla stessa porta.
-2101	[ENETUNREACH] La rete non è raggiungibile.
-2106	[EISCONN] Il socket è già connesso.
-2110	[ETIMEDOUT] Si è avuto timeout durante il tentativo di connessione. Su richiesta di SYN (n tentativi ogni x secondi) non si è avuto risposta (server occupato).
-2111	[ECONNREFUSED] Non c'è nessuno in ascolto sull'indirizzo remoto (su richiesta di SYN il client ha ricevuto un RST inciato direttamente dalla gestione TCP del server quando qualcosa non va) oppure il server ha raggiunto il numero di connessioni massime possibili.
-2114	[EALREADY] Il socket non è bloccante e un tentativo precedente di connessione non si è ancora concluso.
-2115	[EINPROGRESS] Il socket non è bloccante e la connessione non può essere conclusa immediatamente.

© 2025 Robox SpA

NOTA: Per la tabella completa dei codici errore vedi Codici di errore funzioni 59 tenendo conto dell'offset.

Funzione udp_recv()

Funzione per la ricezione di un messaggio UDP da un server.

La stazione (nPort e nAddr) è stata precedentemente specificata tramite una udp_open_client 651().

Sintassi	**i32 udp_recv (i32 idx, buff, i32 buffLen)**
idx	Numero dell'handler della comunicazione UDP.
buff	Buffer generico destinato a contenente il pacchetto dati ricevuto. Può essere una variabile string, struct_p 918 (struttura packed) oppure un array di struct_p. Nel caso di struct_p, la variabile deve essere dichiarata globale.
buffLen	Numero di byte massimo da ricevere.
Risultato	Rende: Valore / Significato >= 0 — Operazione eseguita con successo. Viene reso il numero di caratteri ricevuti. -1 — *idx* illegale. -2 — Socket non aperto. -3 — Socket aperto come server e non come client. -1000 -:- -2000 — Codice di errore della funzione di S.O. recv(), con offset 1000. Per vedere più nello specifico alcuni errori resi, vedi Tabella errori resi 657.
Validità	Rule, Task.
Note	La udp_recv() può essere usata solo per comunicazioni da parte del client.

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Tabella errore resi

© 2025 Robox SpA

Valore	Significato
-1004	[EINTR] La lettura è stata interrotta prima che alcun dato sia stato ricevuto.
-1009	[EBADF] Id non valido.
-1011	[EAGAIN] Ritentare. Il socket era di tipo non-blocking e la ricezione avrebbe bloccato l'esecuzione, oppure un timeout sulla ricezione è stato impostato e tale timeout è scaduto prima che alcun dato sia stato ricevuto.
-1014	[EFAULT] Il puntatore buffer punta a un'area di memoria fuori dallo spazio di processo.
-1022	[EINVAL] La lunghezza totale degli I/O è maggiore del massimo rappresentabile dal tipo restituito.
-1088	[ENOTSOCK] Id non valido, non si riferisce ad un socket.
-1105	[ENOBUFS] Il messaggio non è stato inviato perchè avrebbe superato la dimensione massima del buffer.
-1107	[ENOTCONN] La connessione non è ancora stata creata.
-1122	[EMSGSIZE] Messaggio troppo lungo.

NOTA: Per la tabella completa dei codici errore vedi Codici di errore funzioni 59 tenendo conto dell'offset.

Funzione udp_recv_from()

Funzione per ricevere un messaggio UDP con la possibilità di sapere la porta e l'indirizzo IP del mittente.

La funzione può essere usata per comunicazioni sia da parte del client sia da parte del server. In particolare, il server utilizza le informazioni di *rPort* e *rIp* restituite da questa funzione per mandare un'eventuale risposta alla stazione client tramite la udp_send_to 667 ().

Sintassi	**i32 udp_recv_from (i32 idx, buff, i32 buffLen, u32 rPort, u32 rIp)**
idx	Numero dell'handler della comunicazione UDP.
buff	Buffer generico destinato a contenente il pacchetto dati ricevuto. Può essere una variabile string, una struct_p 918 (struttura packed) oppure un array di struct_p. Nel caso di struct_p, la variabile deve essere dichiarata globale.

© 2025 Robox SpA

Sintassi	**i32 udp_recv_from (i32 idx, buff, i32 buffLen, u32 rPort, u32 rIp)**
buffLen	Numero di byte massimo da ricevere.
rPort	Numero della porta remota da cui sono stati ricevuti i dati (il suo contenuto viene scritto da questa funzione).
rIp	Indirizzo IP remoto da cui sono stati ricevuti i dati (il suo contenuto viene scritto da questa funzione, non è necessario specificarlo prima della chiamata).
Risultato	Rende: Valore / Significato >= 0 — Operazione eseguita con successo. Viene reso il numero di caratteri ricevuti. -1 — *idx* illegale. -2 — Socket non aperto. -1000 -:- -2000 — Codice di errore della funzione di S.O. recv_from(), con offset 1000. Per vedere più nello specifico alcuni errori resi, vedi Tabella errori resi 659.
Validità	Rule, Task.
Note	In testa al buffer di ricezione dati deve essere sempre riservato il posto per un u32, che contiene l'identificatore *msgId*. Questo identificatore deve essere ritrasmesso senza modifiche (tramite la funzione udp_send_to 667 ()) in testa al buffer che contiene la risposta. In assenza di questo identificatore, un client che utilizza le funzioni udp_send_notify 665() e udp_recv_notify 660() non sarà in grado di identificare la risposta come corretta.

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Tabella errore resi

Valore	Significato
-1004	[EINTR] La lettura è stata interrotta prima che alcun dato sia stato ricevuto.
-1009	[EBADF] Id non valido.

© 2025 Robox SpA

Valore	Significato
-1011	[EAGAIN] Ritentare. Il socket era di tipo non-blocking e la ricezione avrebbe bloccato l'esecuzione, oppure un timeout sulla ricezione è stato impostato e tale timeout è scaduto prima che alcun dato sia stato ricevuto.
-1014	[EFAULT] Il puntatore buffer punta a un'area di memoria fuori dallo spazio di processo.
-1022	[EINVAL] La lunghezza totale degli I/O è maggiore del massimo rappresentabile dal tipo restituito.
-1088	[ENOTSOCK] Id non valido, non si riferisce ad un socket.
-1105	[ENOBUFS] Il messaggio non è stato inviato perchè avrebbe superato la dimensione massima del buffer.
-1107	[ENOTCONN] La connessione non è ancora stata creata.
-1122	[EMSGSIZE] Messaggio troppo lungo.

NOTA: Per la tabella completa dei codici errore vedi Codici di errore funzioni 59 tenendo conto dell'offset.

Funzione udp_recv_notify()

Funzione per la ricezione di una risposta da un server ad un messaggio UDP precedentemente inviato.

Questa chiamata ha senso solo dopo aver chiamato la corrispondente funzione udp_send_notify 665(). La funzione può essere usata solo per comunicazioni da parte del client.

Sintassi	**i32 udp_recv_notify (i32 idx, buff, i32 buffLen, real tOutRx, real tOutTot)**
idx	Numero dell'handler della comunicazione UDP.
buff	Buffer generico destinato a contenente il pacchetto dati ricevuto. Può essere una variabile string, una struct_p 918 (struttura packed) oppure un array di struct_p. Nel caso di struct_p, la variabile deve essere dichiarata globale. Se struttura, il primo membro deve essere dichiarato di tipo u32 e rappresenta l'identificatore del messaggio *msgId*; il valore viene attribuito dalla funzione udp_send_notify 665(). *msgId* deve essere utilizzato lato server nel messaggio di risposta

© 2025 Robox SpA

<table>
<tr><td>Sintassi</td><td>i32 udp_recv_notify (i32 idx, buff, i32 buffLen, real tOutRx, real tOutTot)</td></tr>
<tr><td></td><td>e serve per garantire l'instradamento al client corretto. Tutti gli altri dati sono ad uso utente.
La dimensione massima di questa struttura è: MAX_UDP_DATA=508 byte.</td></tr>
<tr><td>buffLen</td><td>Numero di byte massimo da ricevere (compreso msgId).</td></tr>
<tr><td>tOutRx</td><td>Allo scadere di questo tempo di timeout viene automaticamente ritrasmesso il messaggio.</td></tr>
<tr><td>tOutTot</td><td>Allo scadere di questo tempo di timeout la funzione viene abortita.
È possibile chiamare la funzione con tOutTot=0 per provocare l'abort della funzione stessa.</td></tr>
<tr><td>Risultato</td><td>Rende:
<table>
<tr><th>Valore</th><th>Significato</th></tr>
<tr><td>> 0</td><td>Operazione eseguita con successo. Viene reso il numero di caratteri inviati.</td></tr>
<tr><td>0</td><td>In attesa di risposta.</td></tr>
<tr><td>-1</td><td>idx illegale.</td></tr>
<tr><td>-2</td><td>Socket non aperto.</td></tr>
<tr><td>-3</td><td>Socket aperto come server e non come client.</td></tr>
<tr><td>-4</td><td>buffLen maggiore del massimo consentito.</td></tr>
<tr><td>-5</td><td>Non risulta una udp_send_notify 665() in corso per l'idx specificato.</td></tr>
<tr><td>-6</td><td>tOutTot scaduto.</td></tr>
<tr><td>-1000 -:- -2000</td><td>Codice di errore della funzione di S.O. sendto(), con offset 1000 (ha reso errore la funzione sendto() di S.O. chiamata per ritentare la trasmissione dopo tmOutRx).</td></tr>
</table>
Per vedere più nello specifico alcuni errori resi, vedi Tabella errori resi 662.</td></tr>
</table>

© 2025 Robox SpA

Sintassi	**i32 udp_recv_notify (i32 idx, buff, i32 buffLen, real tOutRx, real tOutTot)**
Validità	Rule, Task.
Note	I due timeout potranno funzionare correttamente solo se la funzione viene chiamata a una frequenza sufficientemente alta rispetto ai tempi specificati.

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Tabella errore resi

Valore	Significato
-1004	[EINTR] La lettura è stata interrotta prima che alcun dato sia stato ricevuto.
-1009	[EBADF] Id non valido.
-1011	[EAGAIN] Ritentare. Il socket era di tipo non-blocking e la ricezione avrebbe bloccato l'esecuzione, oppure un timeout sulla ricezione è stato impostato e tale timeout è scaduto prima che alcun dato sia stato ricevuto.
-1014	[EFAULT] Il puntatore buffer punta a un'area di memoria fuori dallo spazio di processo.
-1022	[EINVAL] La lunghezza totale degli I/O è maggiore del massimo rappresentabile dal tipo restituito.
-1088	[ENOTSOCK] Id non valido, non si riferisce ad un socket.
-1105	[ENOBUFS] Il messaggio non è stato inviato perchè avrebbe superato la dimensione massima del buffer.
-1107	[ENOTCONN] La connessione non è ancora stata creata.
-1122	[EMSGSIZE] Messaggio troppo lungo.

NOTA: Per la tabella completa dei codici errore vedi Codici di errore funzioni 59 tenendo conto dell'offset.

Funzione udp_send()

Funzione per l'invio di un messaggio UDP ad un server.

La stazione (nPort e nAddr) è stata precedentemente specificata tramite una udp_open_client 65 ().

© 2025 Robox SpA

Sintassi	**i32 udp_send (i32 idx, buff, i32 dataLen)**
idx	Numero dell'handler della comunicazione UDP.
buff	Buffer generico contenente il pacchetto dati da trasmettere. Può essere una variabile string, una struct_p 918 (struttura packed) oppure un array di struct_p. Nel caso delle struct_p, la variabile deve essere dichiarata globale.
DataLen	Numero di byte totali da trasmettere.
Risultato	Rende: Valore / Significato >= 0 — Operazione eseguita con successo. Viene reso il numero di caratteri inviati. Il messaggio è sempre inviato per intero, quindi il numero di byte inviati è sempre uguale a *dataLen* a meno di errori. -1 — *idx* illegale. -2 — Socket non aperto. -3 — Socket aperto come server e non come client. -1000 -:- -2000 — Codice di errore della funzione di S.O. sendto(), con offset 1000. Per vedere più nello specifico alcuni errori resi, vedi Tabella errori resi 663.
Validità	Rule, Task.
Note	La udp_send() può essere usata solo per comunicazioni da parte del client.

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Tabella errore resi

Valore	Significato
-1009	[EBADF] Id non valido.

© 2025 Robox SpA

Valore	Significato
-1011	[EAGAIN o EWOULDBLOCK] Ritentare. Il socket era di tipo non-blocking e la ricezione avrebbe bloccato l'esecuzione, oppure un timeout sulla ricezione è stato impostato e tale timeout è scaduto prima che alcun dato sia stato ricevuto.
-1013	[EACCES] L'opzione SO_BROADCAST non è stata impostata sul socket e un indirizzo impostato è di tipo broadcast.
-1014	[EFAULT] Il puntatore buffer punta a un'area di memoria fuori dallo spazio di processo.
-1022	[EINVAL] La lunghezza totale degli I/O è maggiore del massimo rappresentabile dal tipo restituito.
-1032	[EPIPE] Il socket ha perso la connessione.
-1088	[ENOTSOCK] Id non valido, non si riferisce ad un socket.
-1089	[EDESTADDRREQ] In un socket non connesso l'indirizzo non è stato specificato.
-1097	[EAFNOSUPPORT] L'indirizzo specificato nella famiglia di indirizzi non può essere usato da questo socket.
-1105	[ENOBUFS] Il messaggio non è stato inviato perchè avrebbe superato la dimensione massima del buffer.
-1106	[EISCONN] Un indirizzo di destinazione è stato specificato e il socket è già connesso.
-1112	[EHOSTDOWN] La destinazione è un host sulla sottorete locale e non risponde al protocollo ARP.
-1113	[EHOSTUNREACH] Destinazione non raggiungibile.
-1122	[EMSGSIZE] Messaggio troppo lungo.

NOTA: Per la tabella completa dei codici errore vedi Codici di errore funzioni 59 tenendo conto dell'offset.

© 2025 Robox SpA

Funzione udp_send_notify()

Funzione per l'invio di un messaggio UDP con notifica a una stazione precedentemente specificata.

Il messaggio prevede obbligatoriamente la risposta, che deve essere letta tramite la funzione udp_recv_notify 660(). La funzione può essere usata solo per comunicazioni da parte del client.

Sintassi	**i32 udp_send_notify (i32 idx, buff, i32 buffLen)**
idx	Numero dell'handler della comunicazione UDP.
buff	Buffer generico contenente il pacchetto dati da trasmettere. Può essere una variabile string, una struct_p 918 (struttura packed) oppure un array di struct_p. Nel caso di struct_p, la variabile deve essere dichiarata globale. Nella struttura, il primo membro deve essere dichiarato di tipo u32 e rappresenta l'identificatore del messaggio *msgId*; il valore viene attribuito dalla funzione udp_send_notify 665(). *msgId* deve essere utilizzato lato server nel messaggio di risposta e serve per garantire l'instradamento al client corretto. Tutti gli altri dati sono ad uso utente. La dimensione massima di questa struttura è: MAX_UDP_DATA=508 byte.
buffLen	Numero di byte totali da trasmettere (compreso *msgId*).
Risultato	Rende:

Valore	Significato
>= 0	Operazione eseguita con successo. Viene reso il numero di caratteri inviati.
-1	*idx* illegale.
-2	Socket non aperto.
-3	Socket aperto come server e non come client.
-4	*buffLen* maggiore del massimo consentito.
-5	Una udp_send_notify 665() già in corso per l'*idx* specificato.
-1000 -:- -2000	Codice di errore della funzione di S.O.

© 2025 Robox SpA

Sintassi	**i32 udp_send_notify (i32 idx, buff, i32 buffLen)**
	<table><tr><th>Valore</th><th>Significato</th></tr><tr><td></td><td>sendto(), con offset 1000.</td></tr></table> Per vedere più nello specifico alcuni errori resi, vedi Tabella errori resi 666.
Validità	Rule, Task.
Note	Una sola funzione udp_send_notify() può essere attiva contemporaneamente verso ciascun server (cioè per ciascun *idx* aperto).

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Tabella errore resi

Valore	Significato
-1009	[EBADF] Id non valido.
-1011	[EAGAIN o EWOULDBLOCK] Ritentare. Il socket era di tipo non-blocking e la ricezione avrebbe bloccato l'esecuzione, oppure un timeout sulla ricezione è stato impostato e tale timeout è scaduto prima che alcun dato sia stato ricevuto.
-1013	[EACCES] L'opzione SO_BROADCAST non è stata impostata sul socket e un indirizzo impostato è di tipo broadcast.
-1014	[EFAULT] Il puntatore buffer punta a un'area di memoria fuori dallo spazio di processo.
-1022	[EINVAL] La lunghezza totale degli I/O è maggiore del massimo rappresentabile dal tipo restituito.
-1032	[EPIPE] Il socket ha perso la connessione.
-1088	[ENOTSOCK] Id non valido, non si riferisce ad un socket.
-1089	[EDESTADDRREQ] In un socket non connesso l'indirizzo non è stato specificato.
-1097	[EAFNOSUPPORT] L'indirizzo specificato nella famiglia di indirizzi non può essere usato da questo socket.
-1105	[ENOBUFS] Il messaggio non è stato inviato perchè avrebbe superato la dimensione

© 2025 Robox SpA

Valore	Significato
	massima del buffer.
-1106	[EISCONN] Un indirizzo di destinazione è stato specificato e il socket è già connesso.
-1112	[EHOSTDOWN] La destinazione è un host sulla sottorete locale e non risponde al protocollo ARP.
-1113	[EHOSTUNREACH] Destinazione non raggiungibile.
-1122	[EMSGSIZE] Messaggio troppo lungo.

NOTA: Per la tabella completa dei codici errore vedi Codici di errore funzioni 59 tenendo conto dell'offset.

Funzione udp_send_to()

Funzione per l'invio di un messaggio UDP ad una stazione (nPort e nAddr) specificata.

Sintassi	**i32 udp_send_to (i32 idx, buff, i32 dataLen, u32 rPort, u32 rIp)**
idx	Numero dell'handler della comunicazione UDP.
buff	Buffer generico contenente il pacchetto dati da trasmettere. Può essere una variabile string, una struct_p 918 (struttura packed) oppure un array di struct_p. Nel caso delle struct_p, la variabile deve essere dichiarata globale.
dataLen	Numero di byte totali da trasmettere.
rPort	Numero della porta remota a cui trasmettere i dati.
rIp	Indirizzo IP a cui trasmettere i dati.
Risultato	Rende:

Valore	Significato
>= 0	Operazione eseguita con successo. Viene reso il numero di caratteri inviati. Il messaggio viene sempre inviato per intero, quindi il numero di byte inviati è sempre uguale a *dataLen* a meno di errori.

© 2025 Robox SpA

Sintassi	**i32 udp_send_to (i32 idx, buff, i32 dataLen, u32 rPort, u32 rIp)**
	<table><tr><th>Valore</th><th>Significato</th></tr><tr><td>-1</td><td>Numero di porta non valido.</td></tr><tr><td>-2</td><td>Indirizzo non valido. Deve essere diverso da 0.</td></tr><tr><td>-3</td><td>*idx* illegale.</td></tr><tr><td>-4</td><td>Socket non aperto.</td></tr><tr><td>-1000 -:- -2000</td><td>Codice di errore della funzione di S.O. sendto(), con offset 1000.</td></tr></table> Per vedere più nello specifico alcuni errori resi, vedi Tabella errori resi 668.
Validità	Rule, Task.
Note	La funzione può essere usata per comunicazioni da parte del client e del server.

Per un esempio concreto vedi in ambiente RDE: Workspace -> Speciali -> Libreria degli esempi.

Tabella errore resi

Valore	Significato
-1009	[EBADF] Id non valido.
-1011	[EAGAIN o EWOULDBLOCK] Ritentare. Il socket era di tipo non-blocking e la ricezione avrebbe bloccato l'esecuzione, oppure un timeout sulla ricezione è stato impostato e tale timeout è scaduto prima che alcun dato sia stato ricevuto.
-1013	[EACCES] L'opzione SO_BROADCAST non è stata impostata sul socket e un indirizzo impostato è di tipo broadcast.
-1014	[EFAULT] Il puntatore buffer punta a un'area di memoria fuori dallo spazio di processo.
-1022	[EINVAL] La lunghezza totale degli I/O è maggiore del massimo rappresentabile dal tipo restituito.
-1032	[EPIPE] Il socket ha perso la connessione.

© 2025 Robox SpA

Valore	Significato
-1088	[ENOTSOCK] Id non valido, non si riferisce ad un socket.
-1089	[EDESTADDRREQ] In un socket non connesso l'indirizzo non è stato specificato.
-1097	[EAFNOSUPPORT] L'indirizzo specificato nella famiglia di indirizzi non può essere usato da questo socket.
-1105	[ENOBUFS] Il messaggio non è stato inviato perchè avrebbe superato la dimensione massima del buffer.
-1106	[EISCONN] Un indirizzo di destinazione è stato specificato e il socket è già connesso.
-1112	[EHOSTDOWN] La destinazione è un host sulla sottorete locale e non risponde al protocollo ARP.
-1113	[EHOSTUNREACH] Destinazione non raggiungibile.
-1122	[EMSGSIZE] Messaggio troppo lungo.

NOTA: Per la tabella completa dei codici errore vedi Codici di errore funzioni 59 tenendo conto dell'offset.

Funzione virt_io()

Funzione per definire una zona di input e output virtuali.

Sono entità a 16 bit non collegate direttamente a nessun dispositivo fisico ma con la possibilità di subire dei forzamenti di valore.

L'accesso avviene come per le normali inp_w e out_w. È cura dell'utilizzatore impostare lo stato reale degli input con l'apposita istruzione.

Il blocco virtuale occuperà tutti gli indirizzi compresi tra *ioBase* e (*ioBase* + maximum 390 (*nWordIn*, *nWordOut*) - 1).

Sintassi	**i32 virt_io (i32 ioBase, i32 nWordIn, i32 nWordOut)**
ioBase	Indice minore del gruppo di I/O.
nWordIn	Numero di input word gestite per questo blocco.
nWordOut	Numero di output word gestite per questo blocco.
Risultato	Rende:

© 2025 Robox SpA

Sintassi	**i32 virt_io (i32 ioBase, i32 nWordIn, i32 nWordOut)**

Valore	Significato
0	Operazione eseguita con successo.
-1	Indice *ioBase* minore di 256.
-2	Numero di word <= 0.
-3	Area I/O già in uso.
-4	Memoria piena. Impossibile creare nuovi I/O virtuali.
-5	Esauriti gestori I/O virtuali.
-10	La funzione può essere utilizzata solo da task 1-:-8.

Validità	Task.
Note	Al massimo sono definibili 4 blocchi diversi.

Esempi

Esempio di utilizzo:

```
; =======
; in task
; =======
int iobase = 2000
int n_win  = 90
int n_wout = 90
int esitoVIO
; initializzazione virtual I/Os
esitoVIO = virt_io(iobase, n_win, n_wout)
; controllo inizializzazione
if(esitoVIO < 0)
    ; initializzazione fallita
    ; ...
endif
; ==============
; in task o rule
; ==============
int desIWsts, owSts
int iw, ow
int esitoVIN, esitoVOUT
```

© 2025 Robox SpA

```
esitoVIN = set_virt_iw(2017, r(2345))
; controllo esito
if(not esitoVIN)
    ; esito fallito
    ; ...
endif
esitoVOUT = get_virt_ow(2017)
```

Funzione w0()

Questa funzione si usa per leggere o scrivere (istruzione) la prima word (a partire da quella meno significativa) dell'argomento.

Lettura:

Sintassi	**u16 w0 (i16|i32|u16|u32|real arg)**
arg	Argomento della word. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende la prima word dell'argomento.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una word non esistente ritorna un errore.

Scrittura (istruzione):

Sintassi	**w0 (i16|i32|u16|u32|real arg) = u16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Note	Questa istruzione non può essere usata per impostare il valore di un registro (es. w0(r(1))=val).

Esempi

Esempio di utilizzo:

```
I32 pluto
; se pluto vale 0x12345678
r(1) = w0(pluto) ; r(1)= 0x5678
r(1) = w1(pluto) ; r(1)= 0x1234
; definisco variabile max come integer 32 bit
I32 max
w0(max) = 0x1234
```

© 2025 Robox SpA

```
w1(max) = 0x5678
; alla fine max sarà = 0x56781234

; da RTE 33.9.0:
I32 pippo = 123
w4(pippo) = 9  ; errore!
```

Funzione w1()

Questa funzione si usa per leggere o scrivere (istruzione) la seconda word (a partire da quella meno significativa) dell'argomento.

Lettura:

Sintassi	**u16 w1 (i32|u32|real arg)**
arg	Argomento della word. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende la seconda word dell'argomento.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una word non esistente ritorna un errore.

Scrittura (istruzione):

Sintassi	**w1 (i32|u32|real arg) = u16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Note	Questa istruzione non può essere usata per impostare il valore di un registro (es. w1(r(1))=val).

Funzione w2()

Questa funzione si usa per leggere o scrivere (istruzione) la terza word (a partire da quella meno significativa) dell'argomento.

Lettura:

Sintassi	**u16 w2 (real arg)**
arg	Argomento della word. Da RTE 33.7.1 può essere un'espressione qualunque.

© 2025 Robox SpA

Sintassi	**u16 w2 (real arg)**
Risultato	Rende la terza word dell'argomento.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una word non esistente ritorna un errore.

Scrittura (istruzione):

Sintassi	**w2 (real arg) = u16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Note	Questa istruzione non può essere usata per impostare il valore di un registro (es. w2(r(1))=val).

Funzione w3()

Questa funzione si usa per leggere o scrivere (istruzione) la quarta word (a partire da quella meno significativa) dell'argomento.

Lettura:

Sintassi	**u16 w3 (real arg)**
arg	Argomento della word. Da RTE 33.7.1 può essere un'espressione qualunque.
Risultato	Rende la quarta word dell'argomento.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una word non esistente ritorna un errore.

Scrittura (istruzione):

Sintassi	**w3 (real arg) = u16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Note	Questa istruzione non può essere usata per impostare il valore di un registro (es. w3(r(1))=val).

© 2025 Robox SpA

Funzione wbe2le()

Funzione che converte big endian word (x16 o x32) in little endian word (x16).

Sintassi	**i16 wbe2le (i16|i32 data)** **u16 wbe2le (u16|u32 data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
U16 input_1
U32 input_2
U16 result
input_1 = 0x0102
input_2 = 0x01020304

result = wbe2le(input_1)
; result = 0x0201
result = wbe2le(input_2)
; result = 0x0403
```

Funzione wbit()

Funzione che permette di settare uno o più bit in un dato di tipo intero se la condizione è TRUE e di resettare uno o più bit in un dato di tipo intero se la condizione è FALSE.

Sintassi	**i32 wbit (i32 data, i32 bitMsk, bool cond)**
data	Grandezza sulla quale si desidera effettuare la scrittura.
bitMsk	Maschera dei bit da scrivere.
cond	Condizione al verificarsi della quale i bit verranno settati/resettati.
Risultato	Rende: Valore: 1 — La condizione è verificata. Valore: 0 — Altrimenti.

© 2025 Robox SpA

Sintassi	**i32 wbit (i32 data, i32 bitMsk, bool cond)**
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
sbit 674(r(1), 0x000000ff, r(100) = 34)
; mette a 1 gli 8 bit bassi quando r(100) è uguale a 34
; mette a 0 gli 8 bit bassi quando r(100) è diverso da 34
; equivale a:
if(r(100) = 34))
    r(1) = r(1) r_or 0x000000ff
else
    r(1) = r(1) r_and r_not 0x000000ff
endif
```

Funzione wle2be()

Funzione che converte little endian word (x16 o x32) in big endian word (x16).

Sintassi	**i16 wle2be (i16|i32 data)** **u16 wle2be (u16|u32 data)**
data	Data input.
Risultato	Valore di ritorno.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
U16 input_1
U32 input_2
U16 result
input_1 = 0x0201
input_2 = 0x04030201

result = wle2be 675(input_1)
; result = 0x0102
result = wle2be(input_2)
; result = 0x0102
```

© 2025 Robox SpA

Funzione write_dis()

Funzione che permette di scrivere su un display.

A seconda del controllo usato, la scrittura avviene su display differenti. Con un controllo RP-2 questa funzione scrive sul display 4x20 integrato sul controllo. Con altri modelli di controllo la scrittura avviene sull'eventuale modulo dispan collegato al controllo.

Sintassi	**i32 write_dis (string str [, i32 len=40] [, i32 offset=0])**
str	Stringa da visualizzare.
len	Numero di caratteri da visualizzare. (opzionale, default 40)
offset	Inizio del buffer. (opzionale, default 0)
Risultato	Rende: Valore: Significato <> 0: Scrittura riuscita. 0: Scrittura fallita.
Validità	Rule, Task.
Note	Le ultime due righe del display sono gestite da RTE tramite opportuni menù. Agire sulla variabile predefinita dis_display per limitare/ridurre tali gestioni.

Esempi

Esempio di utilizzo:

```
; definizioni delle variabili
STRING 929 stringaRiga1, stringaRiga2
; scriviamo sulle prime due righe del display data ed ora
stringaRiga1 = "DATA : " # 808 str_i(day, 2, 10) #"/"# str_i(month, 2, 1
stringaRiga2 = "ORA  : " # str_i(hour, 2, 10)#"/"# str_i(minute, 2, 10
; scrivo la prima stringa
write_dis 676 (stringaRiga1,20,0)
; scrivo la seconda stringa
write_dis(stringaRiga2,20,20)

; si vuole disabilitare il display da parte di RTE e pulirlo
dis_display = 0xf
; scrivo una stringa vuota di 40 caratteri
write_dis("                                        ", 40, 0)
```

© 2025 Robox SpA

Istruzioni

Istruzione __main_loop__

Questa istruzione può essere usata in alternativa alle istruzioni "loop:" o "goto 721 loop" per identificare il blocco principale di un programma R3.

Sintassi	**__main_loop__**
Validità	Task.
Note	Se presente break 694 non viene eseguito il codice fino a end_main_loop 707. Il comportamento delle istruzioni __main_loop__ ed end_main_loop sono analoghe al costrutto while 789 ed end_while 710 con condizione sempre vera.

Esempi

Esempio di utilizzo:

```
; chiamo una routine di inizializzazione
call 694 initParam
; inizio del main loop
__main_loop__
; istruzioni del main loop
  ...
; misuro la durata del main loop e memorizzo il valore nella variabile
  T1 = loop_time 389()
; eseguo una statistica sulla durata del main loop
  reso = statistics 621(T1,statT1)
; fine del main loop --> torno all'inizio
end_main_loop
```

Esempio di utilizzo:

```
; chiamo una routine di inizializzazione
call 694 initParam
; inizio del main loop
__main_loop__
; istruzioni del main loop
  ...
; se il valore della variabile "finito" è uguale a 1 esco dal loop,
; quindi le istrizioni da questo punto alla fine del loop non saranno
  if 678 (finito)
    break 694
; altre istruzioni del main loop
  ...
; fine del main loop --> torno all'inizio
end_main_loop 707
; altre istruzioni esterne al main loop
```

© 2025 Robox SpA

Istruzione _else

Questa istruzione è utilizzabile solo congiuntamente al blocco _if di cui individua l'alternativa.

Sintassi	**_else**
Validità	Rule, Task.
Note	Così come accade per l'istruzione _if, anche _else condiziona solo la linea successiva. Nel caso in cui l'utente inserisca nel ramo una seconda istruzione essa verrà sempre eseguita (anche se la condizione _else è falsa)! Questa istruzione non deve mai essere seguita da: _IF, _ELSE, IF, DO, FOR, SELECT, WHILE, __MAIN_LOOP__.

Esempio

```
; Se il controllo assi è acceso da più di 60 secondi (fronte di salita
 _if(tfb > 60)
; allora nel registro stringa sr(1) sarà memorizzata la scritta
; " sono acceso da più di un minuto "
  sr(1) = " sono acceso da più di un minuto "
; altrimenti nel registro stringa sr(1) sarà memorizzata la scritta
; " sono acceso da meno di un minuto "
_else
  sr(1) = " sono acceso da meno di un minuto "
```

Istruzione _if

Questa istruzione condiziona l'esecuzione della sola linea successiva, che viene eseguita solo se la condizione è vera.

Sintassi	**_if**
Validità	Rule, Task.
Note	Nel caso in cui l'utente inserisca nel ramo una seconda istruzione essa verrà sempre eseguita (anche se la condizione _if è falsa)! Può essere usata da sola o con _else (per gestire anche la condizione falsa). Questa istruzione non deve mai essere seguita da: _IF, _ELSE, IF, DO, FOR, SELECT, WHILE, __MAIN_LOOP__.

Esempi

Esempio di utilizzo:

© 2025 Robox SpA

```
; sul fronte di salita dell'input digitale 257
_if(rise 562(inp(257)))
; settiamo l'uscita digitale 257
  out(257) = 1
; sul fronte di salita dell'input digitale 258
_if(rise 562(inp(258)))
; resettiamo l'uscita digitale 257
  out(257) = 0
```

Esempio di utilizzo:

```
; Se il controllo assi è acceso da più di 60 secondi (fronte di salita
_if(tfb > 60)
; allora nel registro stringa sr(1) sarà memorizzata la scritta
; " sono acceso da più di un minuto "
  sr(1) = " sono acceso da più di un minuto "
; altrimenti nel registro stringa sr(1) sarà memorizzata la scritta
; " sono acceso da meno di un minuto "
_else 678
  sr(1) = " sono acceso da meno di un minuto "
```

Istruzione aux

Inizio del blocco aux all'interno di una rule. Il blocco deve terminare con l'istruzione end_aux.

Sintassi	**aux**
Validità	Rule.
Note	Nel blocco aux end_aux il programmatore dovrebbe inserire le gestioni avulse dal movimento degli assi (tale blocco può essere omesso).

Esempi

Esempio di utilizzo:

```
; inizio della Rule
rule 748 1
    ...
    ; bloccco ref (opzionale)
    ref 744
        ...
    ; fine blocco ref
    end_ref 708
    ; blocco motion (obbligatorio)
    motion 730
        ...
    ; fine blocco motion
```

© 2025 Robox SpA

```
    end_motion 707
    ; blocco aux (opzionale)
    aux
        ...
        ; attivo l'output 250 per segnalare che sono in questo blocco
        OUT(250) = 1
        ...
    ; fine blocco aux
    end_aux 705
    ...
; fine della rule
end_rule 708
```

Istruzione axes

Definizione degli assi coinvolti all'interno di una rule 748.

Sintassi	**axes (i32 nax1, i32 nax2, ..., i32 naxn)**
nax1	Numero del primo asse coinvolto.
nax2	Numero del secondo asse coinvolto.
naxn	Numero dell'ennesimo asse coinvolto.
Validità	Rule.
Note	

Esempi

Esempio di utilizzo:

```
; i primi tre assi sono coinvolti nella rule
axes(1, 2, 3)
```

Nota: l'esempio sopra è equivalente a: axes_m 680 (0x7).

Istruzione axes_m

Definizione degli assi coinvolti all'interno di una rule espressi tramite una maschera.

Sintassi	**axes_m (i32 axmask)**
axmask	maschera degli assi coinvolti.
Validità	Rule.
Note	

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
; i primi 12 assi sono coinvolti nella rule
axes_m(0xfff)
```

Nota: l'esempio sopra è equivalente a: axes 680 (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)

Istruzione b0

Questa istruzione si usa per scrivere byte 0 dell'argomento.

Sintassi	**b0 (i8|i16|i32|u8|u16|u32|real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Istruzione b1

Questa istruzione si usa per scrivere il byte 1 dell'argomento.

Sintassi	**b1 (i16|i32|u16|u32|real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Istruzione b2

Questa istruzione si usa per scrivere il byte 2 dell'argomento.

Sintassi	**b2 (i32|u32|real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**b2 (i32|u32|real arg) = u8 value**
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Istruzione b3

Questa istruzione si usa per scrivere il byte 3 dell'argomento.

Sintassi	**b3 (i32|u32|real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Istruzione b4

Questa istruzione si usa per scrivere il byte 4 dell'argomento.

Sintassi	**b4 (real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Istruzione b5

Questa istruzione si usa per scrivere il byte 5 dell'argomento.

Sintassi	**b5 (real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

© 2025 Robox SpA

Istruzione b6

Questa istruzione si usa per scrivere il byte 6 dell'argomento.

Sintassi	**b6 (real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Istruzione b7

Questa istruzione si usa per scrivere il byte 7 dell'argomento.

Sintassi	**b7 (real arg) = u8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Istruzione bcc_set_double

Questa funzione permette di scrivere un REAL nel campo "data" di una struttura BCC_MSGUSER[820] all'offset specificato.

Sintassi	**bcc_set_double (bcc_msguser str, int offset, real value)**
str	Struttura di tipo bcc_msguser[820].
offset	Offset in numero di byte a cui si vuole scrivere.
value	Il valore che si desidera inserire.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
Real valueToSend = 3.14
Int offset = 10
```

© 2025 Robox SpA

```
bcc_set_double(bcc_str, offset, valueToSend)
```

Istruzione bcc_set_float

Questa funzione permette di scrivere un FLOAT nel campo "data" di una struttura BCC_MSGUSER all'offset specificato.

Sintassi	**bcc_set_double (bcc_msguser str, int offset, float value)**
str	Struttura di tipo bcc_msguser.
offset	Offset in numero di byte a cui si vuole scrivere.
value	Il valore che si desidera inserire.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
float valueToSend = 3.14
int offset = 10

bcc_set_float(bcc_str, offset, valueToSend)
```

Istruzione bcc_set_i16

Questa funzione permette di scrivere un i16 nel campo "data" di una struttura BCC_MSGUSER all'offset specificato.

Sintassi	**bcc_set_double (bcc_msguser str, int offset, i16 value)**
str	Struttura di tipo bcc_msguser.
offset	Offset in numero di byte a cui si vuole scrivere.
value	Il valore che si desidera inserire.
Validità	Rule, Task.
Note	

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
bcc_msguser bcc_str
i16 valueToSend = 0x1234
int offset = 10

bcc_set_i16(bcc_str, offset, valueToSend)
```

Istruzione bcc_set_i32

Questa funzione permette di scrivere un i32 nel campo "data" di una struttura BCC_MSGUSER 820 all'offset specificato.

Sintassi	**bcc_set_double (bcc_msguser str, int offset, i32 value)**
str	Struttura di tipo bcc_msguser 820.
offset	Offset in numero di byte a cui si vuole scrivere.
value	Il valore che si desidera inserire.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
i32 valueToSend = 0x12345678
int offset = 10

bcc_set_i32(bcc_str, offset, valueToSend)
```

Istruzione bcc_set_i8

Questa funzione permette di scrivere un i8 nel campo "data" di una struttura BCC_MSGUSER 820 all'offset specificato.

Sintassi	**bcc_set_double (bcc_msguser str, int offset, i8 value)**
str	Struttura di tipo bcc_msguser 820.
offset	Offset in numero di byte a cui si vuole scrivere.
value	Il valore che si desidera inserire.
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**bcc_set_double (bcc_msguser str, int offset, i8 value)**
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
i8 valueToSend = 0x12
int offset = 10

bcc_set_i8(bcc_str, offset, valueToSend)
```

Istruzione bcc_set_string

Questa funzione permette di scrivere una STRING nel campo "data" di una struttura BCC_MSGUSER 820 all'offset specificato.

Sintassi	**bcc_set_double (bcc_msguser str, int offset, string value)**
str	Struttura di tipo bcc_msguser 820.
offset	Offset in numero di byte a cui si vuole scrivere.
value	Il valore che si desidera inserire.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
string valueToSend = "Hello world!"
int offset = 10

bcc_set_string(bcc_str, offset, valueToSend)
```

Istruzione bcc_set_u16

Questa funzione permette di scrivere una u16 nel campo "data" di una struttura BCC_MSGUSER 820 all'offset specificato.

Sintassi	**bcc_set_double (bcc_msguser str, int offset, u16 value)**
str	Struttura di tipo bcc_msguser 820.

© 2025 Robox SpA

Sintassi	**bcc_set_double (bcc_msguser str, int offset, u16 value)**
offset	Offset in numero di byte a cui si vuole scrivere.
value	Il valore che si desidera inserire.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
u16 valueToSend = 0x1234
int offset = 10

bcc_set_u16(bcc_str, offset, valueToSend)
```

Istruzione bcc_set_u32

Questa funzione permette di scrivere un u32 nel campo "data" di una struttura BCC_MSGUSER 820 all'offset specificato.

Sintassi	**bcc_set_double (bcc_msguser str, int offset, u32 value)**
str	Struttura di tipo bcc_msguser 820.
offset	Offset in numero di byte a cui si vuole scrivere.
value	Il valore che si desidera inserire.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
u32 valueToSend = 0x12345678
int offset = 10

bcc_set_u32(bcc_str, offset, valueToSend)
```

© 2025 Robox SpA

Istruzione bcc_set_u8

Questa funzione permette di scrivere un u8 nel campo "data" di una struttura BCC_MSGUSER[820] all'offset specificato.

Sintassi	**bcc_set_double (bcc_msguser str, int offset, u8 value)**
str	Struttura di tipo bcc_msguser[820].
offset	Offset in numero di byte a cui si vuole scrivere.
value	Il valore che si desidera inserire.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
bcc_msguser bcc_str
u8 valueToSend = 0x12
int offset = 10

bcc_set_u8(bcc_str, offset, valueToSend)
```

Istruzione bit_mask_add

Permette di sommare a una variabile di tipo INT un valore espresso in esadecimale.

Sintassi	**bit_mask_add name value**
name	Nome della variabile INT alla quale sommare il valore.
value	Valore sommato alla variabile. Deve essere scritto in esadecimale, con "0x" davanti.
Note	ATTENZIONE! questa istruzione viene gestita direttamente dall'ambiente RDE. È sconsigliato utilizzarla da task o rule.

Esempi

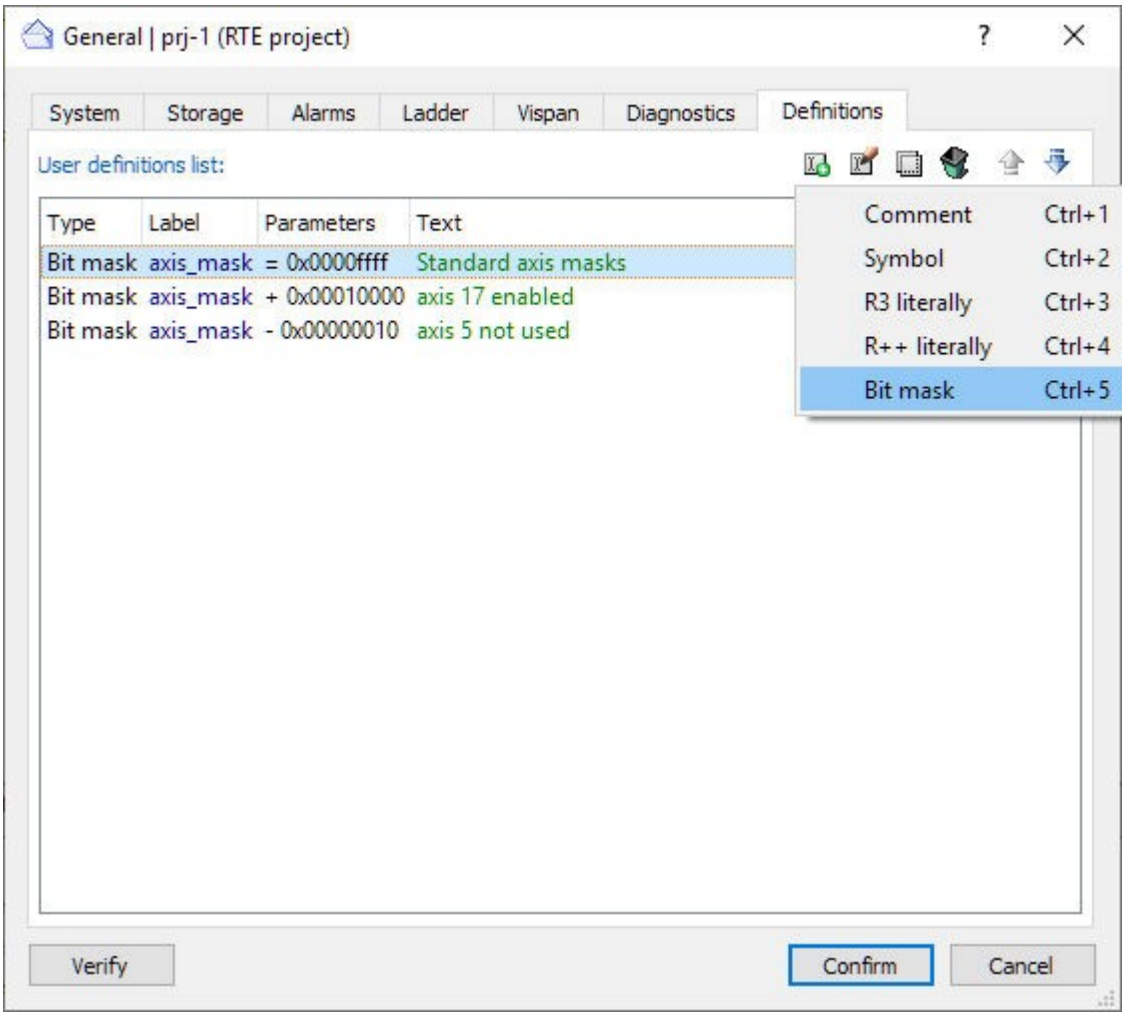
Esempio di utilizzo:

Da progetto RDE -> Generali -> Definizioni -> Maschera a bit, aggiungiamo queste tre definizioni:

- axis_mask, imposta valore 0x0000FFFF
- axis_mask, aggiungi bit 0x00010000

© 2025 Robox SpA

- axis_mask, rimuovi bit 0x00000010



Una volta generata la configurazione, nel file cfgdef.i3 avremo:

```
; Standard axis masks
bit_mask_set axis_mask 0x0000ffff
; axis 17 enabled
bit_mask_add axis_mask 0x00010000
; axis 5 not used
bit_mask_remove axis_mask 0x00000010
```

Da programma abbiamo quindi a disposizione la variabile axis_mask di tipo INT, inizializzata al valore 0x0001FFEF

```
; task:
if(axis_mask.4)
    ...
endif
```

© 2025 Robox SpA

Istruzione bit_mask_remove

Permette di sottrarre a una variabile di tipo INT un valore espresso in esadecimale.

Sintassi	**bit_mask_remove name value**
name	Nome della variabile INT alla quale sottrarre il valore.
value	Valore sottratto alla variabile. Deve essere scritto in esadecimale, con "0x" davanti.
Note	ATTENZIONE! questa istruzione viene gestita direttamente dall'ambiente RDE. È sconsigliato utilizzarla da task o rule.

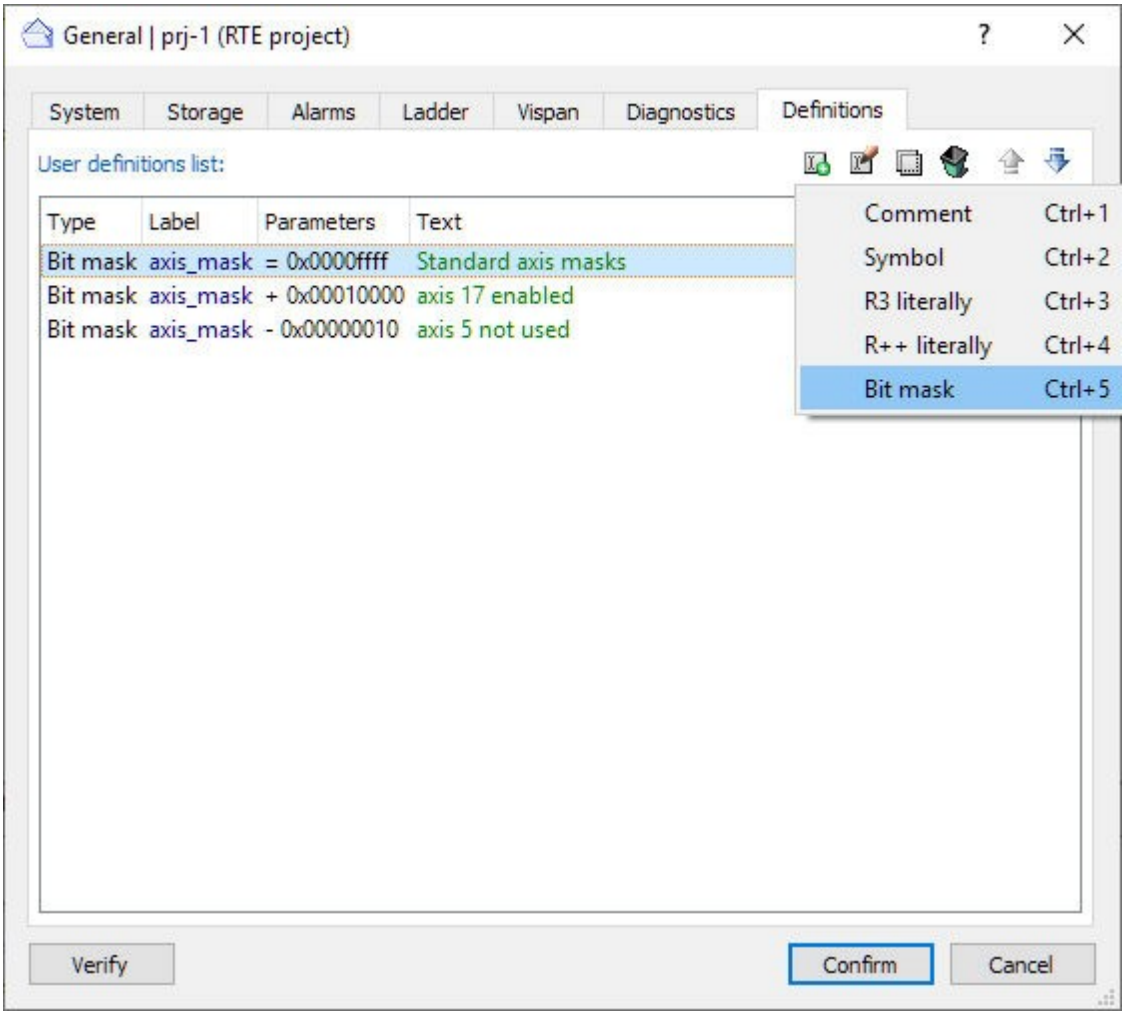
Esempi

Esempio di utilizzo:

Da progetto RDE -> Generali -> Definizioni -> Maschera a bit, aggiungiamo queste tre definizioni:

- axis_mask, imposta valore 0x0000FFFF
- axis_mask, aggiungi bit 0x00010000
- axis_mask, rimuovi bit 0x00000010

© 2025 Robox SpA



Una volta generata la configurazione, nel file cfgdef.i3 avremo:

```
; Standard axis masks
bit_mask_set axis_mask 0x0000ffff
; axis 17 enabled
bit_mask_add axis_mask 0x00010000
; axis 5 not used
bit_mask_remove axis_mask 0x00000010
```

Da programma abbiamo quindi a disposizione la variabile axis_mask di tipo INT, inizializzata al valore 0x0001FFEF

```
; task:
if (axis_mask.4)
   ....
endif
```

© 2025 Robox SpA

Istruzione bit_mask_set

Permette di inizializzare una variabile di tipo INT a un valore espresso in esadecimale.

Sintassi	**bit_mask_set name value**
name	Nome della variabile INT che verrà inizializzata.
value	Valore al quale la variabile viene inizializzata. Deve essere scritto in esadecimale, con "0x" davanti.
Note	ATTENZIONE! questa istruzione viene gestita direttamente dall'ambiente RDE. È sconsigliato utilizzarla da task o rule.

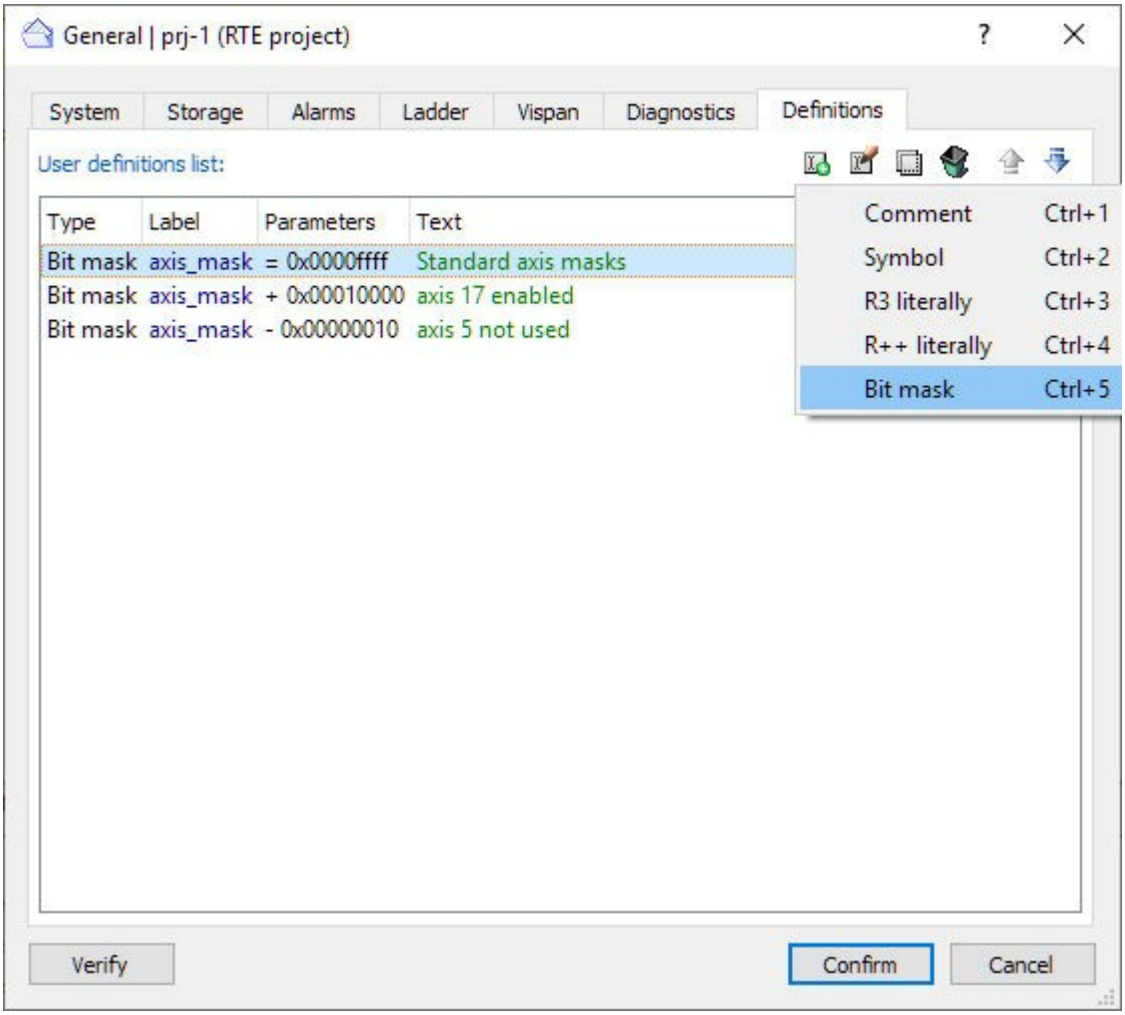
Esempi

Esempio di utilizzo:

Da progetto RDE -> Generali -> Definizioni -> Maschera a bit, aggiungiamo queste tre definizioni:

- axis_mask, imposta valore 0x0000FFFF
- axis_mask, aggiungi bit 0x00010000
- axis_mask, rimuovi bit 0x00000010

© 2025 Robox SpA



Una volta generata la configurazione, nel file cfgdef.i3 avremo:

```
; Standard axis masks
bit_mask_set axis_mask 0x0000ffff
; axis 17 enabled
bit_mask_add axis_mask 0x00010000
; axis 5 not used
bit_mask_remove axis_mask 0x00000010
```

Da programma abbiamo quindi a disposizione la variabile axis_mask di tipo INT, inizializzata al valore 0x0001FFEF

```
; task:
if (axis_mask.4)
   ....
endif
```

© 2025 Robox SpA

Istruzione break

Istruzione di uscita da un blocco di selezione.

Sintassi	**break**
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; selezione in base alla variabile pippo
select 764 (pippo)
    ; caso pippo = 1
    case 696 1
        ...
        break
    ; caso pippo = 2
    case 2
        ...
        break
    ; se nessun caso sopra si è verificato
    default 702
        ...
end select 709
```

Istruzione call

Istruzione call senza passaggio di parametri

Questa istruzione causa l'esecuzione del sottoprogramma specificato, finito il quale, l'esecuzione riprende dalla successiva istruzione.

Sintassi	**call name()**
name	É il nome della funzione 715 (function) da eseguire.
Validità	Rule, Task.
Note	Le parentesi utilizzate per la delimitazione dei parametri possono essere omesse, ma solamente nel caso la funzione non abbia parametri entranti.

Esempi

Esempio di utilizzo:

```
$task1
; chiamata della funzione
```

© 2025 Robox SpA

```
call inizializzaVariabili()
; main loop del programma
__main_loop__
    ; istruzioni del programma
    ...
end_main_loop
;*********************
; Funzione inizializza variabili
function[715] inizializzaVariabili
    pippo = 0
    pluto = 1
    ...
end_fun[706]
```

Istruzione call con passaggio di parametri

Causa l'esecuzione del sottoprogramma specificato fornendogli i parametri specificati, finito il quale, l'esecuzione riprende dalla successiva istruzione.

Sintassi	**call name (val1, val2, ... , valn)**
name	È il nome della function()[715] da eseguire.
val1	È il valore del primo parametro passato alla funzione.
val2	È il valore del secondo parametro passato alla funzione.
valn	È il valore dell'ennesimo parametro passato alla funzione.
Validità	Rule, Task.
Note	I tipi dati dei parametri passati alla function()[715] devono coincidere con quelli dichiarati in essa. Nel caso si passi un valore reale ad una funzione in cui è stato dichiarato un parametro di tipo intero, si avrà una conversione automatica di tipo, da reale a intero, senza segnalazione di errore.

Esempi

Esempio di utilizzo:

```
; chiamata di funzione senza utilizzo del valore di ritorno
call calc_diag1(12,17)
; chiamata di funzione dentro un'espressione matematica
; utilizza il valore di ritorno
rr(1) = 5.0 + calc_diag2(12,17)
...
```

© 2025 Robox SpA

```
; funzione calc_diag1
function 715 calc_diag1(REAL val1, REAL val2)
    if 723((val1<0) or (val2<0))
        rr(1) = 0
        return 746
    endif 710
    rr(1) = sqrt 620(val1*val1 + val2*val2)
end_fun 706

; funzione calc_diag2
function 715 REAL 928 calc_diag2(REAL val1, REAL val2)
    if 723((val1<0) or (val2<0))
        return 746 0.0
    endif 710
    return 746 sqrt(val1*val1 + val2*val2)
end_fun 706
```

Istruzione case

Inizio di un gruppo di istruzioni dentro al blocco di selezione select.

Se la variabile di selezione assume un valore uguale a k, permette di eseguire le istruzioni presenti nel case.

Sintassi	**case k**
k	Valore costante.
Validità	Rule, Task.
Note	Il tipo dato della costante k deve coincidere con il tipo dato della variabile confrontata.

Esempi

Esempio di utilizzo:

```
; selezione in base alla variabile pippo
select 764(pippo)
    ; caso pippo = 1
    case 1
        ...
        break 694
    ; caso pippo = 2
    case 2
        ...
        break
    ; se nessun caso sopra si è verificato
    default 702
        ...
end_select 709
```

© 2025 Robox SpA

Istruzione continue

Istruzione che forza il salto alla gestione delle reiterazioni in un ciclo for, while o do.

Sintassi	**continue**
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
for 713 (i=0, i<10 , inc(i))
    if 723 (i<5)
        ...
        ; reiteriamo il ciclo for, passando all'indice successivo
        continue
    endif 710
    ; quindi qui ci arrivo solo se i >= 5
    ...
end_for 706
```

Istruzione ct_cd

Istruzione che imposta l'input per decrementare un contatore.

Sintassi	**ct_cd (counter cnt, inCond)**
cnt	Nome del counter 921 (definito come tipo di variabile).
inCond	Ingresso di conteggio (fronte di salita). Può essere una qualsiasi condizione, anche un'espressione (es, cp(1) > 1000).
Validità	Rule, Task.
Note	I counter non sono ritentivi.

Sul fronte di salita dell'ingresso (*inCond*), viene decrementato di 1 il valore corrente (CV). L'uscita (QU) va in stato di ON quando il valore corrente (CV) diventa <= del valore di preset (PV). Il conteggio (CV) è una variabile di tipo i32 non limitata (dal massimo valore negativo passa al massimo valore positivo).

Esempi

Esempio di utilizzo:

```
counter 921 C1
```

© 2025 Robox SpA

```
ct_pv(C1, 10)        ; settiamo il preset a 10
ct_cu(C1, inp(1))    ; Up con input 1 (pulsante ADV su CPU) ...
ct_cd(C1, inp(2))    ; Down con input 2 (pulsante MODE su CPU) ...

out(257) = ct_qu(C1) ; emettiamo out 257 se valore corrente >= prese
out(258) = ct_qd(C1) ; emettiamo out 258 se valore corrente <= 0
currVal  = ct_cv(C1) ; Totalizzatore

_if(inp(3))
    ct_load(C1)      ; carica il valore corrente con quello di prese
_if(inp(4))
    ct_reset(C1)     ; azzera il valore corrente e CT_QD va in ON

; inp 1 = ADV ; inp 2 = MODE ; inp 3 = feed- ; inp 4 = feed+
```

Istruzione ct_cdr

Istruzione che imposta l'input per decrementare un contatore circolare.

Sintassi	**ct_cdr (counter cnt, inCond)**
cnt	Nome del counter (definito come tipo di variabile).
inCond	Ingresso di conteggio (fronte di salita). Può essere una qualsiasi condizione, anche un'espressione (es, cp(1) > 1000).
Validità	Rule, Task.
Note	I counter non sono ritentivi.

Sul fronte di salita dell'ingresso (*inCond*), viene decrementato di 1 il valore corrente (CV). L'uscita (QU) va in stato di ON quando il valore corrente (CV) diventa <= del valore di preset (PV). Il conteggio (CV) è una variabile di tipo i32 che arrivata al valore 0 viene reimpostata al valore di preset (PV).

Esempi

Esempio di utilizzo:

```
counter C1

ct_pv(C1, 10)        ; settiamo il preset a 10
ct_cur(C1, inp(1))    ; Up con input 1 (pulsante ADV su CPU) ...
ct_cdr(C1, inp(2))    ; Down con input 2 (pulsante MODE su CPU) ...

out(257) = ct_qu(C1) ; emettiamo out 257 se valore corrente >= prese
out(258) = ct_qd(C1) ; emettiamo out 258 se valore corrente <= 0
currVal  = ct_cv(C1) ; Totalizzatore
```

© 2025 Robox SpA

```
_if(inp(3))
    ct_load(701)(C1)       ; carica il valore corrente con quello di prese
_if(inp(4))
    ct_reset(702)(C1)      ; azzera il valore corrente e CT_QD va in ON
```

Istruzione ct_cu

Istruzione che imposta l'input per incrementare un contatore.

Sintassi	**ct_cu (counter cnt, inCond)**
cnt	Nome del counter(921) (definito come tipo di variabile).
inCond	Ingresso di conteggio (fronte di salita). Può essere una qualsiasi condizione, anche un'espressione (es, cp(1) > 1000).
Validità	Rule, Task.
Note	Sullo stesso counter può agire anche un counter ct_cd(697). I counter non sono ritentivi.

Sul fronte di salita dell'ingresso (*inCond*) viene incrementato di 1 il valore corrente (CV). L'uscita (QU) va in stato di ON quando il valore corrente (CV) diventa >= del valore di preset (PV).

Il conteggio (CV) è una variabile di tipo i32 non limitata (dal massimo valore positivo passa al massimo valore negativo).

Diagnostica

Viene generato allarme 9901 ill.Fun in caso utilizzo di parametri illegali.

Esempi

Esempio di utilizzo:

```
counter(921) C1

ct_pv(701)(C1, 10)         ; settiamo il preset a 10
ct_cu(699)(C1, inp(1))     ; Up con input 1 (pulsante ADV su CPU) ...
ct_cd(699)(C1, inp(2))     ; Down con input 2 (pulsante MODE su CPU) ...

out(257) = ct_qu(295)(C1)  ; emettiamo out 257 se valore corrente >= prese
out(258) = ct_qd(295)(C1)  ; emettiamo out 258 se valore corrente <= 0
currVal  = ct_cv(294)(C1)  ; Totalizzatore

_if(inp(3))
```

© 2025 Robox SpA

```
    ct_load(C1)       ; carica il valore corrente con quello di prese
_if(inp(4))
    ct_reset(C1)      ; azzera il valore corrente e CT_QD va in ON

; inp 1 = ADV ; inp 2 = MODE ; inp 3 = feed- ; inp 4 = feed+
```

Istruzione ct_cur

Istruzione che imposta l'input per incrementare un contatore circolare.

Sintassi	**ct_cur (counter cnt, inCond)**
cnt	Nome del counter (definito come tipo di variabile).
inCond	Ingresso di conteggio (fronte di salita). Può essere una qualsiasi condizione, anche un'espressione (es, cp(1) > 1000).
Validità	Rule, Task.
Note	Sullo stesso counter può agire anche un counter ct_cdr. I counter non sono ritentivi.

Sul fronte di salita dell'ingresso (*inCond*) viene incrementato di 1 il valore corrente (CV). L'uscita (QU) va in stato di ON quando il valore corrente (CV) diventa > del valore di preset (PV).

Il conteggio (CV) e una variabile di tipo i32 la quale una volta arrivata al valore di preset (PV) viene azzerata.

Diagnostica

Viene generato allarme 9901 ill.Fun in caso utilizzo di parametri illegali.

Esempi

Esempio di utilizzo:

```
counter C1

ct_pv(C1, 10)          ; settiamo il preset a 10
ct_cur(C1, inp(1))     ; Up con input 1 (pulsante ADV su CPU) ...
ct_cdr(C1, inp(2))     ; Down con input 2 (pulsante MODE su CPU) ...

out(257) = ct_qu(C1)   ; emettiamo out 257 se valore corrente >= prese
out(258) = ct_qd(C1)   ; emettiamo out 258 se valore corrente <= 0
currVal  = ct_cv(C1)   ; Totalizzatore

_if(inp(3))
```

© 2025 Robox SpA

```
    ct_load(C1)       ; carica il valore corrente con quello di prese
_if(inp(4))
    ct_reset(C1)      ; azzera il valore corrente e CT_QD va in ON
```

Istruzione ct_cv

Istruzione per impostare il valore corrente del counter.

NOTA: L'analoga funzione ct_cv() serve invece per leggere il valore corrente del counter.

Sintassi	**ct_cv (counter cnt, real value)**
cnt	Nome del counter (definito come tipo di variabile).
value	Valore corrente da impostare.
Validità	Rule, Task.
Note	Utile per creare counter ritentivi.

Esempi

Esempio di utilizzo:

```
ct_cv(C1, rr(1000))     ; scrivo il valore contenuto nel registro rea
currVal = ct_cv(C1)   ; scrivo il valore corrente del counter nella var
```

Istruzione ct_load

Istruzione per caricare il valore corrente (CV) del counter con quello di preset (CP).

Sintassi	**ct_load (counter cnt)**
cnt	Nome del counter (definito come tipo di variabile).
Validità	Rule, Task.
Note	

Istruzione ct_pv

Istruzione per impostare il valore di preset del counter.

NOTA: L'analoga funzione ct_pv() serve invece per leggere il valore di preset del counter.

Sintassi	**ct_pv (counter cnt, real preset)**
cnt	Nome del counter (definito come tipo di variabile).

© 2025 Robox SpA

Sintassi	**ct_pv (counter cnt, real preset)**
preset	Valore del preset.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
ct_pv(C1, rr(1000))     ; scrivo il valore contenuto nel registro rea
presetVal = ct_pv(C1)   ; scrivo il valore del preset nella variabile p
```

Istruzione ct_reset

Istruzione per resettare a 0 il contatore del counter specificato.

Sintassi	**ct_reset (counter cnt)**
cnt	Nome del counter (definito come tipo di variabile).
Validità	Rule, Task.
Note	

Istruzione default

Inizio del gruppo delle istruzioni di default dentro al blocco di selezione del costrutto select.

Sintassi	**default**
Validità	Rule, Task
Note	Se non è possibile associare la variabile di selezione a nessun "case", il gruppo di istruzioni "default", se presente, verrà eseguito.

Esempi

Esempio di utilizzo:

```
; selezione in base alla variabile pippo
select(pippo)
    ; caso pippo = 1
    case 1
        ...
        break
    ; caso pippo = 2
    case 2
```

© 2025 Robox SpA

```
        ...
        break
    ; se nessun caso sopra si è verificato
    default
        ...
end_select
```

Istruzione do

Tutte le istruzioni seguenti a questa funzione verranno ripetute fine a che la condizione end_do_while diventa falsa.

Sintassi	**do**
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; azzeriamo i primi 100 registri ritentivi reali
Int i = 0
do
    i = i+1
    nvrr(i) = 0
end_do_while(i <= 100)
```

Istruzione dw0

Questa istruzione si usa per scrivere la prima dword (a partire da quella meno significativa) dell'argomento.

Sintassi	**dw0 (i32|u32|real arg) = u32 val**
arg	Argomento della prima dword.
value	Valore da imporre in *arg*.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una dword non esistente ritorna un errore.

© 2025 Robox SpA

Istruzione dw1

Questa istruzione si usa per scrivere la seconda dword (a partire da quella meno significativa) dell'argomento.

Sintassi	**dw1 (i32|u32|real arg) = u32 val**
arg	Argomento della seconda dword.
value	Valore da imporre in *arg*.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una dword non esistente ritorna un errore.

Istruzione dwell

Sospende l'esecuzione del task per il numero specificato di secondi.

Sintassi	**dwell (real val)**
val	Numero di secondi di attesa. Se viene usato un numero negativo l'istruzione non sospenderà l'esecuzione del programma (equivalente a dwell(0)).
Validità	Task.
Note	

Esempi

Esempio di utilizzo:

```
; il task viene sospeso per 5 secondi
dwell(5)
; il task viene sospeso per mezzo secondo
dwell(0.5)
; il task viene sospeso per un numero di secondi pari al valore
; contenuto nel registro reale rr(23)
dwell(rr(23))
```

Istruzione else

Inizio di un gruppo di istruzioni di scelta alternativa nel costrutto if 723 - else 704 - endif 710.

Sintassi	**else**
Validità	Rule, Task.
Note	

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
; se il contenuto del registro reale rr(2) è uguale a zero
if 723 (rr(2) = 0)
    ; allora il contenuto del registro reale rr(1) viene moltiplicato
    rr(1) = rr(1) * 2
; altrimenti il contenuto del registro reale rr(1) viene diviso per 2
else
    rr(1) = rr(1) / 2
    ; fine del costrutto if_else_endif
endif 710
```

Istruzione end_aux

Fine del blocco aux all'interno della regola (rule).

Sintassi	**end_aux**
Validità	Rule.
Note	

Esempi

Esempio di utilizzo:

```
; inizio della Rule
rule 748 1
    ...
    ; bloccco ref (opzionale)
    ref 744
        ...
        ; fine blocco ref
    end_ref 708
    ; blocco motion (obbligatorio)
    motion 730
        ...
        ; fine blocco motion
    end_motion 707
    ; blocco aux (opzionale)
    aux 679
        ...
        ; attivo l'output 250 per segnalare che sono in questo blocco
        OUT(250) = 1
        ...
        ; fine blocco aux
    end_aux
    ...
; fine della rule
end_rule 708
```

© 2025 Robox SpA

Istruzione end_do_while

Fine del costrutto do.

Sintassi	**end_do_while**
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; azzeriamo i primi 100 registri ritentivi reali
Int i = 0
do 703
    inc(i)
    nvrr(i) = 0
end_do_while(i < 100)
```

Istruzione end_for

Fine del costrutto for.

Sintassi	**end_for**
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; costrutto for
for 713(i = 0, i < 22, i=i+2)
    pippo = pippo + i
; fine del costrutto for
end_for
```

Istruzione end_fun

Fine della funzione function().

Sintassi	**end_fun**
Validità	Rule, Task.
Note	

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

```
; *********************************
; * Funzione inizializza variabili *
; *********************************
function 715 inizializzaVariabili
    pippo = 0
    pluto = 1
    ...
end_fun
```

Istruzione end_main_loop

Questa istruzione può essere usata in alternativa alle istruzioni loop: / goto 721 loop per identificare il blocco principale di un programma R3.

Sintassi	**end_main_loop**
Validità	Task.
Note	Se presente break 694 non viene eseguito il codice fino a end_main_loop. Il comportamento delle istruzioni __main_loop__ 677 / end_main_loop sono analoghe al costrutto while 789 / end_while 710 con condizione sempre vera.

Esempi

Esempio di utilizzo:

```
; inizio del loop principale del programma (main loop)
__main_loop__ 677
   ; istruzioni del loop principale del programma (main loop)
   ...
; fine del loop principale del programma (main loop)
end_main_loop
```

Istruzione end_motion

Fine del blocco motion all'interno di una rule 748

Sintassi	**end_motion**
Validità	Rule.
Note	

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
; inizio della rule
rule 748 1
    ; definizione degli assi coinvolti
    axes 680 (1)
    ; inizio blocco motion
    motion 730
        ; istruzioni del blocco motion
        iv(1) = 0
    ; fine del blocco motion
    end_motion
; fine della rule
end_rule 708
```

Istruzione end_ref

Fine del blocco ref all'interno di una rule.

Sintassi	**end_ref**
Validità	Rule.
Note	

Esempi

Esempio di utilizzo:

```
; inizio della rule
rule 748 1
    ; inizio blocco ref
    ref 744
        ; istruzioni del blocco ref
        ...
    ; fine del blocco ref
    end_ref
    ; altri blocchi (motion, aux)
    ...
; fine della rule
end_rule 708
```

Istruzione end_rule

Fine del blocco rule all'interno del task di motion 730.

Sintassi	**end_rule**
Validità	Rule.

© 2025 Robox SpA

Sintassi	**end_rule**
Note	

Esempi

Esempio di utilizzo:

```
; inizio della rule
rule 748 1
    ; definizione degli assi coinvolti
    axes 680 (1)
    ; blocco ref (opzionale)
    ref 744
        ; istruzioni del blocco ref
        ...
    ; fine del blocco ref
    end ref 708
    ; blocco motion (obbligatorio)
    motion 730
        ; istruzioni del blocco motion
        ...
    ; fine del blocco motion
    end motion 707
    ; blocco aux (opzionale)
    aux 679
        ; istruzioni del blocco aux
        ...
    ; fine del blocco aux
    end aux 705
; fine della rule
end_rule
```

Istruzione end_select

Fine del costrutto select.

Sintassi	**end_select**
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; selezione in base alla variabile pippo
select 764(pippo)
```

© 2025 Robox SpA

```
    ; caso pippo = 1
    case 696 1
        ...
        break 694
    ; caso pippo = 2
    case 2
        ...
        break
    ; se nessun caso sopra si è verificato
    default 702
        ...
end_select
```

Istruzione end_while

Fine del costrutto while.

Sintassi	**end_while**
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; inizio del costrutto while
while 789 (inp(1))
    ; istruzioni R3 interne al costrutto while
    ...
; fine del costrutto while
end_while
```

Istruzione endif

Fine del costrutto if.

Sintassi	**endif**
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:

```
; se il contenuto del registro reale rr(2) è uguale a zero
```

© 2025 Robox SpA

```
if(rr(2) = 0)
    ; allora il contenuto del registro reale rr(1) viene moltiplicato
    rr(1) = rr(1) * 2
; altrimenti il contenuto del registro reale rr(1) viene diviso per 2
else
    rr(1) = rr(1) / 2
; fine del costrutto if_else_endif
endif
```

Istruzione excl_lam

Istruzione per includere/escludere gli assi logici del powerset.

Sintassi	**powersetname.EXCL_LAM = axMask**
powersetname	Nome del powerset.
axMask	maschera degli assi logici da includere/escludere (1 escludi, 0 includi).
Validità	Rule, Task.
Note	Se il powerset è in potenza, l'operazione è abortita e viene generato l'allarme 9904. Se uno degli assi da includere è già usato in un altro powerset, l'operazione è abortita e viene generato l'allarme 9909.

Esempi

Esempio di utilizzo:

In cfgdefs.i3:

```
...
; {7:0} Power set variables
...
power_set sys
power_set ps_qua        ; powerset definition
...
```

In task o rule:

```
INT maskAx              ; axes mask excluded/reincluded
...
ps_qua.EXCL_LAM = maskAx
```

© 2025 Robox SpA

Istruzione excl_pam

Istruzione per includere/escludere gli assi fisici del powerset.

Sintassi	**powersetname.EXCL_PAM = axMask**
powersetname	Nome del powerset.
axMask	maschera degli assi fisici da includere/escludere (1 escludi, 0 includi).
Validità	Rule, Task.
Note	Se il powerset è in potenza, l'operazione è abortita e viene generato l'allarme 9904. Se uno degli assi da reincludere è già usato in un altro powerset, l'operazione è abortita e viene generato l'allarme 9909.

Esempi

Esempio di utilizzo:

In cfgdefs.i3:

```
...
; {7:0} Power set variables
...
power_set sys
power_set ps_qua         ; powerset definition
...
```

In task o rule:

```
INT maskAx               ; axes mask excluded/reincluded
...
ps_qua.EXCL_PAM = maskAx
```

Istruzione fill_mem

Questa istruzione permette di inizializzare una variabile, generalmente di tipo indicizzato, assegnandole un valore specificato ripetuto per un determinato numero di volte.

Sintassi	**fill_mem (varDest, nEl, val)**
varDest	È la variabile che si vuole inizializzare. può essere una variabile locale oppure un registro intero, REAL o STRING.
nEl	È il numero di elementi da inizializzare, REAL o I32. Può essere anche un'espressione matematica.
val	È il valore utilizzato nella inizializzazione, REAL o I32. Può essere anche un'espressione matematica.

© 2025 Robox SpA

Sintassi	**fill_mem (varDest, nEl, val)**
Validità	Rule, Task.
Note	Nel caso di registri, l'inizializzazione viene estesa ai registri successivi a quello indicato. Nel caso delle variabili locali, che avranno tipicamente almeno un indice, occorre fare attenzione a non tentare di inizializzare un numero di elementi superiore alla dimensione dichiarata della variabile stessa. In caso contrario si avrà un opportuno messaggio di errore in fase di esecuzione del programma. Questo controllo è eseguito solamente nel caso in cui var_dest sia una variabile locale. Il numero nEl non rappresenta un numero di byte, ma un numero di elementi; ad es. nel caso di variabili reali, se si specifica nEl = 5 verranno inizializzate cinque locazioni reali consecutive della variabile specificata (in totale 40 bytes).

NOTA IMPORTANTE: nel caso di inizializzazione di registri interi o di variabili intere, val deve sempre essere uguale a zero.

Esempi

Esempio con registri interi:

```
; i registri interi da 1000 a 1099 vengono inizializzati a 0
fill_mem(r(1000), 100, 0)
```

Esempio con array locale:

```
; le prime 20 locazioni dell'array monodimensionale pluto vengono iniz
REAL 928 pluto [40]
fill_mem(pluto, 20, 12.34)
```

Esempio con stringa:

```
; I primi 40 caratteri della stringa messaggio vengono inizializzati c
STRING 929 messaggio
fill_mem(messaggio, 40, 0x20)
```

Istruzione for

Questa istruzione permette di ripetere l'esecuzione di un gruppo di istruzioni per un numero definito di volte.

Sintassi	**for (iniz, cond, update)**
iniz	Espressione di inizializzazione del ciclo, tipicamente una assegnazione di variabile (opzionale, default 0)

© 2025 Robox SpA

Sintassi	**for (iniz, cond, update)**
cond	Espressione booleana che descrive la condizione di permanenza all'interno del ciclo FOR.
update	Espressione di aggiornamento, tipicamente incremento/decremento variabile. (opzionale, default +1).
Validità	Rule, Task.
Note	Al contrario di come succede in C, in R3 non é concessa la dichiarazione di una variabile dentro gli argomenti di un ciclo for. É quindi necessario dichiarare la variabile precedentemente.

Esempi

Esempio di utilizzo:

```
Int i
...
for(i = 0, i < 22, i=i+2)
    pippo = pippo + i
end_for
```

In questo esempio la sequenza degli eventi è la seguente:

1. La variabile di conteggio i viene inizializzata a zero,
2. La condizione di permanenza in ciclo "i < 22" viene valutata e riconosciuta vera,
3. Viene eseguito il blocco di istruzioni contenuto nel ciclo FOR (in questo caso la sola istruzione "pippo = pippo + i"),
4. La variabile "i" viene incrementata di due,
5. Il ciclo riprende da capo (punto 3) e viene eseguito complessivamente 11 volte, cioè fino a quando rimane vera la condizione di permanenza in ciclo.

Nota: Nel caso in cui la condizione di permanenza in ciclo sia subito riconosciuta falsa, il blocco di istruzioni del ciclo FOR non verrà mai eseguito.

Esempio di inutilizzo dei dati facoltativi:

Utilizzando il ciclo FOR è possibile omettere sia la inizializzazione della variabile di conteggio sia l'incremento della variabile di conteggio ad esempio le seguenti istruzioni:

```
for( ,1 , )
    rr(14) = 0
    rr(15) = 0
end_for
```

Provocano la ripetizione delle istruzioni di ciclo all'infinito, la condizione di permanenza in ciclo vale 1 ed è sempre vera.

Esempio di utilizzo con l'istruzione continue:

© 2025 Robox SpA

È possibile l'utilizzo dell'istruzione continue all'interno di un ciclo FOR, la sua esecuzione causa il salto dalla linea in questione all'istruzione end_for senza eseguire altre istruzioni

```
for(i = 1, i < 7, i = i+1)
    if 723 (i = 3)
    ; salta l'azzeramento di r(3)
       continue 697
    endif 710
    r(i) = 0
end_for 706
```

Esempio di utilizzo con l'istruzione break:

È possibile l'utilizzo dell'istruzione break all'interno di un ciclo FOR, la sua esecuzione causa l'uscita immediata dal ciclo FOR senza eseguire altre istruzioni.

```
for( ,1 , )
    ; condizione di permanenza sempre vera
    if 723(ip(3) > 100.3)
        ; esci dal loop se IP(3)>100.3
        break 694
    endif 710
    ; conta quante volte viene eseguito il loop
    rr(100) = rr(100)+1
end_for 706
```

Esempio con assegnazioni dinamiche:

Il valore iniziale della variabile di controllo del ciclo FOR è uguale alla somma del contenuto del registro intero non volatile nvr(12) e del registro intero non volatile nvr(14). Il ciclo viene eseguito finché j è minore della differenza tra il contenuto del registro intero non volatile nvr(22) e del registro intero non volatile nvr(24). La variabile j viene aumentata di 1 a ogni iterazione.

```
for(j = nvr(12) + nvr(14), j < (nvr(22)-nvr(24)), inc 377(j))
    ...
end_for 706
```

Istruzione function

Istruzione che permette di dichiarare sottoprogrammi richiamabili da qualsiasi punto del programma utente.

L'utilizzo di funzioni permette di raggruppare in una unica porzione di codice quelle istruzioni che devono essere eseguite ripetutamente, allo scopo di ridurre l'occupazione di memoria e di migliorare la chiarezza del programma.

Funzione semplice

© 2025 Robox SpA

Sintassi	**function name ( )** .... istruzioni del sottoprogramma ... **end_fun**
name	È l'identificatore della funzione e segue le normali regole 790 valide per i nomi delle variabili.
Validità	Rule, Task.
Note	Usare istruzione call 694 per richiamare la funzione.

Funzione con passaggio di parametri

Sintassi	**function [type] name ([par1][, parN])** .... istruzioni del sottoprogramma ... **[return** value **]** ; (solo se valore di ritorno presente) **end_fun**
type	È l'eventuale keyword i32 oppure real che definisce il tipo del parametro da passare o il tipo del valore di ritorno. (opzionale)
name	È l'identificatore della funzione e segue le normali regole 790 valide per i nomi delle variabili.
par1	È il valore del primo parametro passato alla funzione.
parN	È l'eventuale valore dell'n-esimo parametro passato alla funzione. (opzionale)
Validità	Rule, Task.
Note	Usare istruzione call 694 per richiamare la funzione. Quando la funzione non ha alcun parametro entrante, le parentesi utilizzate per la delimitazione dei parametri possono essere omesse. Nel caso si passi un valore reale ad una funzione in cui è stato dichiarato un parametro di tipo intero, si avrà una conversione automatica di tipo, da reale a intero, senza segnalazione di errore. Da RC3E 33.01.08 è stato aggiunto il passaggio di parametri tipo array alle funzioni

© 2025 Robox SpA

Sintassi	**function [type] name ([par1][, parN])** .... istruzioni del sottoprogramma ... **[return** value **]** ; (solo se valore di ritorno presente) **end_fun**
	(array di int, di real e di strutture). Le strutture passate ad una funzione sono sempre e comunque passate come reference. Da RC3E 33.01.21 è possibile passare oggetti RPE: (tipo Axes_Group, Point_l, Point_c, Point_j, Path, Transform).

Esempi

Esempio di utilizzo:

```
;chiamata di funzione senza utilizzo del valore di ritorno
call calc_diag1(12,17)
;chiamata di funzione dentro una espressione matematica
rr(1) = 5.0 + calc_diag2(12,17)
; funzione calc_diag1
function calc_diag1 (REAL val1, REAL val2)
    if ((val1<0) or (val2<0))
        rr(1) = 0.0
    else
        rr(1) = sqrt(val1*val1 + val2*val2)
    endif
end_fun
; funzione calc_diag2
function REAL calc_diag2 (REAL val1, REAL val2)
    if ((val1<0) or (val2<0))
        return 0.0
    endif
    return sqrt(val1*val1 + val2*val2)
end_fun
```

Esempio di utilizzo:

```
STRUCT mys
    INT uno
    REAL due
END_STRUCT
mys s1
mys s2
mys s3

INT incs
INT myi1[5]
```

© 2025 Robox SpA

```
INT myi2[5]
myi1[0]= 1
myi1[1]= 2
myi1[2]= 3
myi1[3]= 4
myi1[4]= 5
; task 1
__MAIN_LOOP__
    IF (incs)
        incs = 0
        CALL f1(s1)
        CALL f2(s2)
        s3.uno = s3.uno+1
        s3.due = s3.due+1
        CALL f3(myi1, myi2)
    ENDIF
END_MAIN_LOOP
; functions
FUNCTION f1 (mys sf)
    sf.uno = sf.uno+1
    sf.due = sf.due+1
END_FUN
FUNCTION f2 (mys@ sf)
    sf.uno = sf.uno+1
    sf.due = sf.due+1
END_FUN
FUNCTION f3 (INT fi[5], INT fi2[5])
    fi2[0] = fi[0]
    fi2[1] = fi[1]
    fi2[2] = fi[2]
    fi2[3] = fi[3]
    fi2[4] = fi[4]
    fi[0] = fi[0] + 1
    fi[1] = fi[1] + 1
    fi[2] = fi[2] + 1
    fi[3] = fi[3] + 1
    fi[4] = fi[4] + 1
END_FUN
```

Esempio di utilizzo:

```
; storage
REAL localVar
struct s1
  real myVar
end_struct
s1 myStruct
; program
call incVariabile(localVar, -1)
call incVariabile(myStruct.myVar, 12.3)
; task 1
__MAIN_LOOP__
```

© 2025 Robox SpA

```
END_MAIN_LOOP
; reference
FUNCTION incVariabile(real@ varToInc , real increm)
     varToInc = varToInc + increm
END_FUN
```

Istruzione get_i64

Questa funzione legge 8 byte da un buffer, all'offset specificato, interpretandoli come una struttura stru_large_int e scrivendo il valore letto nella struttura data.

La funzione speculare è la set_i64() 770.

Sintassi	**get_i64 (buffer, i32 offset, stru_large_int data)**
buffer	può essere un array, struttura, registro
offset	offset in bytes
data	variabile di tipo stru_large_int 872 in cui salvare il valore letto
Validità	Tasks, Rules.
Note	

Esempi

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della get_i64. Per fare ciò, impostate un valore compreso tra 0 e 127 - sizeof(my_data) alla variabile offset, poi impostate un valore diverso da zero alle variabili val_high e val_low, poi verificare che la struttura my_data sia uguale a set_val.

```
u8 buffer[128]
i32 val_high
u32 val_low
stru_large_int set_val
stru_large_int my_data

; -------- loop begin --------
__MAIN_LOOP__

   ; range check
   if(offset >= 0 and (offset + sizeof(my_data)) < 128)
      set_val.high = val_high
      set_val.low  = val_low
      set_i64(buffer, offset, set_val)
      get_i64(buffer, offset, my_data)
   endif

END_MAIN_LOOP
```

© 2025 Robox SpA

Istruzione get_u64

Questa funzione legge 8 byte da un buffer, all'offset specificato, interpretandoli come una struttura stru_large_uint e scrivendo il valore letto nella struttura data.

La funzione speculare è la set_u64() 774.

Sintassi	**get_u64 (buffer, i32 offset, stru_large_uint data)**
buffer	può essere un array, struttura, registro
offset	offset in bytes
data	variabile di tipo stru_large_uint 873 in cui salvare il valore letto
Validità	Tasks, Rules.
Note	

Esempi

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della get_u64. Per fare ciò, impostate un valore compreso tra 0 e 127 - sizeof(my_data) alla variabile offset, poi impostate un valore diverso da zero alle variabili val_high e val_low, poi verificare che la struttura my_data sia uguale a set_val.

```
u8 buffer[128]
u32 val_high
u32 val_low
stru_large_uint set_val
stru_large_uint my_data

; -------- loop begin --------
__MAIN_LOOP__

   ; range check
   if(offset >= 0 and (offset + sizeof(my_data)) < 128)
      set_val.high = val_high
      set_val.low  = val_low
      set_u64(buffer, offset, set_val)
      get_u64(buffer, offset, my_data)
   endif

END_MAIN_LOOP
```

© 2025 Robox SpA

Istruzione goto

Questa istruzione trasferisce l'esecuzione del programma all'istruzione desiderata.

Sintassi	**goto string label**
label	Etichetta che individua l'istruzione del programma da cui continuare l'esecuzione.
Validità	Rule, Task.
Note	Non è possibile saltare all'esterno di una funzione o di una rule.

Le etichette possono essere lunghe 32 caratteri al massimo e il primo carattere deve essere alfabetico.

I caratteri successivi possono essere alfabetici o numerici. E' permesso inoltre l'uso del tratto basso "_".

Esempi

Esempio di utilizzo:

```
; salto alla label "PIPPO"
goto PIPPO
....
; label "PIPPO", nuovo punto di esecuzione
; notare la presenza dei due punti ":"
PIPPO:
....
```

Istruzione group

Per comprendere a fondo il significato di questa fondamentale istruzione del linguaggio di programmazione R3 bisogna aver presente quanto segue:

- Nel task sincrono (file $RULE 798) l'utente ha la possibilità di scrivere un numero molto grande di routine chiamate RULE (regole). Ogni RULE ha un compito ben preciso e il fatto di poterne scrivere tante offre la possibilità al programmatore di suddividere le varie problematiche che si troverà ad affrontare.
- Il task sincrono può eseguire contemporaneamente fino a 32 esecutori di RULE (RC).
- Per associare agli esecutori (RC) il numero della RULE il programmatore deve utilizzare l'istruzione group 721.
- Per specificare l'ordine di processazione degli esecutori (RC) il programmatore deve utilizzare l'istruzione order 735 (default in sequenza da 1 a 32).
- La variabile predefinita RC(n) individua l'n-simo dei 32 esecutori. Il suo contenuto è il numero di RULE associato all'esecutore ennesimo mediante l'istruzione group.
- La variabile predefinita OC(n) individua l'ordine di processazione dei 32 esecutori (RC).

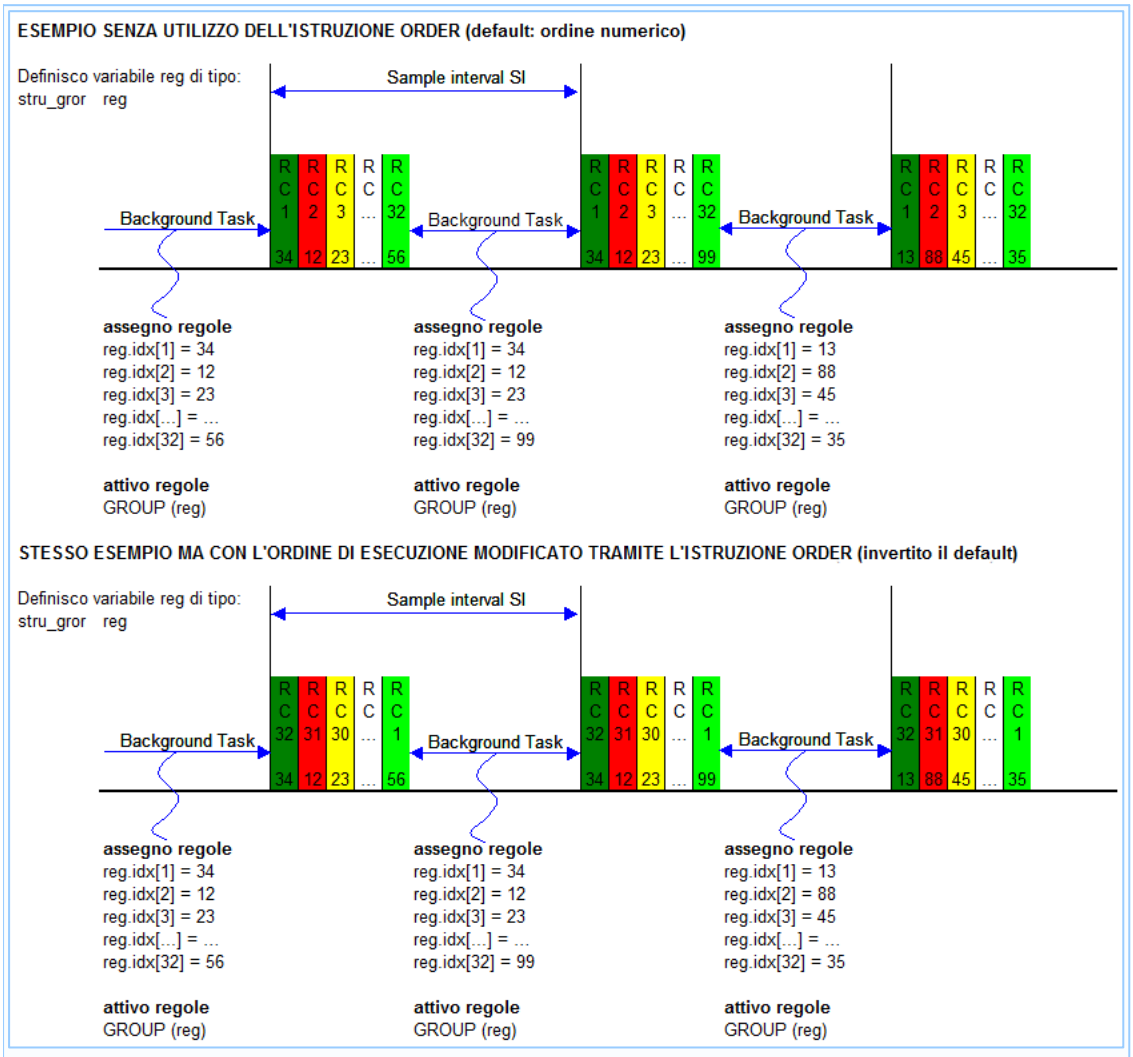
Per meglio comprendere analizziamo il momento in cui RTE si accinge ad eseguire il task sincrono (l'intervallo temporale è rappresentato dalla variabile predefinita si). RTE esegue fino a 32 esecutori di RULE (RC) i quali contengono le RULE attivate dall'istruzione group nell'ordine sequenziale specificato dalla istruzione order.

© 2025 Robox SpA

Sintassi	**group stru_gror stru**
stru	Struttura di tipo stru_gror 869 in cui verranno inserite le regole.
Validità	Rule, Task.
Note	

Esempi

Esempio di utilizzo:



Nel primo esempio con order di default avremo: OC(1) = 1 , OC(2) = 2 , ... , OC(32) = 32

Nel primo esempio con order invertito avremo: OC(1) = 32 , OC(31) = 2 , ... , OC(32) = 1

NOTA: Volendo modificare la regola di un solo esecutore, bisogna accertarsi che la struttura rimanga congrua, in quanto essa viene assunta nella sua globalità.

© 2025 Robox SpA

Esempio di utilizzo:

```
; definizione variabile assegna di tipo stru_gror
stru_gror 869 assegna
; l'esecutore di regole rc(1) esegue la rule numero 100
assegna.idx[1] = 100
; l'esecutore di regole rc(2) esegue la rule numero 200
assegna.idx[2] = 200
; l'esecutore di regole rc(3) esegue la rule numero 300
assegna.idx[3] = 300
; RTE eseguira le regole 100,200,300 presenti nel file $RULE (task sin
; Con quale ordine?
; RTE rispetterà l'ordine impostato con l'istruzione order
group 721 (assegna)
```

Istruzione if

Questa istruzione permette di far prendere decisioni logiche al controllo assi che esegue il programma R3.

Sintassi	**if (condizione)** ;blocco1 [**else** ;blocco2] **endif**
Validità	Rule, Task.
Note	

Se la "condizione" risulta vera, viene eseguito il gruppo di istruzioni definite come "blocco1".

Se la "condizione" risulta falsa, viene eseguito il gruppo di istruzioni definite come "blocco2".

Esempi

Esempi di utilizzo:

```
; se il contenuto del registro reale rr(2) è uguale a zero...
if(rr(2) = 0)
; eseguo questa istruzione...
    rr(1) = rr(1) * 2
else 704
; altrimenti eseguo quest'altra istruzione...
    rr(1) = rr(1) / 2
; fine del blocco if-else-endif
endif 710
```

Nota: Nel caso non sia necessario eseguire le istruzioni "blocco2", la keyword else 704 può essere omessa.

© 2025 Robox SpA

Esempio di applicazione:

```
; etichetta LOOP
LOOP:
; se il negato di START è VERO...
if(NOT 809(START))
; allora salto a "LOOP"
    goto 721 LOOP
endif 710
; arrivo qui solo se START è vero (vale a dire diverso da zero)
```

Esempio

```
; La nidificazione di blocchi if è ammessa:
; verifico lo stato di UP
if( UP )
; se UP è VERO controllo se PIPPO è minore o uguale a 255
    if( PIPPO <= 255 )
        PIPPO = PIPPO + 1
    endif 710
else 704
; se UP è FALSO controllo se PIPPO è maggiore o uguale a 0
    if( PIPPO >= 0 )
        PIPPO = PIPPO - 1
    endif 710
endif 710
```

Istruzione inf_report

Questa istruzione permette di scrivere una linea di informazione definita dall'utente nel report, ispezionabile tramite il comando report inviato da shell o tramite debugger da finestra di progetto.

Sintassi	**inf_report (i32 numId, string info)**
numId	Permette di settare un codice che, se in accordo con la variabile predefinita rep_mask, abilita la scrittura del messaggio nel report. Nota: i valori validi devono essere definiti nella maschera 0xFFE0. Tutti gli altri valori verranno commutati nel valore 0xFF00.
info	Linea di informazioni da inserire nel report.
Validità	Rule, Task.
Note	Evitare l'uso ricorsivo della funzione per non compromettere le tempistiche di esecuzione.

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
; memorizzo il valore di num_id nella variabile "tipo"
tipo = 0x100
; memorizzo la stringa di "info" nel registro stringa sr(1)
sr(1) = "prova di report"
...
; se memo è VERO, allora scrivo la linea di informazione nel report
if(memo)
    memo = 0
    inf_report(tipo, sr(1))
endif
```

Istruzione int_after

Istruzione che permette di mandare in esecuzione una regola dopo un prefissato tempo dalla sua chiamata. La regola deve essere scritta in un apposito file che inizia con il metacomando $RULE_TIME. Questa regola ha priorità più elevata delle rule sincrone, di conseguenza il codice in essa scritto deve essere lo stretto necessario!

Sintassi	**int_after (i32 tempo, stru_int struct)**
tempo	Tempo dopo il quale mandare in esecuzione la regola asincrona [1 - 65535 micro-secondi].
struct	Struttura di tipo stru_int che contiene le informazioni relative (opzionale, se omessa disabilita l'istruzione).
Validità	Rule, Task.
Note	L'istruzione deve essere chiamata una sola volta. Per cambiare il tempo occorre richiamare l'istruzione. Se l'istruzione viene chiamata con valore di tempo 0 o senza la struttura dati di appoggio viene disabilitata.

Diagnostica

- 2802 Interrupt rule - unknown number <STRU_INT.nRule>
- 9900 ill.Arg A:1 T:x ST:xxx Se il tempo specificato non è nel range ammesso [1 -:- 65535]
- 9900 ill.Arg A:2 T:x ST:xxx Se non è stato caricato in RAM il task delle rule a tempo ($rule_time) (Per caricare scrivere comando "load fa:mytask.cof" in autoexec.stp)
- 9901 ill.Funct. T:x ST:xxx Se l'interrupt a timer è già stato abilitato

Esempi

Esempio di utilizzo:

Si vuole emettere un output digitale per un tempo determinato

Nel file mytask.r3:

© 2025 Robox SpA

```
$ rule_time
rule 748 100
    out(257) = 0
end_rule 708
```

Nel programma principale (main program):

```
; definisco la variabile parAfter di tipo STRU_INT 870
stru_int parAfter
; esecuzione one shot
parAfter.cntr.0 = 1
; Il totalizzatore viene utilizzato come flag
parAfter.cntr.1 = 1
; regola presente nel file mytask.r3
parAfter.nrule = 100
; disabilitiamo le foto di quota degli assi
parAfter.maskPhoto = 0x0

...

LOOP:
; quando si attiva l'input inp(2)
if 723 (rise(inp(2)))
    ; attivo output digitale out(257)
    out(257) = 1
    ; abilito esecuzione rule a tempo dopo 1 ms (essa mi resetterà l'o
    int_after(1000 , parAfter)
endif 710
...
goto 721 LOOP
```

Istruzione int_inp

Istruzione che permette di mandare in esecuzione una regola sul fronte di variazione di un input digitale in grado di generare interrupt. La regola deve essere scritta in un apposito file che inizia con il metacomando $RULE_INP 798. Questa regola ha priorità più elevata delle rule 748 sincrone, di conseguenza il codice in essa scritto deve essere lo stretto necessario!

Sintassi	**int_inp (i32 numInp, stru_int struct [, i32 riseE=1] [, i32 fallE=0])**
numInp	Numero dell'input sul quale osservare la variazione
struct	Struttura di tipo stru_int 870 che contiene le informazioni relative (opzionale, se omessa disabilita la funzione)
riseE	Variazione sul fronte di salita (opzionale, default 1)
fallE	Variazione sul fronte di discesa (opzionale, default 0)

© 2025 Robox SpA

Sintassi	**int_inp (i32 numInp, stru_int struct [, i32 riseE=1] [, i32 fallE=0])**
Validità	Rule, Task.
Note	L'istruzione deve essere chiamata una sola volta. Se l'istruzione viene chiamata senza la struttura dati di appoggio viene disabilitata.

Diagnostica

- 2802 Interrupt rule - unknown number <STRU_INT.nRule>
- 9900 ill.Arg A:1 T:x ST:xxx Se l'input digitale indicato non è interruptante, o se non è programmabile il fronte richiesto
- 9900 ill.Arg A:3 T:x ST:xxx Se sia r_e che f_e sono nulli
- 9900 ill.Arg A:-1 T:x ST:xxx Se canale interno illegale
- 9900 ill.Arg A:-2 T:x ST:xxx Se canale già programmato, se il file contenente le rule specificate non è caricato in memoria
- 9900 ill.Arg A:-6 T:x ST:xxx Errore durante programmazione interrupt (es. fronte specificato non gestito)

Esempi

Esempio di utilizzo:

```
; Si vuole catturare la quota reale cp sulla variazione dell'input 257
; In tal caso le informazioni che ci da la variabile predefinita STRU_INT 870
; sono sufficienti quindi si può evitare di scrivere la regola asincro
; Nel programma principale (task1.r3):
; definisco la variabile parInp di tipo STRU_INT 870
STRU_INT 870 parInp
; esecuzione continua
parInp.cntr.0 = 0
; Il totalizzatore viene utilizzato come flag
parInp.cntr.1 = 1
; Nessuna regola asincrona (non è obbligatorio scrivere la regola)
parInp.nrule =  0
; abilitiamo foto su quota reale dei primi 4 assi
parInp.maskPhoto = 0xF
...
; inizio codice
; abilito foto su variazione input
int_inp(257 , parInp)
LOOP:
; se la foto è stata eseguita
if 723 (rise(parInp .count = 1))
; resetto il flag di foto eseguita
parInp .count = 0
; appoggiamo le foto su input su variabili di appoggio
appoQuotaAx_1 = parInp.pos [1]
```

© 2025 Robox SpA

```
appoQuotaAx_2 = parInp.pos [2]
appoQuotaAx_3 = parInp.pos [3]
appoQuotaAx_4 = parInp.pos [4]
endif 710
...
goto 721 LOOP
; Se sulla variazione si avesse voluto mandare in esecuzione una regol
; avremmo dovuto procedere come segue:
; Inizializzazione (da task1.r3) della variabile parInp.nrule al valor
parInp.nrule =  456
```

Creazione della regola asincrona (task2.r3):

```
$rule_inp
rule 748 456
    aux 748
        ; incrementiamo il registro rr(1))
        inc 377 (rr(1))
    end_aux 705
end_rule 708
```

Istruzione int_timer

Istruzione che permette di mandare in esecuzione una regola alla frequenza impostata. La regola deve essere scritta in un apposito file che inizia con il metacomando $RULE_TIME 798. Questa regola ha priorità più elevata delle rule 748 sincrone di conseguenza il codice in essa scritto deve essere lo stretto necessario!

Sintassi	**int_timer (i32 freq, stru_int struct)**
freq	Frequenza di esecuzione della regola asincrona a tempo [16 - 5000 Hz]
struct	Struttura di tipo stru_int 870 che contiene le informazioni relative (opzionale, se omessa disabilita la funzione)
Validità	Rule, Task
Note	L'istruzione deve essere chiamata una sola volta. Per cambiare la frequenza occorre richiamare l'istruzione. Se l'istruzione viene chiamata con valore di frequenza 0 o senza la struttura dati di appoggio viene disabilitata.

Diagnostica

- 2802 Interrupt rule - unknown number <STRU_INT.nRule>
- 9900 ill.Arg A:1 T:x ST:xxx Se la frequenza non è nel range ammesso [16 -:- 5000 Hz]
- 9900 ill.Arg A:2 T:x ST:xxx Se non è stato caricato in RAM il task delle rule a tempo ($rule_time)
- 9901 ill.Funct. T:x ST:xxx Se l'interrupt a timer è già stato abilitato

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

Nal file mytask.r3 :

```
$ rule_time
rule 100
      AUX
            out(257) = not out(257)
      END_AUX
end_rule
```

Nel programma principale (main program):

```
; definisco la variabile parTimer di tipo STRU_INT
STRU_INT parTimer
; esecuzione continua
parTimer.cntr.0 = 0
; Il totalizzatore è abilitato a contare
parTimer.cntr.1 = 0
; regola presente nel file mytask.r3
parTimer.nrule = 100
; disabilitiamo le foto di quota reale
parTimer.maskPhoto = 0x0
...
LOOP:
if(abilita)
    ; abilito esecuzione rule a tempo alla frequenza di 2KHz
    ; l'onda quadra avrà quindi frequenza 1KHz
    int_timer(2000 , parTimer)
else
    ; disabilito esecuzione rule a tempo alla frequenza di 2KHz
    int_timer(0)
endif
goto LOOP
```

Istruzione lit

Definisce una stringa di testo a cui si associa un nome simbolico. Tale nome può essere utilizzato ripetutamente all'interno del programma R3 e viene automaticamente riconosciuto in fase di compilazione.

Sintassi	**lit string name string str**
name	È il nome simbolico associato alla stringa di testo. Deve sottostare alle seguenti regole: • massimo 60 caratteri • il primo carattere deve essere una lettera
str	È la stringa di testo che va a sostituire il nome simbolico ogni volta che questo viene

© 2025 Robox SpA

Sintassi	**lit string name string str**
	incontrato. Le limitazioni sono le seguenti: • Lunghezza massima stringa 190 caratteri
Validità	Rule, Task.
Note	Nomi simbolici vengono riconosciuti indifferentemente se compaiono in minuscolo o in maiuscolo. Un nome viene riconosciuto solo se costituisce una parola a se stante, mentre viene ignorato se fa parte di una parola più lunga. Non è ammessa la definizione di nomi simbolici uguali ad una qualsiasi parola riservata del linguaggio R3.

Lo scopo di questo comando è quello di rendere o più compatto o più intellegibile il sorgente di un programma.

La definizione deve precedere l'uso; in caso contrario si avrà un messaggio di errore del compilatore.

Le istruzioni literally non provocano un allungamento del codice oggetto prodotto dal compilatore; possono però provocare un allungamento del tempo di compilazione.

Esempi

Esempio di file litio.i3:

```
lit iAriaCompressa  inp(257)
lit oSirena         out(257)
lit rrSpeedManual   nvrr(1)
lit rrAccAuto       nvrr(123)
```

Esempio di utilizzo nel codice:

```
; Verifico stato mandata aria compressa e attivo la sirena se manca pr
if(not iAriaCompressa)
    oSirena = true
else
    oSirena = false
endif

; definisco i valori dei registri ritentivi della macchina
rrSpeedManual = 12.5
rrAccAuto = 10000.0
```

Istruzione motion

Inizio del blocco motion all'interno di una rule.

Sintassi	**motion**
Validità	Rule.
Note	

© 2025 Robox SpA

Il campo motion - end_motion 707 contiene il codice per la generazione della traiettoria ideale.

Esempi

Esempio di rule generica:

```
; inizio della Rule
rule 748 1
    ...
    ; bloccco ref (opzionale)
    ref 744
        ...
    ; fine blocco ref
    end ref 708
    ; blocco motion (obbligatorio)
    motion
        ...
        iv(1) = 0
        ...
    ; fine blocco motion
    end motion 707
    ; blocco aux (opzionale)
    aux 679
        ...
    ; fine blocco aux
    end aux 705
    ...
    ; fine della rule
end rule 708
```

Istruzione move_byte

Questa istruzione copia il numero specificato di bytes dall'indirizzo sorgente all'indirizzo destinazione.

Sintassi	**move_byte (sorg, dest, i32 num)**
sorg	È l'indirizzo a partire dal quale inizia la zona di memoria da copiare. Può essere una variabile locale intera, reale o stringa oppure un registro intero o reale o stringa.
dest	È l'indirizzo a partire dal quale inizia la zona di memoria dove scaricare i dati. Può essere una variabile locale intera, reale o stringa oppure un registro intero o reale o stringa.
num	È il numero di byte da copiare.
Validità	Rule, Task.
Note	Il sistema non garantisce l'atomicità del buffer. Occorre prestare particolare

© 2025 Robox SpA

Sintassi	**move_byte (sorg, dest, i32 num)**
	attenzione se i dati mossi possono essere cambiati da più task a priorità diverse.

Esempi

Esempio di utilizzo:

```
; copio i registri da r(1) a r(1001) per una lunghezza di 100 byte
; siccome ogni registro occupa 4 byte, saranno copiati 25 registri
move_byte(r(1), r(1001), 100)
```

Istruzione move_mem

Questa istruzione permette di ricopiare una porzione di memoria di lunghezza specificata, a partire da una locazione sorgente, in una locazione di destinazione.

Sintassi	**move_mem (sorg, dest, num)**
sorg	È la variabile a partire da cui si intende iniziare a copiare. Non può essere una espressione matematica. Può essere un registro intero, reale o stringa, oppure una variabile locale intera, reale o stringa.
dest	È la variabile a partire dalla cui locazione viene scaricato il blocco di memoria ricopiato. Non può essere una espressione matematica. Può essere un registro intero, reale o stringa, oppure una variabile locale intera, reale o stringa. Nel caso dei registri, la copiatura viene estesa ai registri successivi a quello indicato come destinazione. Nel caso delle variabili locali, che avranno tipicamente almeno un indice, occorre fare attenzione a non tentare di trasferire un numero di elementi superiore alla dimensione dichiarata della variabile destinazione. In caso contrario si avrà in esecuzione un opportuno messaggio di errore in fase di esecuzione del programma. Questo controllo è eseguito solamente nel caso in cui dest sia una variabile locale.
num	È il numero di elementi da ricopiare. Può essere una espressione matematica. Il numero num non rappresenta un numero di bytes, ma un numero di elementi; ad es. nel caso di variabili reali, se si specifica num = 5, verranno ricopiate cinque locazioni reali consecutive della variabile specificata (in totale, 40 bytes). Il tipo di variabile sarà uguale al tipo di sorg.
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**move_mem (sorg, dest, num)**
Note	Il sistema non garantisce l'atomicità del buffer. Occorre prestare particolare attenzione se i dati mossi possono essere cambiati da più task a priorità diverse.

Nel caso delle stringhe, il carattere 0 terminatore di stringa viene ricopiato solo se è compreso nel numero di caratteri di cui si richiede la copiatura. Di conseguenza, per trasferire interamente la stringa "ciao" occorre eseguire una move_mem con num = 5.

Esempi

Esempio di utilizzo:

```
; definizione delle variabili
real 928 pluto[40]
string 929 messaggio
; i 20 registri reali dal registro rr(5) al registro rr(24)
; vengono ricopiati nell'array monodimensionale pluto
move_mem(rr(5), pluto[0], 20)
; i primi 40 caratteri del registro stringa sr(3)
; vengono ricopiati nella stringa locale messaggio
move_mem(sr(3), messaggio, 40)
```

Istruzione mt_dis

Permette di disabilitare l'esecuzione da parte di RTE del task o dei task in time-sharing specificati.

Sintassi	**mt_dis (n, n, n....)**
n	Numero del task da disabilitare
Validità	Task.
Note	Si ricorda che dopo la fase di boot rte manda in esecuzione solo il task 1. È a cura dell' applicativo utente, richiamare le rule 748 ed abilitare / disabilitare o mettere in attesa i rimanenti task

Esempi

Esempio di utilizzo:

```
; disabilita l'esecuzione da parte di RTE del task 2
mt_dis(2)
; disabilita l'esecuzione da parte di RTE di tutti i task escluso il t
mt_dis(2,3,4,5,6,7,8)
```

© 2025 Robox SpA

Istruzione mt_en

Permette di abilitare l'esecuzione da parte di RTE del task o dei task in time-sharing specificati.

Sintassi	**mt_en (n, n, n....)**
n	Numero del task da abilitare.
Validità	Task.
Note	Si ricorda che dopo la fase di boot rte manda in esecuzione solo il task 1. È a cura dell' applicativo utente, richiamare le rule 748 ed abilitare / disabilitare o mettere in attesa i rimanenti task. Deve essere presente il comando di caricamento del task da flash-card a ram (es: LOAD fa:taskname.cof)

Esempi

Esempio di utilizzo:

```
; abilita l'esecuzione del task 2
mt_en(2)
; abilita l'esecuzione di tutti gli 8 task
mt_en(2,3,4,5,6,7,8)
```

Istruzione mt_rr_flag

NOTA: Deprecata.

Questa istruzione permette di poter mandare in esecuzione un OB infinito (loop).

Impostando a 1 il BIT0 RTE si preoccuperà di far girare l'OB in partizione di tempo insieme agli altri task.

La stessa cosa poteva essere ottenuta precedentemente agendo sul sorgente dell'OB tramite l'istruzione tswait.

Sintassi	**mt_rr_flag = value**
bit0	Disable step-synchrounous task switching to be set before a long OB FUNCTION CALL or always to be restore to 0 immediatly after the OB FUNCTION CALL
Validità	Task.
Note	

© 2025 Robox SpA

Istruzione mt_wait

Questa istruzione arresta il programma (quindi il task) finché la condizione espressa non diventa vera.

Sintassi	**mt_wait (cond)**
cond	È una qualunque espressione.
Validità	Task.
Note	Di norma deve essere utilizzata nei task di cui è richiesta l'esecuzione a comando (evento). Infatti i task, stando fermi in condizioni di attesa, lasciano più tempo di esecuzione agli altri task (time-sharing). Per arrestare il task per un intervallo di tempo usare dwell 704.

Esempi

Esempi di utilizzo:

```
; Il task rimane fermo finchè non si verifica una delle due condizioni
mt_wait(inp(3) or out(17))
; Il task rimane fermo finchè il registro intero volatile r(17) non as
mt_wait(r(17) = 1000)

; Se si passa la costante 0 blocca il task per sempre
mt_wait(0)

;Se si passa una costante diversa da 0, anche decimale, la funzione no
mt_wait(0.2)
```

Istruzione order

Questa istruzione permette di definire l'ordine di processazione da parte di RTE degli esecutori di regole (rc) attivati mediante l'istruzione group.

Sintassi	**order stru_gror struct**
struct	Struttura di tipo stru_gror 869.
Validità	Rule, Task.
Note	Di default essi sono tutti abilitati e seguono l'ordine naturale. (rc(1)=primo , rc(2)=secondo ,..., rc(32)=ultimo) Nota: Per disabilitare da parte di RTE l'esecuzione degli rc, basterà assegnare il valore -1 all'esecutore.

Esempi

© 2025 Robox SpA

Esempi di utilizzo:

```
STRU_GROR ordine
STRU_GROR assegna
; stabiliamo la sequenza di esecuzione degli esecutori delle regole (r
; l'esecutore 3 (rc(3)) verrà eseguito per primo (=>RULE #30)
ordine.idx[1] = 3
; l'esecutore 2 (rc(2)) verrà eseguito per secondo (=>RULE #20)
ordine.idx[2] = 2
; l'esecutore 1 (rc(1)) verrà eseguito per terzo (=>RULE #10)
ordine.idx[3] = 1
; dal quarto in poi vengono disabilitati.
ordine.idx[4] = -1
; eseguo l'operazione che definisce l'ordine di esecuzione
order(ordine)
; stabiliamo quale regola devono eseguire gli esecutori
; l'esecutore di regole rc(1) esegue la RULE numero 10
assegna.idx[1] = 10
; l'esecutore di regole rc(2) esegue la RULE numero 20
assegna.idx[2] = 20
; l'esecutore di regole rc(3) esegue la RULE numero 30
assegna.idx[3] = 30
; eseguo l'operazione che assegna le regole agli esecutori
group(assegna)
; RTE eseguira nell'ordine le regole 30, 20, 10 presenti nel file $RUL
; Per avere una stima della durata temporale di tale esecuzione utiliz
```

```
; definizione della variabile assegnaRegole di tipo STRU_GROR
STRU_GROR assegnaRegole
; codice
; l'esecutore di regole rc(1) esegue la RULE numero 136
assegnaRegole.idx[1] = 136
; l'esecutore di regole rc(2) esegue la RULE numero 245
assegnaRegole.idx[2] = 245
; l'esecutore di regole rc(3) esegue la RULE numero 36
assegnaRegole.idx[3] = 36
; eseguo l'operazione che assegna le regole agli esecutori
group(assegnaRegole)
; RTE eseguira le regole 136, 245, 36 presenti nel file $RULE (task a
; Con quale ordine?
; RTE rispetterà l'ordine impostato con l'istruzione order
```

```
; definizione delle variabili ordine ed assegnaRegole di tipo STRU_GRO
STRU_GROR ordine
STRU_GROR assegnaRegole
; codice
; stabiliamo la sequenza di esecuzione degli esecutori delle regole (r
; l'esecutore 3 (rc(3)) verrà eseguito per primo (=>RULE #30)
ordine.idx[1] = 3
; l'esecutore 2 (rc(2)) verrà eseguito per secondo (=>RULE #20)
ordine.idx[2] = 2
```

© 2025 Robox SpA

```
; l'esecutore 1 (rc(1)) verrà eseguito per terzo (=>RULE #10)
ordine.idx[3] = 1
; gli esecutori da 4 a 32 vengono disabilitati - Il risultato è un gua
ordine.idx[4] = -1
; eseguo l'operazione che definisce l'ordine di esecuzione
order(ordine)
; l'esecutore di regole rc(1) esegue la RULE numero 136
assegnaRegole.idx[1] = 136
; l'esecutore di regole rc(2) esegue la RULE numero 245
assegnaRegole.idx[2] = 245
; l'esecutore di regole rc(3) esegue la RULE numero 36
assegnaRegole.idx[3] = 36
; eseguo l'operazione che assegna le regole agli esecutori
group(assegnaRegole)
; RTE eseguira le regole 136, 245, 36 presenti nel file $RULE (task a
; Con quale ordine?
; RTE rispetterà l'ordine impostato con l'istruzione order
```

Istruzione out_m

Questa istruzione viene utilizzata per modificare le variabili booleane siano esse logiche oppure fisiche. Essa imposta lo stato della variabile in modo permanente finché una successiva istruzione analoga non ne modifichi ulteriormente lo stato.

Sintassi	**out_m (s ch, s ch, ..., s ch)**
s	Rappresenta lo stato della variabile booleana: • - = FALSE (aperto se output fisico) • + = TRUE (chiuso se output fisico) (può essere omesso)
ch	È l'indice che identifica la variabile OUTPUT. Può essere un'espressione intera e può variare da 1 a 2304.
Validità	Rule, Task.
Note	È possibile effettuare la stessa operazione direttamente imponendo lo stato del canale fisico. es: OUT(n) = 1 o OUT(n) = 0.

In una istruzione possono essere specificati fino a 10 output (segnalazione di errore a livello di compilazione se ecceduti). Gli indici devono essere separati tra loro dal carattere "virgola".

Diagnostica

- 9900 ill. Arg A:1 T:x ST:xxx se il numero di out gestiti è maggiore di 10 (dove x = numero di task e xxx numero di step)

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

```
; l'uscita del canale 12 viene attivata
out_m(12)
; l'uscita del canale 12 viene disattivata
out_m(-12)

; l'uscita del canale 56 viene attivata
; mentre l'uscita canale 24 viene disattivata
out_m(+56 , -24)
; se eccita è attivo...
if(eccita)
; vengono attivate le uscite dei canali da 257 a 266
    out_m(257,258,259,260,261,262,263,264,265,266)
else
; altrimenti vengono disattivate le uscite dei canali da 257 a 266
    out_m(-257,-258,-259,-260,-261,-262,-263,-264,-265,-266)
endif
```

Istruzione qwbe2le

Converte big endian qword (x64) in little endian qword (x64).

Sintassi	**qwbe2le (stru_large_int|stru_large_uint stru)**
stru	Struttura di tipo stru_large_int o stru_large_uint da convertire.
Validità	Tasks, Rules.
Note	

Esempi

Esempio di utilizzo:

```
STRU_LARGE_UINT input

input.low  = 0x01020304
input.high = 0x05060708
QWBE2LE (input)
; input.low = 0x08070605, input.high = 0x04030201
```

© 2025 Robox SpA

Istruzione qwbedwx2le

Converte dword exchange big endian qword (x64) in little endian qword (x64).

Sintassi	**qwbedwx2le (stru_large_int|stru_large_uint stru)**
stru	Struttura di tipo stru_large_int[872] o stru_large_uint[873] da convertire.
Validità	Tasks, Rules.
Note	

Esempi

Esempio di utilizzo:

```
STRU_LARGE_UINT input

input.low  = 0x05060708
input.high = 0x01020304
QWBEDWX2LE (input)
; input.low = 0x08070605, input.high = 0x04030201
```

Istruzione qwle2be

Converte little endian qword (x64) in big endian qword (x64).

Sintassi	**qwle2be (stru_large_int|stru_large_uint stru)**
stru	Struttura di tipo stru_large_int[872] o stru_large_uint[873] da convertire.
Validità	Tasks, Rules.
Note	

Esempi

Esempio di utilizzo:

```
STRU_LARGE_UINT input

input.low  = 0x08070605
input.high = 0x04030201
QWLE2BE (input)
; input.low = 0x01020304, input.high = 0x05060708
```

© 2025 Robox SpA

Istruzione qwle2bedwx

Converte little endian qword (x64) in dword exchange big endian qword (x64).

Sintassi	**qwle2bedwx (stru_large_int| stru_large_uint stru)**
stru	Struttura di tipo stru_large_int[872] o stru_large_uint[873] da convertire.
Validità	Tasks, Rules.
Note	

Esempi

Esempio di utilizzo:

```
STRU_LARGE_UINT input

input.low  = 0x08070605
input.high = 0x04030201
QWLE2BEDWX (input)
; input.low = 0x05060708, input.high = 0x01020304
```

Istruzione qwle2bedwxf

Converte little endian qword (x64) in dword exchange big endian qword (x64).

Sintassi	**qwle2bedwxf (real data, stru_large_int| stru_large_uint stru)**
data	Data input
stru	Struttura di tipo stru_large_int[872] o stru_large_uint[873] da convertire.
Validità	Tasks, Rules.
Note	

Esempi

Esempio di utilizzo:

```
REAL input
STRU_LARGE_UINT result

input = 3.141592653589793116 ; Data memory: 18 2d 44 54 21 09 40 hex.
QWLE2BEDWXF(input, result)
; result.low  = 0x182d4454 -> data memory: 54 44 2d 18 hex.
; result.high = 0xfb210940 -> data memory: 40 09 21 fb hex.
```

© 2025 Robox SpA

Istruzione qwle2bef

Converte little endian real (x64) in big endian qword (x64).

Sintassi	**qwle2bef (real data, stru_large_int|stru_large_uint stru)**
data	Data input.
stru	Struttura di tipo stru_large_int[872] o stru_large_uint[873] da convertire.
Validità	Tasks, Rules.
Note	

Esempi

Esempio di utilizzo:

```
REAL input
STRU_LARGE_UINT result

input = 3.141592653589793116 ; Data memory: 18 2d 44 54 21 09 40 hex.
QWLE2BEF (input, result)
; result.low  = 0xfb210940 -> data memory: 40 09 21 fb hex.
; result.high = 0x182d4454 -> data memory: 54 44 2d 18 hex.
```

Istruzione qwle2mbe

Converte little endian qword (x64) in middle big endian qword (x64).

Sintassi	**qwle2mbe (stru_large_int|stru_large_uint stru)**
stru	Struttura di tipo stru_large_int[872] o stru_large_uint[873] da convertire.
Validità	Tasks, Rules.
Note	

Esempi

Esempio di utilizzo:

```
STRU_LARGE_UINT input

input.low  = 0x08070605
input.high = 0x04030201
QWLE2MBE (input)
; input.low = 0x03040102, input.high = 0x07080506
```

© 2025 Robox SpA

Istruzione qwle2mbedwx

Converte little endian qword (x64) in dword exchange middle big endian qword (x64).

Sintassi	**qwle2mbedwx (stru_large_int| stru_large_uint stru)**
stru	Struttura di tipo stru_large_int[872] o stru_large_uint[873] da convertire.
Validità	Tasks, Rules.
Note	

Esempi

Esempio di utilizzo:

```
STRU_LARGE_UINT input

input.low  = 0x08070605
input.high = 0x04030201
QWLE2MBEDWX (input)
; input.low = 0x07080506, input.high = 0x03040102
```

Istruzione qwle2mbedwxf

Converte little endian real (x64) in dword exchange middle big endian qword (x64).

Sintassi	**qwle2mbedwxf (real data, stru_large_int| stru_large_uint stru)**
data	Data input.
stru	Struttura di tipo stru_large_int[872] o stru_large_uint[873] da convertire.
Validità	Tasks, Rules.
Note	

Esempi

Esempio di utilizzo:

```
REAL input
STRU_LARGE_UINT result

input = 3.141592653589793116 ; Data memory: 18 2d 44 54 21 09 40 hex.
QWLE2MBEDWXF (input, result)
; result.low  = 0x4454182d -> data memory: 2d 18 54 44 hex.
; result.high = 0x0940fb21 -> data memory: 21 fb 40 09 hex.
```

© 2025 Robox SpA

Istruzione qwle2mbef

Converte little endian real (x64) in middle big endian qword (x64).

Sintassi	**qwle2mbef (real data, stru_large_int|stru_large_uint stru)**
data	Data input.
stru	Struttura di tipo stru_large_int[872] o stru_large_uint[873] da convertire.
Validità	Tasks, Rules.
Note	

Esempi

Esempio di utilizzo:

```
REAL input
STRU_LARGE_UINT result

input = 3.141592653589793116 ; Data memory: 18 2d 44 54 21 09 40 hex.
QWLE2MBEF (input, result)
; result.low  = 0x0940fb21 -> data memory: 21 fb 40 09 hex.
; result.high = 0x4454282d -> data memory: 2d 18 54 44 hex.
```

Istruzione qwmbe2le

Converte middle big endian qword (x64) in little endian qword (x64).

Sintassi	**qwmbe2le (stru_large_int|stru_large_uint stru)**
stru	Struttura di tipo stru_large_int[872] o stru_large_uint[873] da convertire.
Validità	Tasks, Rules.
Note	

Esempi

Esempio di utilizzo:

```
STRU_LARGE_UINT input

input.low  = 0x03040102
input.high = 0x07080506
QWMBE2LE (input)
; input.low = 0x08070605, input.high = 0x04030201
```

© 2025 Robox SpA

Istruzione qwmbedwx2le

Converte dword exchange middle big endian qword (x64) in little endian qword (x64).

Sintassi	**qwmbedwx2le (stru_large_int| stru_large_uint stru)**
stru	Struttura di tipo stru_large_int[872] o stru_large_uint[873] da convertire.
Validità	Tasks, Rules.
Note	

Esempi

Esempio di utilizzo:

```
STRU_LARGE_UINT input

input.low  = 0x07080506
input.high = 0x03040102
QWMBEDWX2LE (input)
; input.low = 0x08070605, input.high = 0x04030201
```

Istruzione ref

Inizio del blocco ref all'interno di una rule[748].

Sintassi	**ref**
Validità	Rule
Note	

Se presente, il campo REF deve essere dichiarato come primo campo all'interno della Rule. In caso contrario si ha segnalazione di errore in compilazione.

Nel campo REF di ciascuna RULE tipicamente è imposto, tramite la variabile sref, il valore che verrà inviato come riferimento agli azionamenti dell'asse o degli assi interessati (usare tref in caso di chiusura anello di velocità sul controllo assi).

Se nel campo REF non viene assegnato alcun valore alla sref dell'asse, il valore utilizzato come riferimento sarà quello della battuta precedente (comportamento analogo si avrà in caso di assegnazione di tref).

Il valore assegnato alle variabili sref può essere frutto di qualsiasi espressione: esso verrà comunque reso intero e limitato all'interno del campo di azione del dispositivo di emissione riferimento configurato nel file RTE.CFG (comportamento analogo si avrà in caso di assegnazione di tref).

Se l'allarme sr_ovf_m è abilitato per l'asse si avrà l'emissione di detto allarme ogni qualvolta, in esecuzione, il valore assegnato alla variabile sref è fuori dal campo di variazione sopra citato (comportamento analogo si avrà in caso di assegnazione di tref ed allarme tr_ovf_m abilitato).

Viene considerato errore, da parte del compilatore, qualsiasi assegnamento di un valore alle variabili sref o tref all'esterno del campo REF.

NOTA: Può essere omesso.

© 2025 Robox SpA

Esempi

Esempio di rule generica con blocco REF:

```
; inizio della Rule
rule 748 1
     ...
     ; bloccco ref (opzionale)
     ref
          ...
          sref(1) = ...
          ...
     ; fine blocco ref
     end ref 708
     ; blocco motion (obbligatorio)
     motion 730
          ...
     ; fine blocco motion
     end motion 707
     ; blocco aux (opzionale)
     aux 679
          ...
     ; fine blocco aux
     end aux 705
     ...
; fine della rule
end rule 708
```

Istruzione ref_prologue

Con questa istruzione è possibile specificare il nome di una funzione che verrà eseguita ad ogni battuta delle rule 748 (che girano a frequenza fissa impostata tramite l'istruzione rule_freq 756) prima dell'esecuzione dei campi ref 744.

Sintassi	**ref_prologue functionName**
functionName	È il nome della funzione da mandare in esecuzione (senza parametri e senza valore di ritorno).
Validità	Rule.
Note	L'istruzione deve essere chiamata una sola volta. Se l'istruzione viene chiamata omettendo *functionName* (= 0) viene disabilitata.

Esempi

Esempio di utilizzo:

```
function 715 rule_init 757()
```

© 2025 Robox SpA

```
ref_prologue 745(refPrologue)
    ; ...
end_fun

function refPrologue
    ; ...
end_fun
```

Istruzione restart_point

Questa istruzione imposta l'interruzione dell'esecuzione di un task in caso di richiesta modifca live.

Sintassi	**restart_point**
Validità	Task.
Note	Possono essere piú di una.

Esempi

Esempio di utilizzo:

```
; chiamo una routine di inizializzazione
if(rFirstBoot)
call 694 initParam
endif
; inizio del main loop
__main_loop__
; istruzioni del main loop
restart_point  ...
  ...
; misuro la durata del main loop e memorizzo il valore nella variabile
  T1 = loop_time 389()
; eseguo una statistica sulla durata del main loop
  reso = statistics 621 (T1,statT1)
; fine del main loop --> torno all'inizio
end_main_loop 707
```

Istruzione return

Interruzione della function invocata dall'istruzione call 694 con eventuale ritorno del valore.

Sintassi	**return val**
val	valore che viene restituito alla sede della chiamata della funzione (opzionale).
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**return val**
Note	Il tipo del valore di ritorno dipende dalla dichiarazione della funzione.

Esempi

Esempio di utilizzo:

```
; chiamata di funzione senza utilizzo del valore di ritorno
call 694 calc_diag1(12,17)
; chiamata di funzione dentro un'espressione matematica
; dentro l'espressione la chiamata della funzione equivale a una varia
rr(1) = 5.0 + calc_diag2(12,17)
; funzione calc_diag1
function 715 calc_diag1 (REAL val1, REAL val2)
    if 723 ((val1<0) or (val2<0))
        rr(1) = 0
        return
    endif 710
    rr(1) = sqrt 620(val1*val1 + val2*val2)
end fun 706
; funzione calc_diag2
function 715 REAL 928 calc_diag2 (REAL val1, REAL val2)
    if 723 ((val1<0) or (val2<0))
        return 0.0
    endif 710
    return sqrt(val1*val1 + val2*val2)
end fun 706
```

© 2025 Robox SpA

Istruzione rule

Identificatore di un blocco di istruzioni (eseguite su interrupt ($RULE[798]) a frequenza impostata tramite l'istruzione rule_freq) destinato a descrivere e determinare le regole di comportamento di uno o più assi.

Sintassi	**RULE num** **axes** n[,n,n,...,n] | **axes_m** m **ref** ; BLOCCO REF (opzionale) **end_ref** **motion** **if**(first_time()) ; solo la prima volta **endif** ; BLOCCO MOTION **end_motion** **aux** ; BLOCCO AUX (opzionale) **end_aux** **END_RULE**
num	È il numero di identificazione della rule. Deve essere una costante. All'interno del file $RULE[798] l'utente può scrivere infinite rule. Fino a 32 esecutori di rule (rc) possono essere attivi contemporaneamente nella battuta. Per decidere quali rule far eseguire nella prossima battuta, l'utente ha a disposizione l'istruzione group[721], mentre per decidere l'ordine di esecuzione nella battuta, l'utente ha a disposizione l'istruzione order[735]. order: istruzione usata solo una volta in inizializzazione. group: istruzione largamente utilizzata in quanto le regole attive devono cambiare in funzione dello stato della macchina.
axes (n[,n,n])	(alternativa ad *axes_m*) È l'identificatore usato per dichiarare quali sono gli assi manipolati dalla rule (1-:-32). *n* sono gli identificatori degli assi. Deve essere dichiarato almeno un asse.
axes_m (mask)	(alternativa ad *axes*) Permette di scrivere la maschera degli assi interessati (il valore può essere una espressione). **NOTA**: Anche questa dichiarazione è opzionale. Nessun controllo viene eseguito né in compilazione né in esecuzione.
ref	Se omesso, il compilatore provvederà a gestire l'algoritmo di chiusura dell'anello di posizione tramite

© 2025 Robox SpA

Sintassi	**RULE num** **axes** n[,n,n,...,n] | **axes_m** m **ref** ; BLOCCO REF (opzionale) **end_ref** **motion** **if**(first_time()) ; solo la prima volta **endif** ; BLOCCO MOTION **end_motion** **aux** ; BLOCCO AUX (opzionale) **end_aux** **END_RULE**
	la seguente formula predefinita "Chiusura anello di posizione proporzionale con feed forward di velocità": epos(n) = p_ip(n, ipp_idx) - cp(n) sref(n) = epos(n) * pro_gai(n) + kff(n) * iv(n) L'utente dovrà semplicemente utilizzare le variabili predefinite pro_gai e kff per tarare gli assi. Se non si desidera avere una componente di feed forward di velocità, è sufficiente impostare a 0 le variabili kff degli assi interessati. Se invece si impone un valore nullo di pro_gai si ottiene un pilotaggio dell'asse in anello aperto. Si tenga presente che con una retroazione di tipo puramente proporzionale l'errore di posizione che si ottiene è direttamente proporzionale alla velocità. La componente di feed forward viene inserita per portare a zero l'errore in velocità. L'utente dovrà settare i valori di ser_thr e ser_gai in modo opportuno per una corretta gestione dell'allarme d'inseguimento (sam). Si ricorda che nella variabile predefinita ser_cthr RTE mette a disposizione la soglia attuale (calcolata battuta per battuta) dell'allarme servo. Usare il tool oscilloscopio da RDE per impostare i valori corretti ser_thr e ser_gai.
end_ref	Fine del blocco ref.
motion	Nel campo motion delle rule sono descritte le equazioni che regolano lo stato del sistema governato dai controlli asse Robox e programmato in linguaggio R3. Le equazioni legheranno le variabili di stato del sistema, variabili che potranno essere, in generale, di tipo qualunque, ma, stante la tipologia delle unità di governo Robox, saranno fondamentalmente di tipo cinematico e cioè posizioni, velocità, accelerazioni. Ovvero, in linguaggio R3:

© 2025 Robox SpA

Sintassi

```
RULE num
    axes n[,n,n,...,n] | axes_m m
    ref
       ; BLOCCO REF (opzionale)
    end_ref
    motion
        if(first_time())
          ; solo la prima volta
        endif
        ; BLOCCO MOTION
    end_motion
    aux
       ; BLOCCO AUX (opzionale)
    end_aux
END_RULE
```

- ip (ideal position) cioè la posizione ideale
- iv (ideal velocity) cioè la velocità ideale
- ia (ideal acceleration) cioè l'accelerazione ideale

e anche:

- cp (current position) cioè la posizione reale
- cv (current velocity) cioè la velocità reale
- ca (current acceleration) cioè l'accelerazione reale

Dove le grandezze ideali sono le grandezze di riferimento, cioè il valore che, in un dato momento, si vorrebbe avessero posizione, velocità ed accelerazione. Le grandezze reali sono i valori attuali di posizione, velocità ed accelerazione come misurati attraverso gli opportuni trasduttori.

La grandezza misurata è di norma la posizione, per cui velocità ed accelerazione reali sono ricavate per derivazione numerica. La velocità sarà la variazione di posizione in un intervallo di campionamento (pari a si); analogamente l'accelerazione sarà la variazione di velocità in un intervallo di campionamento.

Si deve focalizzare che la posizione ideale (ip) rappresenta la posizione desiderata per un certo asse in un determinato momento. Non va confusa con la posizione dove si desidera che l'asse vada, obbedendo ad una certa legge di movimento (e impiegando pertanto un certo tempo). Nel campo motion di ciascuna rule è espressa la legge di movimento di uno o più assi. Una e una sola delle variabili ip, iv, ia potrà esservi rappresentata in forma esplicita (cioè comparire in una istruzione di assegnamento a sinistra del segno "=").

Non è permesso assegnare valori a più di una delle tre variabili sopra citate. RTE si fa carico di calcolare correttamente le due variabili mancanti, secondo le modalità sotto riportate.

© 2025 Robox SpA

Sintassi	**RULE num** **axes** n[,n,n,...,n] | **axes_m** m **ref** ; BLOCCO REF (opzionale) **end_ref** **motion** **if**(first_time()) ; solo la prima volta **endif** ; BLOCCO MOTION **end_motion** **aux** ; BLOCCO AUX (opzionale) **end_aux** **END_RULE**
	Se due o tre variabili, riferite allo stesso asse, vengono assegnate nel campo motion, RTE genera l'allarme 2800 durante l'esecuzione. Se nessuna delle tre variabili è assegnata in forma esplicita, non si ha segnalazione di errore da parte del compilatore, ma in esecuzione vengono a mancare alcuni controlli importanti (allarme servo, allarme crash, allarme speed etc.); viene inoltre a mancare il calcolo automatico delle rimanenti variabili legate agli assi. Leggi relative a più assi sono ammesse nella RULE. È sconsigliabile utilizzare la stessa RULE per descrivere il movimento di assi fra loro indipendenti. In particolare:
	• SE È ASSEGNATO IP IL COMPORTAMENTO SARÀ IL SEGUENTE: Se la posizione impostata è maggiore/minore dei finecorsa software parametrizzati nelle variabili predefinite max_str(n) e min_str(n) e la variabile predefinita cam è abilitata allora: * Se la quota ideale ip è fuori dai finecorsa software max_str e min_str e ci si vuole allontanare ulteriormente RTE genera l'allarme crash e la IP imposta viene ignorata (viene mantenuta la precedente) * Se la quota ideale ip è sul finecorsa software max_str e min_str e si vuole uscire RTE genera l'allarme crash e la ip imposta viene limitata al paracarro relativo. Finchè la ip sarà fuori dal paracarro non sarà possibile resettare l'allarme. Agendo sulla variabile predefinita ip_write_bounds, cambiandogli il valore di default (0xFFFFFFFF), sarà possibile modificare il comportamento come segue:

© 2025 Robox SpA

Sintassi

```
RULE num
    axes n[,n,n,...,n] | axes_m m
    ref
       ; BLOCCO REF (opzionale)
    end_ref
    motion
        if(first_time())
          ; solo la prima volta
        endif
        ; BLOCCO MOTION
    end_motion
    aux
       ; BLOCCO AUX (opzionale)
    end_aux
END_RULE
```

* Se la quota ideale ip è sul finecorsa software max_str e min_str e si vuole uscire RTE genera l'allarme crash e la ip imposta, anche se fuori dal paracarro viene assunta.

Anche se la ip risulta fuori dal paracarro sarà possibile resettare l'allarme. Verrà rigenerato se si cercherà di allontanarsi ulteriormente.

Viene quindi calcolata da RTE la velocità. Se essa risulta maggiore della velocità massima parametrizzata nelle variabili predefinite max_spe(n) e l'allarme spam è abilitato, viene emesso l'allarme velocità.

Viene calcolata da RTE l'accelerazione. Se essa risulta maggiore della accelerazione massima parametrizzata nella variabile predefinita max_acc(n) e l'allarme acam è abilitato, viene emesso l'allarme accelerazione.

L'allarme Servo (inseguimento) viene gestito se abilitato tramite la variabile predefinita sam.

- SE È ASSEGNATO IV IL COMPORTAMENTO SARÀ IL SEGUENTE:

Se la velocità impostata è maggiore della velocità massima parametrizzata nelle variabile predefinita max_spe(n):

* Viene limitata alla velocità massima stessa (max_spe(n)) se e solo se la variabile predefinita spam è abilitata.

* Viene data segnalazione all'utente tramite la variabile predefinita splm se e solo se la variabile predefinita spam è abilitata.

Viene calcolata da RTE l'accelerazione. Se essa risulta maggiore della accelerazione massima parametrizzata nella variabile predefinita max_acc(n):

© 2025 Robox SpA

Sintassi	**RULE num** **axes** n[,n,n,...,n] | **axes_m** m **ref** ; BLOCCO REF (opzionale) **end_ref** **motion** **if**(first_time()) ; solo la prima volta **endif** ; BLOCCO MOTION **end_motion** **aux** ; BLOCCO AUX (opzionale) **end_aux** **END_RULE**
	* Viene generato allarme accelerazione se e solo se la variabile predefinita acam è abilitata. Viene quindi calcolata da RTE la posizione in base alla formula: ip = ip + iv * si Se la posizione risulta maggiore/minore dei finecorsa software parametrizzati nelle variabili predefinite max_str(n) e min_str(n) e la variabile predefinita cam è abilitata, viene emesso l' allarme crash (lower e upper). L'allarme Servo (inseguimento) viene gestito se abilitato tramite la variabile predefinita sam.
	• SE È ASSEGNATO IA IL COMPORTAMENTO SARÀ IL SEGUENTE: Se l'accelerazione è maggiore della accelerazione massima parametrizzata nelle variabili predefinite max_acc(n): * Viene limitata alla accelerazione massima stessa se e solo se la variabile predefinita acam è abilitata. * Viene data segnalazione all'utente tramite la variabile predefinita acclm se e solo se la variabile predefinita acam è abilitata. Viene quindi calcolata da RTE la velocità in base alla formula: iv = iv + ia * si Se la velocità risulta maggiore della velocità massima parametrizzata nelle variabili predefinite max_spe(n) e l'allarme spam è abilitato, viene emesso l'allarme velocità. Viene quindi calcolata da RTE la posizione in base alla formula = ip + iv * si Se la posizione risulta maggiore/minore dei finecorsa software parametrizzati nelle variabili predefinite max_str(n) e min_str(n) e la variabile predefinita cam

© 2025 Robox SpA

Sintassi	**RULE num** **axes** n[,n,n,...,n] | **axes_m** m **ref** ; BLOCCO REF (opzionale) **end_ref** **motion** **if**(first_time()) ; solo la prima volta **endif** ; BLOCCO MOTION **end_motion** **aux** ; BLOCCO AUX (opzionale) **end_aux** **END_RULE**
	è abilitata, viene emesso l' allarme crash (lower e upper). L'allarme servo (inseguimento) viene gestito se abilitato tramite la variabile predefinita sam.
end_motion	Fine del blocco *motion*.
aux	Nei campi *aux* di ciascuna RULE tipicamente sono espresse tutte quelle condizioni di contorno necessarie per la gestione del processo. L'ordine di scrittura dei blocchi *aux*, *motion* è libero e ne determina l'ordine di esecuzione.
end_aux	Fine del blocco *aux*.
Validità	Rule.
Note	Il blocco *ref*, se presente, deve essere il primo blocco della rule. Non c'è limite al numero di blocchi *aux* all'interno di un blocco rule. Non è possibile avere più di un blocco *motion*, ne più di un blocco *ref*. La dichiarazione delle formule userà le normali istruzioni del linguaggio R3.

Esempi

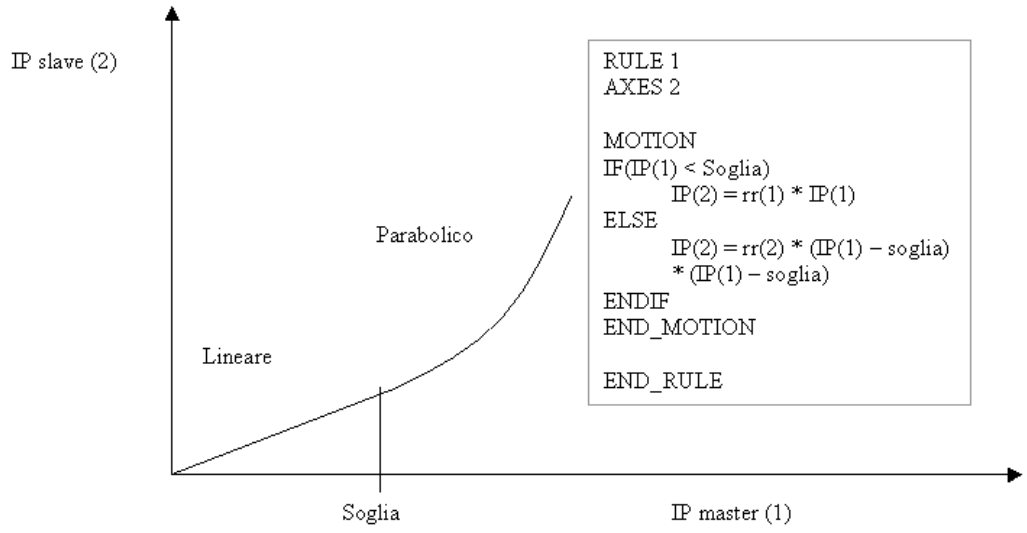
Esempio di utilizzo:

```
rule 7
axes 680(1)
ref 744
    PASSA_1 = epos(1) * pro_gai(1)
```

© 2025 Robox SpA

```
    ; emissione output analogico in funzione dell'errore di posizione
    sref(1) = PASSA_1 + iv(1) * kff(1)
end_ref 708
motion 730
    ; ...
end_motion 707
end_rule 708

rule 10
axes (2)
ref
    ; la tensione di riferimento è pari all'errore di posizione moltip
    sref(2) = epos(2) * pro_gai(2)
end_ref

motion
    ; definizione di tipo di traiettoria
    ; la quota ideale dell'asse 2 deve essere quella dell'asse 1
    ; moltiplicata per il valore contenuto nel registro RR(7)
    ip(2) = rr(7) * cp(1)
end_motion
end_rule
```

Esempio di utilizzo:




```
Errata corrige: ip(2) = (rr(1)*soglia) + (rr(2) * (ip(1)-soglia) *
(ip(1)-soglia))

; Per costruzione della traiettoria significa stabilire
; istante per istante la posizione ideale di un determinato asse.
```

© 2025 Robox SpA

```
; La traiettoria è calcolata realtime (si) così risulta possibile
; implementare qualsiasi tipo di legge di movimento.
```

Istruzione rule_epilogue

Con questa istruzione è possibile specificare il nome di una funzione che verrà eseguita ad ogni battuta delle rule 748 (che girano a frequenza fissa impostata tramite l'istruzione rule_freq) dopo l'esecuzione dei campi ref 744 e dopo le esecuzioni dei campi motion/aux. 730

Sintassi	**rule_epilogue functionName**
functionName	È il nome della funzione da mandare in esecuzione (senza parametri e senza valore di ritorno).
Validità	Rule.
Note	L'istruzione deve essere chiamata una sola volta. Se l'istruzione viene chiamata con functionName = 0 viene disabilitata.

Esempi

Esempio di utilizzo:

```
function 715 rule_init()
; Abilitiamo esecuzione della rule_prologue nella rule di inizializzaz
rule_epilogue(myRuleName)
...
end_fun 706
; funzione rule_epilogue
function 715 myRuleName
...
end_fun 706
```

Istruzione rule_freq

Funzione che permette di impostare da programma la frequenza di esecuzione del file delle rule 748 ($RULE 798).

Sintassi	**rule_freq (int freq)**
freq	valore contenente la frequenza (min: 25 Hz max:5000 Hz) (opzionale, default 100Hz - 10 ms) 25 Hz = 40 ms; 1KHz = 1 ms; 2KHz = 500 micro secondi;
Validità	Rule, Task.
Note	

Gestione della frequenza di esecuzione rules

© 2025 Robox SpA

Esistono tre modi per impostare la frequenza di esecuzione (interrupt) del task delle RULES:

- Nel caso in cui vengano gestite reti canbus e/o EtherCAT, la frequenza di esecuzione delle rules viene impostata pari alla frequenza di scambio dei PDO impostata nei rispettivi editor di configurazione fieldbus. Se si stanno gestendo entrambe le reti, la frequenza deve essere uguale in entrambi gli editor.
- Se non vengono gestiti fieldbus, la frequenza si imposta da configuratore di progetto (vedi SYS_FREQ). Se è presente inoltre il flag di "Frequenza di sistema non modificabile" non sarà possibile una variazione di frequenza da programma tramite l'istruzione rule_freq, e un tentativo di utilizzo di tale istruzione genererà allarme.

Nota: Se la frequenza delle rules impostata nel configuratore è diversa dalla frequenza di scambio dei PDO delle reti canbus e/o EtherCAT, l'impostazione da configuratore di progetto viene ignorata e viene data segnalazione nel report.

- Se non vengono gestiti fieldbus e non è inserito il flag "Frequenza di sistema non modificabile" nel configuratore di progetto, è possibile utilizzare l'istruzione rule_freq per modificare la frequenza.

Esempi

Esempi di utilizzo:

```
; la frequenza delle rule viene impostata a 200 Hz
rule_freq(200)
; la frequenza delle rule viene impostata al valore contenuto nel regi
rule_freq(r(1))
; definisco la costante VALORE_FREQ e le assegno il valore 400
lit 729 VALORE_FREQ 400
; la frequenza delle rule viene impostata al valore della costante VAL
; quindi a 400 Hz
rule_freq(VALORE_FREQ)
```

Istruzione rule_init

Questa funzione speciale, se definita dall'utente all'interno del file delle rules ($RULE 798), viene automaticamente richiamata da RTE una sola volta all'avvio, prima della prima esecuzione delle rules.

Sintassi	**function rule_init()** `; codice utente` ... **end_fun**
Validità	Rule.
Note	La funzione viene eseguita una sola volta. Utile per eseguire inizializzazioni.

Esempi

Implementazione standard di RULE_INIT():

```
function 715 rule_init()
```

© 2025 Robox SpA

```
    ; Definizione della frequenza di battuta delle rules
    rule_freq 756 (500)
    ; Definizione delle funzioni di prologue ed epilogue
    rule_prologue 759 (myRulesPrologue)
    rule_epilogue 756 (myRulesEpilogue)
    ref_prologue 745 (myRefprologue)
    ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    ; Inserire qui tutte le inizializzazioni
    ; delle variabili locali usate nelle rule
    ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
end_fun 706
```

Istruzione rule_periodic

Istruzione che permette di mandare in esecuzione ripetutamente una regola ogni "periodo" impostato. La regola deve essere scritta in un apposito file che inizia con il metacomando $RULE_PERIODIC 798. Questa regola ha priorità minore delle rule 748 sincrone e, di conseguenza, il periodo può essere soggetto a fluttuazioni (gitter).

Sintassi	**rule_periodic (i32 period, stru_int struct)**
period	Intervallo di tempo tra due esecuzioni successive [1 - 1.000.000 ms].
struct	Struttura di tipo stru_int 870 che contiene le informazioni relative (opzionale).
Validità	Rule, Task.
Note	L'istruzione deve essere chiamata una sola volta. Per cambiare il periodo occorre richiamare l'istruzione. Se l'istruzione viene chiamata con valore di periodo 0 viene disabilitata.

Diagnostica

- 2802 Interrupt rule - unknown number < STRU_INT.nRule>
- 9900 ill.Arg A:1 T:x ST:xxx Se il periodo non è nel range specificato [0 -:- 1.000.000 ms]
- 9900 ill.Arg A:2 T:x ST:xxx Se non è stato caricato in RAM il task delle rule asincrone ($rule_int) (Per caricare scrivere comando "load /fa/mytask.cof" in autoexec.stp)
- 9114 Rule Periodic : too long Se l'esecuzione del codice è troppo lunga
- 9115 Rule Periodic : reduced freq.

Esempi

Si vuole creare un task che commuti lo stato di un output digitale ogni 5 minuti:

Nel file mytask.r3:

```
$rule_periodic
```

© 2025 Robox SpA

```
rule 748 1
    out(257) = not 809 out(257)
end rule 708
```

Nel task:

```
; definisco la variabile parPeriodic di tipo STRU_INT 870
STRU_INT 870 parPeriodic
; esecuzione continua
parPeriodic.cntr.0 = 0
; Il totalizzatore è abilitato a contare
parPeriodic.cntr.1 = 0
; regola presente nel file mytask.r3
parPeriodic.nrule = 1
; non si vuole la foto della cp
parPeriodic.maskPhoto = 0x0
...
; condizione di abilitazione della rule periodica
if(rise(inp(2)))
    ; abilito esecuzione rule periodic
    rule_periodic(300000 , parPeriodic)
endif
```

Istruzione rule_prologue

Con questa istruzione è possibile specificare il nome di una funzione che verrà eseguita ad ogni battuta delle rule 748 (che girano a frequenza fissa impostata tramite l'istruzione rule_freq 756) dopo l'esecuzione dei campi ref 744 ma prima dell'esecuzioni dei campi motion/aux. 730

Sintassi	**rule_prologue functionName**
functionName	È il nome della funzione da mandare in esecuzione (senza parametri e senza valore di ritorno).
Validità	Rule.
Note	L'istruzione deve essere chiamata una sola volta. Se l'istruzione viene chiamata con functionName = 0 viene disabilitata.

Esempi

Esempio di utilizzo:

```
function 715 rule_init()
    ; Abilitiamo esecuzione della rule_prologue nella rule di iniziali
    rule_prologue(myRulePrologue)
    ...
end_fun 706
; funzione rule_prologue
function 715 myRulePrologue
```

© 2025 Robox SpA

```
...
end_fun 706
```

Istruzione rule_start_norc

Questa istruzione viene utilizzata con RPE, e si desidera avere anche la funzione rule_prologue, o rule_epilogue, comunque attiva

Normalmente la rule_int viene eseguita alla prima occorrenza di una delle segenti istruzion/keyword:

- gruoup 721
- order 735
- rc

Utilizzando rule_start_norc RTE, anche in assenza delle occorrenze di cui sopra, esegue la rule_init 757. e disabilita l'esecuzione di tutte le rules 748

Sintassi	**rule_start_norc**
Validità	Task.
Note	I campi ref 744 ed end_ref 744 di default non vengono aggiornati se nessuna regola è presente. Volendo riattivare le rules occorrerebbe agire sulla istruzione order (le regole precedente attive verranno rimesse in esecuzione).

Istruzione sb0

Questa istruzione si usa per scrivere il primo byte (a partire da quello meno significativo) dell'argomento con estensione del segno.

Sintassi	**sb0 (i8|i16|i32|u8|u16|u32|real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

© 2025 Robox SpA

Istruzione sb1

Questa istruzione si usa per scrivere il secondo byte (a partire da quello meno significativo) dell'argomento con estensione del segno.

Sintassi	**sb1 (i16|i32|u16|u32|real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Istruzione sb2

Questa istruzione si usa per scrivere il terzo byte (a partire da quello meno significativo) dell'argomento con estensione del segno.

Sintassi	**sb2 (i32|u32|real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Istruzione sb3

Questa istruzione si usa per scrivere il quarto byte (a partire da quello meno significativo) dell'argomento con estensione del segno.

Sintassi	**sb3 (i32|u32|real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

© 2025 Robox SpA

Istruzione sb4

Questa istruzione si usa per scrivere il quinto byte (a partire da quello meno significativo) dell'argomento con estensione del segno.

Sintassi	**sb4 (real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Istruzione sb5

Questa istruzione si usa per scrivere il sesto byte (a partire da quello meno significativo) dell'argomento con estensione del segno.

Sintassi	**sb5 (real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Istruzione sb6

Questa istruzione si usa per scrivere il settimo byte (a partire da quello meno significativo) dell'argomento con estensione del segno.

Sintassi	**sb6 (real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

© 2025 Robox SpA

Istruzione sb7

Questa istruzione si usa per scrivere l'ottavo byte (a partire da quello meno significativo) dell'argomento con estensione del segno.

Sintassi	**sb7 (real arg) = i8 value**
arg	Variabile in cui verrà imposto il valore *value*.
value	Valore da imporre.
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso a un byte non esistente ritorna errore.

Istruzione sdw0

Questa istruzione si usa per scrivere la prima dword (a partire da quella meno significativa) dell'argomento con estensione del segno.

Sintassi	**sdw0 (i16|i32|u16|u32|real arg) = i32 value**
arg	Argomento della Dword.
value	Valore da imporre in arg
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una dword non esistente ritorna un errore.

Istruzione sdw1

Questa istruzione si usa per scrivere la seconda dword (a partire da quella meno significativa) dell'argomento con estensione del segno.

Sintassi	**sdw1 (i32|u32|real arg) = i32 value**
arg	Argomento della Dword.
value	Valore da imporre in arg
Validità	Rule, Task.
Note	Da RTE 33.9.0 l'accesso ad una dword non esistente ritorna un errore.

© 2025 Robox SpA

Istruzione select

Questa istruzione permette di selezionare un gruppo di istruzioni da eseguire in funzione del valore di una variabile.

Sintassi	**SELECT (i32 var)** **CASE const** *; istruzioni R3* **BREAK** **CASE const** *; istruzioni R3* **BREAK** **DEFAULT** *; istruzioni R3* **END_SELECT**
var	Variabile che in funzione del suo valore seleziona quale gruppo di istruzioni eseguire.
const	Il suo valore è l'identificatore numerico del gruppo di istruzioni. Deve essere una costante numerica intera, altrimenti viene generato un allarme alla compilazione.
Validità	Rule, Task.
Note	Si intende per "gruppo di istruzioni" quelle istruzioni che sono comprese tra l'identificatore case e il successivo break, o fra l'identificatore default e l'identificatore end_select 709 (se l'identificatore default è presente). E' possibile avere più identificatori case associati allo stesso break; questo significa che se un gruppo di istruzioni non viene terminato da break, anche le istruzioni successive verranno eseguite fino al primo break (o all'identificatore end_select 709). Se il valore di variabile non soddisfa nessuno dei case specificati (ossia non coincide con nessuno dei valori cost associati agli identificatori case), vengono eseguite le istruzioni del gruppo default. Questo blocco può essere omesso ed in tal caso nessuna istruzione viene eseguita.

Esempi

Esempio di utilizzo con macchina a stati:

```
;-------------------------------------------------------
; Gestioni dipendenti dal modo operativo (rModoOperativo)
;-------------------------------------------------------
select(rModoOperativo)
      ; Gestione mancanza potenza
      case M_MIS_POWER
            call HndMisPower
            break
      ; Gestione attesa cambio modalità
      case M_IDLE
```

© 2025 Robox SpA

```
            call HndIdle
            break
      ; Gestione cicli zero
      case M_CICLO_ZERO
            call HndZeroCycle
            break
      ; Gestione movimenti manuali
      case M_MANUALE
            call HndManual
            break
      ; Gestione posizionamenti iniziali
      case M_POSIZ
            call HndPosiz
            break
      ; Gestione movimentazioni in automatico
      case M_AUTO
            call HndAuto
            break
      ; qui ci arrivo solo se nessun caso precedente
      default
            call HndMisPower
end_select
```

Istruzione ser_con

Imposta i parametri caratteristici del canale seriale di comunicazione specificato.

Sintassi	**ser_con (i32 n, i32 baud, i32 parity, i32 bit, i32 stop)**
n	Numero del canale seriale (1, 2, 3, 4).
baud	Valore di baud rate, da scegliere fra i seguenti: 110 - 150 - 300 - 600 - 1200 - 2400 - 4800 - 9600 - 19200 - 38400 - 57600 - 115200 **Nota**: Da RTE 34.22.01 è stata aggiunta la gestione anche per i seguenti baud rate: 128000 - 256000 - 512000 - 1000000 - 2000000
parity	Tipo di parità, da scegliere fra i seguenti: • E (parità pari) • O (parità dispari) • N (nessun bit di parità)
bit	Numero di bit della parola (valore ammesso: 7, 8).
stop	Numero di bit di stop, da scegliere fra i valori 1 e 2.

© 2025 Robox SpA

Validità	Rule, Task.
Note	Da RTE 34.36.1 è possibile impostare i parametri della comunicazione seriale anche alle porte che comunicano con protocollo BCC3.

Diagnostica

- 9900 ill. Arg A:1 T:x ST:xxx se numero di seriale illegale

Esempi

Esempio di utilizzo:

```
; La porta seriale numero 1 del controllo collegato viene programmata
; frequenza di 9600 Baud
; senza parità
; 8 bit di parola
; 1 bit di stop.
; L'esecuzione di questa istruzione cambia i parametri della porta sel
; esclusivamente sul controllo connesso.
; Non cambia, evidentemente, quelli della porta seriale del PC
; eventualmente connesso al controllo stesso.
ser_con(1, 9600, 'N', 8, 1)
```

Istruzione ser_w_sts

Istruzione che permette di impostare i due segnali DTR e RTS del canale seriale specificato.

Sintassi	**ser_w_sts (i32 nSer, i32 mask, i32 sts)**
nSer	Numero del canale seriale.
mask	Maschera di abilitazione dei bit della word di stato che si intende modificare. Può assumere i seguenti valori:
sts	È il valore dei segnali DTR, RTS che si intende forzare.

Valore	Significato
0	Nessuna modifica abilitata.
1	Abilita modifica per segnale DTR.
2	Abilita modifica per segnale RTS.
4	Abilita modifica per segnali DTR e RTS.

© 2025 Robox SpA

Sintassi	**ser_w_sts (i32 nSer, i32 mask, i32 sts)**
	Forzare un segnale ad 1 significa portarlo nella sua condizione attiva; non si fa quindi riferimento ai livelli fisici dei segnali, ma ai livelli logici.
Validità	Rule, Task.
Note	È possibile modificare lo stato di questi segnali solamente se è stata preventivamente selezionata la modalità MODEM per il canale seriale specificato (si veda l'istruzione ser_mod 592).

Esempi

Esempio di utilizzo:

```
ser_w_sts 766(1, 1, 0)
; per la seriale 1 viene forzato il segnale DTR a 0
; non viene invece influenzato il segnale RTS
```

Istruzione set_float

Questa funzione scrive un FLOAT in un buffer all'offset specificato.

Sintassi	**set_float (buffer, i32 offset, float set_val)**
buffer	può essere un array, struttura , registro
offset	offset in bytes
set_val	il valore FLOAT da scrivere
Validità	Tasks, Rules.
Note	NON viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer. Vedi esempio. Se l'utente desidera scrivere un float nel buffer 'data' di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_set_float(). La funzione speculare è la get_float() 350.

Esempi

Una volta caricato il programma di esempio su un controllo, tramite una shell e' possibile verificare il funzionamento della set_float. Per fare ciò, impostate un valore compreso tra 0 e 127 - sizeof(my_data) alla variabile offset, poi impostate un valore diverso da zero alla variabile set_val e verificare che la variabile my_data sia uguale a set_val.

```
u8 buffer[128]
```

© 2025 Robox SpA

```
i32 offset
float set_val
float my_data

; -------- loop begin --------
__MAIN_LOOP__

   ; range check
   if(offset >= 0 and (offset + sizeof(my_data)) < 128)
      set_float(buffer, offset, set_val)
      my_data = get_float(buffer, offset)
   endif

END_MAIN_LOOP
```

Istruzione set_i16

Questa funzione scrive una word I16 in un buffer all'offset specificato.

Sintassi	**set_i16 (buffer, i32 offset, i16 set_val)**
buffer	può essere un array, una struttura o un registro
offset	offset in bytes
set_val	la word da scrivere
Validità	Tasks, Rules.
Note	NON viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer. Vedi esempio. Se l'utente desidera scrivere una word nel buffer 'data' di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_set_i16(). La funzione speculare è la get_i16() 353.

Esempi

Una volta caricato il programma di esempio su un controllo, tramite una shell e' possibile verificare il funzionamento della set_i16. Per fare ciò, impostate un valore compreso tra 0 e 127 - sizeof(my_data) alla variabile offset, poi impostate un valore diverso da zero alla variabile set_val e verificare che la variabile my_data sia uguale a set_val.

```
u8 buffer[128]
i32 offset
i16 set_val
i16 my_data

; -------- loop begin --------
```

© 2025 Robox SpA

```
__MAIN_LOOP__

   ; range check
   if(offset >= 0 and (offset + sizeof(my_data)) < 128)
      set_i16(buffer, offset, set_val)
      my_data = get_i16(buffer, offset)
   endif

END_MAIN_LOOP
```

Istruzione set_i32

Questa funzione scrive una variabile i32 in un buffer all'offset specificato.

Sintassi	**set_i32 (buffer, i32 offset, i32 set_val)**
buffer	può essere un array, una struttura o un registro.
offset	offset in bytes.
set_val	il valore I32 da scrivere.
Validità	Tasks, Rules.
Note	NON viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer. Vedi esempio. Se l'utente desidera scrivere un i32 nel buffer 'data' di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_set_i32(). La funzione speculare è la get_i32() 354.

Esempi

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della set_i32. Per fare ciò, impostate un valore compreso tra 0 e 127 - sizeof(my_data) alla variabile offset, poi impostate un valore diverso da zero al registro R(1) e verificare che la variabile my_data sia uguale a R(1).

```
u8 buffer[128]
i32 offset
i32 my_data

; -------- loop begin --------
__MAIN_LOOP__

   ; range check
   if(offset >= 0 and (offset + sizeof(my_data)) < 128)
      set_i32(buffer, offset, r(1))
      my_data = get_i32(buffer, offset)
```

© 2025 Robox SpA

```
   endif

END_MAIN_LOOP
```

Istruzione set_i64

Questa funzione scrive in un buffer, all'offset specificato, il valore di una struttura stru_large_int.

Sintassi	**set_i64 (buffer, i32 offset, stru_large_int data)**
buffer	può essere un array, struttura, registro.
offset	offset in bytes.
data	variabile di tipo stru_large_int da scrivere.
Validità	Tasks, Rules.
Note	La funzione speculare è la get_i64() 719.

Esempi

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della set_i64. Per fare ciò, impostate un valore compreso tra 0 e 127 - sizeof(my_data) alla variabile offset, poi impostate un valore diverso da zero alle variabili val_high e val_low, poi verificare che la struttura my_data sia uguale a set_val.

```
u8 buffer[128]
i32 val_high
u32 val_low
stru_large_int set_val
stru_large_int my_data

; -------- loop begin --------
__MAIN_LOOP__

   ; range check
   if(offset >= 0 and (offset + sizeof(my_data)) < 128)
      set_val.high = val_high
      set_val.low  = val_low
      set_i64(buffer, offset, set_val)
      get_i64(buffer, offset, my_data)
   endif

END_MAIN_LOOP
```

© 2025 Robox SpA

Istruzione set_i8

Questa funzione scrive una varaibile i8 in un buffer all'offset specificato.

Sintassi	**set_i8 (buffer, i32 offset, i8 set_val)**
buffer	può essere un array, una struttura o un registro
offset	offset in bytes
set_val	il byte da scrivere
Validità	Tasks, Rules.
Note	NON viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer. Vedi esempio. Se l'utente desidera scrivere un byte nel buffer 'data' di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_set_i8(). La funzione speculare è la get_i8() 355.

Esempi

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della set_i8. Per fare ciò, impostate un valore compreso tra 0 e 127 - sizeof(my_data) alla variabile offset, poi impostate un valore diverso da zero alla variabile set_val e verificare che la variabile my_data sia uguale a set_val.

```
u8 buffer[128]
i8 set_val
i32 offset
i8 my_data

; -------- loop begin --------
__MAIN_LOOP__

   ; range check
   if(offset >= 0 and (offset + sizeof(my_data)) < 128)
      set_i8(buffer, offset, set_val)
      my_data = get_i8(buffer, offset)
   endif

END_MAIN_LOOP
```

© 2025 Robox SpA

Istruzione set_real

Questa funzione scrive un REAL in un buffer all'offset specificato.

Sintassi	**set_real (buffer, i32 offset, real set_val)**
buffer	può essere un array, struttura , registro
offset	offset in bytes
set_val	il valore real da impostare nel buffer
Validità	Tasks, Rules.
Note	NON viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer. Vedi esempio. Se l'utente desidera scrivere un real nel buffer 'data' di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_set_double(). La funzione speculare è la get_real() 355.

Esempi

Una volta caricato il programma di esempio su un controllo, tramite una shell e' possibile verificare il funzionamento della set_real. Per fare ciò, impostate un valore compreso tra 0 e 127 - sizeof(my_data) alla variabile offset, poi impostate un valore diverso da zero alla variabile set_val e verificare che la variabile my_data sia uguale a set_val.

```
u8 buffer[128]
i32 offset
real set_val
real my_data

; -------- loop begin --------
__MAIN_LOOP__

   ; range check
   if(offset >= 0 and (offset + sizeof(my_data)) < 128)
      set_real(buffer, offset, set_val)
      my_data = get_real(buffer, offset)
   endif

END_MAIN_LOOP
```

Istruzione set_u16

Questa funzione scrive una word U16 in un buffer all'offset specificato.

Sintassi	**set_u16 (buffer, i32 offset, u16 set_val)**
buffer	può essere un array, una struttura o un registro

© 2025 Robox SpA

Sintassi	**set_u16 (buffer, i32 offset, u16 set_val)**
offset	offset in bytes
set_val	la word da scrivere
Validità	Tasks, Rules.
Note	NON viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer. Vedi esempio. Se l'utente desidera scriver una word nell buffer 'data' di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_set_u16(). La funzione speculare è la get_u16() 356.

Esempi

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della set_u16. Per fare ciò, impostate un valore compreso tra 0 e 127 - sizeof(my_data) alla variabile offset, poi impostate un valore diverso da zero alla inp_w 200 e verificare che la variabile my_data sia uguale a inp_w 200.

```
u8 buffer[128]
i32 offset
u16 my_data

; -------- loop begin --------
__MAIN_LOOP__

   ; range check
   if(offset >= 0 and (offset + sizeof(my_data)) < 128)
      set_u16(buffer, offset, inp_w(200))
      my_data = get_u16(buffer, offset)
   endif

END_MAIN_LOOP
```

Istruzione set_u32

Questa funzione scrive una variabile u32 in un buffer all'offset specificato.

Sintassi	**set_u32 (buffer, i32 offset, u32 set_val)**
buffer	può essere un array, una struttura o un registro.
offset	offset in bytes.
set_val	il valore U32 da scrivere.
Validità	Tasks, Rules.

© 2025 Robox SpA

Sintassi	**set_u32 (buffer, i32 offset, u32 set_val)**
Note	NON viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer. Vedi esempio. Se l'utente desidera scrivere un u32 nel buffer 'data' di una struttura di tipo bbc_msguser 820 allora deve utilizzare la funzione bcc_set_u32(). La funzione speculare è la get_u32() 357.

Esempi

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della set_u32. Per fare ciò, impostate un valore compreso tra 0 e 127 - sizeof(my_data) alla variabile offset, poi impostate un valore diverso da zero alla variabile set_val e verificare che la variabile my_data sia uguale a set_val.

```
u8 buffer[128]
u32 offset
u32 set_val
u32 my_data

; -------- loop begin --------
__MAIN_LOOP__

   ; range check
   if(offset >= 0 and (offset + sizeof(my_data)) < 128)
      set_u32(buffer, offset, set_val)
      my_data = get_u32(buffer, offset)
   endif

END_MAIN_LOOP
```

Istruzione set_u64

Questa funzione scrive in un buffer, all'offset specificato, il valore di una struttura stru_large_int.

Sintassi	**set_u64 (buffer, i32 offset, stru_large_uint data)**
buffer	può essere un array, struttura, registro.
offset	offset in bytes.
data	variabile di tipo stru_large_int in cui salvare il valore letto.
Validità	Tasks, Rules.
Note	La funzione speculare è la get_u64() 720.

© 2025 Robox SpA

Esempi

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della get_u64. Per fare ciò, impostate un valore compreso tra 0 e 127 - sizeof(my_data) alla variabile offset, poi impostate un valore diverso da zero alle variabili val_high e val_low, poi verificare che la struttura my_data sia uguale a set_val.3

```
u8 buffer[128]
u32 val_high
u32 val_low
stru_large_uint set_val
stru_large_uint my_data

; -------- loop begin --------
__MAIN_LOOP__

   ; range check
   if(offset >= 0 and (offset + sizeof(my_data)) < 128)
      set_val.high = val_high
      set_val.low  = val_low
      set_u64(buffer, offset, set_val)
      get_u64(buffer, offset, my_data)
   endif

END_MAIN_LOOP
```

Istruzione set_u8

Questa funzione scrive una variabile u8 in un buffer all'offset specificato.

Sintassi	**set_u8 (buffer, i32 offset, u8 setVal)**
buffer	Può essere un array, una struttura o un registro.
offset	Offset in bytes.
setVal	Il byte da scrivere.
Validità	Tasks, Rules.
Note	Non viene eseguito il range check, è a cura dell'utente controllare che sommando l'offset e la dimensione del dato non si superi la dimensione massima del buffer. Vedi esempio. Se l'utente desidera scrivere un byte nel buffer 'data' di una struttura di tipo bbc_msguser 820 allora deve utilizzare l'istruzione bcc_set_u8 688 (). La funzione speculare è la get_u8 358 ().

Esempi

© 2025 Robox SpA

Una volta caricato il programma di esempio su un controllo, tramite una shell è possibile verificare il funzionamento della set_u8. Per fare ciò, impostate un valore compreso tra 0 e 127 - sizeof(my_data) alla variabile offset, poi impostate un valore diverso da zero alla variabile set_val e verificare che la variabile my_data sia uguale a set_val.

```
u8 buffer[128]
u8 set_val
i32 offset
u8 my_data

; -------- loop begin --------
__MAIN_LOOP__

   ; range check
   if(offset >= 0 and (offset + sizeof(my_data)) < 128)
      set_u8(buffer, offset, set_val)
      my_data = get_u8(buffer, offset)
   endif

END_MAIN_LOOP
```

Istruzione set_version

Permette di scrivere la versione software dell'applicativo nell'area di memoria che viene visualizzata dal comando "SYSINFO" (da inviare da shell) e dai visualizzatori 4*20.

Sintassi	**set_version (i32 version, i32 release, i32 build, string desc)**
version	Valore contenente il numero di versione.
release	Valore contenente il numero di release.
build	Valore contenente il numero di build.
descr	Testo descrittivo di 32 caratteri massimo.
Validità	Rule, Task.
Note	

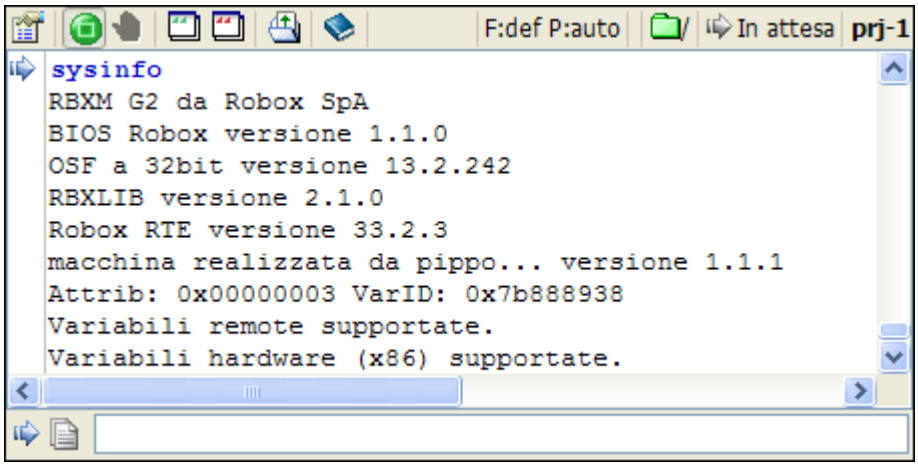
Esempi

Se inserisco l'istruzione:

```
set_version(1,1,1,"macchina realizzata da pippo....")
```

Con in comando sysinfo visualizzo:

© 2025 Robox SpA



Istruzione sw0

Questa istruzione si usa per scrivere la prima word (a partire da quella meno significativa) dell'argomento con estensione del segno.

Sintassi	**sw0 (i16|i32|u16|u32|real arg) = i16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Note	

Istruzione sw1

Questa istruzione si usa per scrivere la seconda word (a partire da quella meno significativa) dell'argomento con estensione del segno.

Sintassi	**sw1 (i32|u32|real arg) = i16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Note	

© 2025 Robox SpA

Istruzione sw2

Questa istruzione si usa per scrivere la terza word (a partire da quella meno significativa) dell'argomento con estensione del segno.

Sintassi	**sw2 (real arg) = i16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Note	

Istruzione sw3

Questa istruzione si usa per scrivere la quarta word (a partire da quella meno significativa) dell'argomento con estensione del segno.

Sintassi	**sw3 (real arg) = i16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Note	

Istruzione terminator

Questa istruzione imposta un altro terminatore attivo in aggiunta al terminatore standard di default LF.

Sintassi	**terminator (i32 num, i32 code)**
num	Individua il dispositivo di ingresso.
code	Rappresenta il codice AscII del terminatore che si vuole impostare.
Validità	Rule, Task.
Note	

Diagnostica

- Genera allarme 9900 ill. Arg A:1 T:x ST:xxx se numero di seriale illegale

Esempi

Esempio di utilizzo:

© 2025 Robox SpA

```
; Imposta ETX (3) come terminatore.
; Le stringhe potranno essere terminate sia da ETX che da LF.
terminator(1, 3)
; Imposta LF come unico terminatore.
; Il terminatore viene usato solo se in "modalità normale" (vedi ser_m
terminator(1, 10)
```

Istruzione tm_et

Questa istruzione si usa per impostare il valore corrente (Elapsed Time) del timer.

ATTENZIONE: Il valore è impostabile solo prima che venga fatto partire il timer (es. prima di chiamare la funzione tm_int 779() per la prima volta), altrimenti viene generato l'allarme 9901.

Sintassi	**tm_et (timer name, real valueEt)**
name	Nome del timer di tipo timer 930.
valueEt	Valore del tempo, in millisecondi, da impostare.
Validità	Rule, Task.
Note	

Istruzione tm_int

Questa istruzione funge da timer per ritardare un fronte di salita di un tempo predefinito.

Funzionamento

Viene creato un timer integrale.

Il tempo trascorso (ET) inizia l'evoluzione non appena si ha un fronte di salita dell'ingresso *start* e continua ad incrementare fino a che l'ingresso rimane alto.

L'uscita (Q) va in stato di ON dopo che il tempo trascorso (ET) raggiunge il periodo di tempo preset specificato (PT).

Al contrario del tm_on 782() il conteggio non viene resettato dall'assenza dell'ingresso, ma solo dopo un comando di reset; in questo modo il tempo trascorso è pari alla somma dei tempi di abilitazione.

Sintassi	**tm_int (timer name, bool start)**
name	Nome del timer di tipo timer 930.
start	Ingresso di avviamento timer (fronte di salita). Può essere una qualsiasi condizione, anche un'espressione (es. cp(1) > 1000).
Validità	Rule, Task.

© 2025 Robox SpA

Sintassi	**tm_int (timer name, bool start)**
Note	I timer non sono ritentivi.

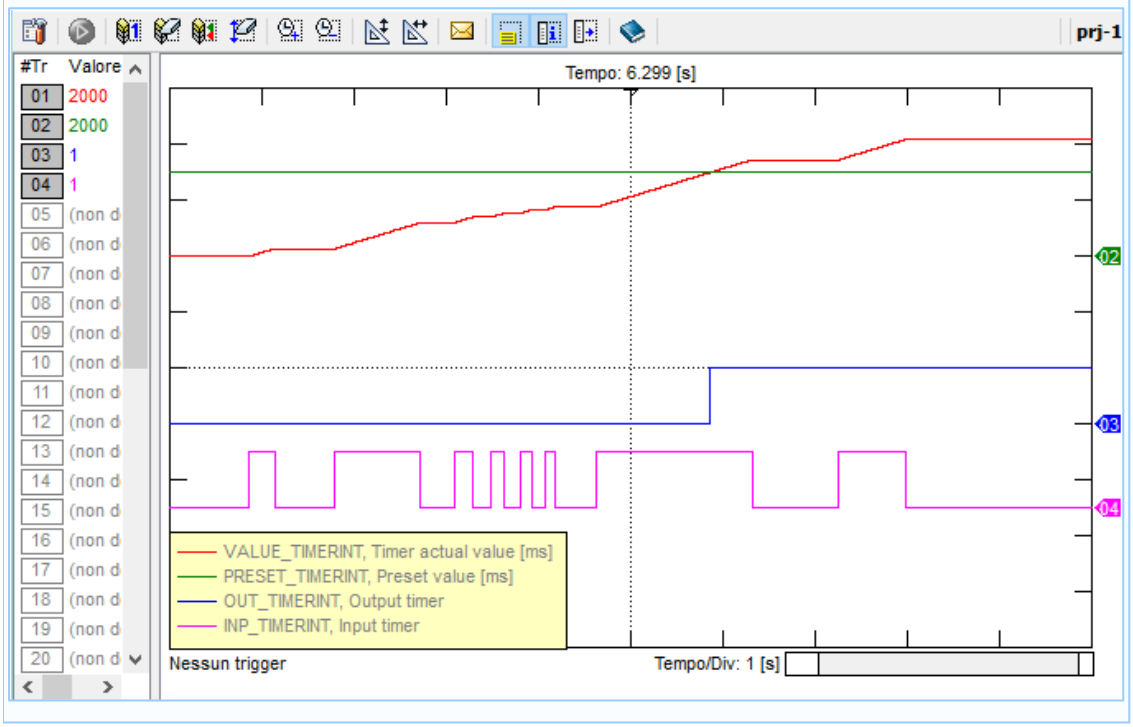
Diagnostica

- Genera allarme 9901 ill. fun. se il timer viene utilizzato precedentemente con un altro tipo di timer

Esempi

Esempio di utilizzo:

```
timer 930 T1
tm_pt 643(T1, 1000)      ; settiamo il preset time ad 1 sec
tm_int 779(T1, inp(1))   ; se inp 1 vero il timer incrementa l'elapsed t
out(257) = tm_q 645(T1)  ; arrivato al valore di preset l'esito va in ON
et = tm_et 779(T1)       ; tempo trascorso
_if(inp(2))
    tm_reset 784(T1)     ; resetta l'elapsed time
```



Istruzione tm_of

Questa istruzione funge da timer per ritardare un fronte di discesa di un tempo predefinito.

Funzionamento

© 2025 Robox SpA

Sul fronte di salita dell'ingresso *start*, l'uscita (Q) va in stato di ON.

Il tempo trascorso (ET) inizia l'evoluzione non appena si ha un fronte di discesa dell'ingresso *start* e continua ad incrementare fino al fronte di salita dell'ingresso stesso, momento in cui viene imposto uguale a 0.

L'uscita (Q) rimane in stato di ON fino a che il tempo trascorso (ET) non raggiunge il periodo di tempo preset specificato (PT).

Sintassi	**tm_of (timer name, bool start)**
name	Nome del timer di tipo timer 930.
start	Ingresso di avviamento timer (fronte di salita). Può essere una qualsiasi condizione, anche un'espressione (es. cp(1) > 1000).
Validità	Rule, Task.
Note	I timer non sono ritentivi.

Diagnostica

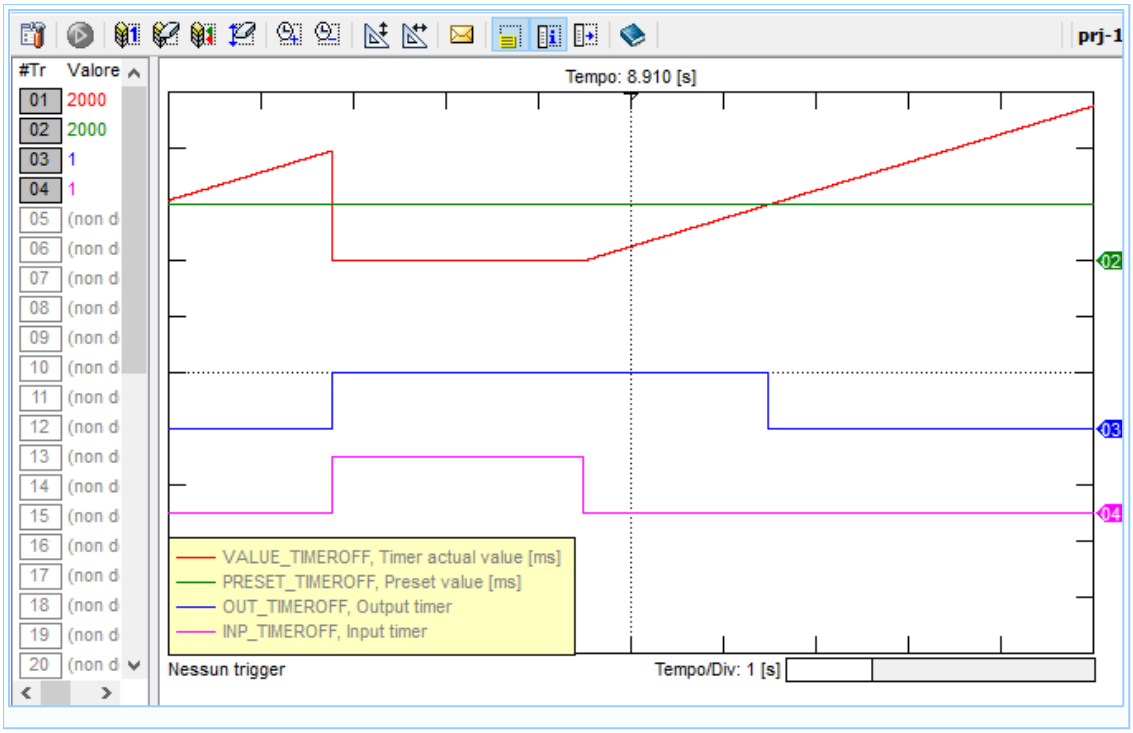
- Genera allarme 9901 ill. fun. se il timer viene utilizzato precedentemente con un altro tipo di timer

Esempi

Esempio di utilizzo:

```
timer 930 T1
tm_pt 643(T1, 1000)      ; settiamo il preset time ad 1 sec
tm_of 780(T1 , inp(1))   ; su salita input 1 (pulsante ADV su CPU) esito
out(257) = tm_q 645(T1)  ; l'esito del timer rimane ON per 1 secondo in a
et = tm_et 779(T1)       ; tempo trascorso
```

© 2025 Robox SpA



Istruzione tm_on

Questa istruzione funge da timer per ritardare un fronte di salita di un tempo predefinito.

Funzionamento

Il tempo trascorso (ET) inizia l'evoluzione non appena si ha un fronte di salita dell'ingresso *start* e continua ad incrementare fino al fronte di discesa dell'ingresso stesso, momento in cui viene imposto uguale a 0.

L'uscita (Q) va in stato di ON dopo che il tempo trascorso (ET) raggiunge il periodo di tempo preset specificato (PT).

Sintassi	**tm_on (timer name, bool start)**
name	Nome del timer di tipo timer 930.
start	Ingresso di avviamento timer (fronte di salita). Può essere una qualsiasi condizione, anche un'espressione (es. cp(1) > 1000).
Validità	Rule, Task.
Note	I timer non sono ritentivi. In caso di approccio programmatico tm_on() va inizializzato e settato a false prima del main loop.

Diagnostica

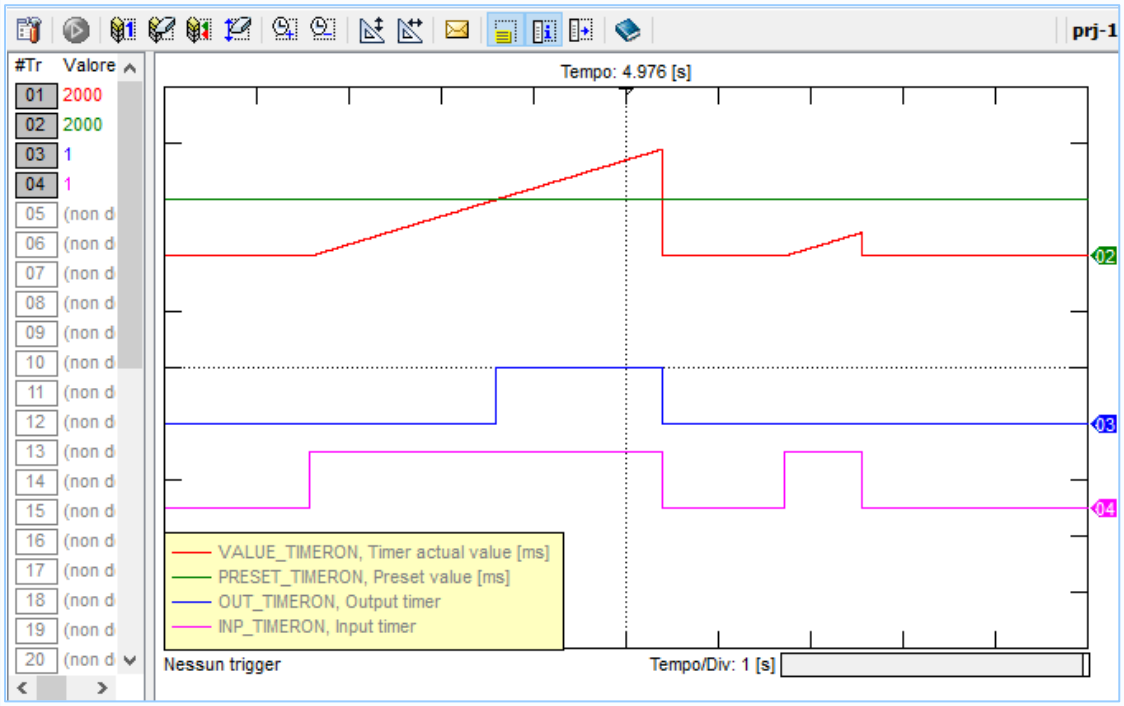
© 2025 Robox SpA

- Genera allarme 9901 ill. fun. se il timer viene utilizzato precedentemente con un altro tipo di timer

Esempi

Esempio di utilizzo:

```
timer 930 T1
tm_pt 643(T1, 1000)      ; settiamo il preset time ad 1 sec
tm_on 782(T1, inp(1))    ; su salita input 1 (pulsante ADV su CPU) inizia
out(257) = tm_q 645(T1)  ; attiviamo l'uscita 257 quando l'esito del time
et = tm_et 779(T1)       ; tempo trascorso
```



Esempio di utilizzo:

NOTA: In caso di approccio programmatico tm_on() va inizializzato e settato a false prima del main loop.

```
timer T1
tm_pt(T1, 2000)     ; settiamo il preset time a 2 sec
tm_on(T1, false)    ; settiamo a false
; =============
; MAIN LOOP
; =============
```

© 2025 Robox SpA

Istruzione tm_pt

Questa istruzione si usa per impostare il valore di preset del timer.

Sintassi	**tm_pt (timer name, real presetTm)**
name	Nome del timer di tipo timer 930.
presetTm	Valore del preset, in millisecondi, da impostare.
Validità	Rule, Task.
Note	

Istruzione tm_pt2

Questa istruzione si usa per impostare il valore del secondo preset del timer quadrato.

Sintassi	**tm_pt2 (timer name, real presetTm)**
name	Nome del timer di tipo timer 930.
presetTm	Valore del preset, in millisecondi, da impostare.
Validità	Rule, Task.
Note	

Istruzione tm_reset

Questa istruzione si usa per resettare il totalizzatore di tempo (ET) di un timer.

Può venire utile se si usano i timer di tipo integrale tm_int 779().

Sintassi	**tm_reset (timer name)**
name	Nome del timer di tipo timer 930.
Validità	Rule, Task.
Note	

Istruzione tm_sqw

Questa istruzione funge da timer per generare un'onda quadra.

Funzionamento

Genera un'onda quadra in uscita della quale l'utente imposta la durata dello stato alto (Q=1) e di quello basso (Q=0) tramite (PT) e (PT2), finché il segnale di abilitazione (EN) è posto a 1; quando questo segnale viene posto a 0 il timer si interromperà fino al successivo cambio di stato.

© 2025 Robox SpA

Sintassi	**tm_sqw (timer name, bool start)**
name	Nome del timer di tipo timer.
start	Ingresso di avviamento del timer (fronte di salita). Può essere una qualsiasi condizione, anche un'espressione (es. cp(1) > 1000).
Validità	Rule, Task.
Note	I timer non sono ritentivi.

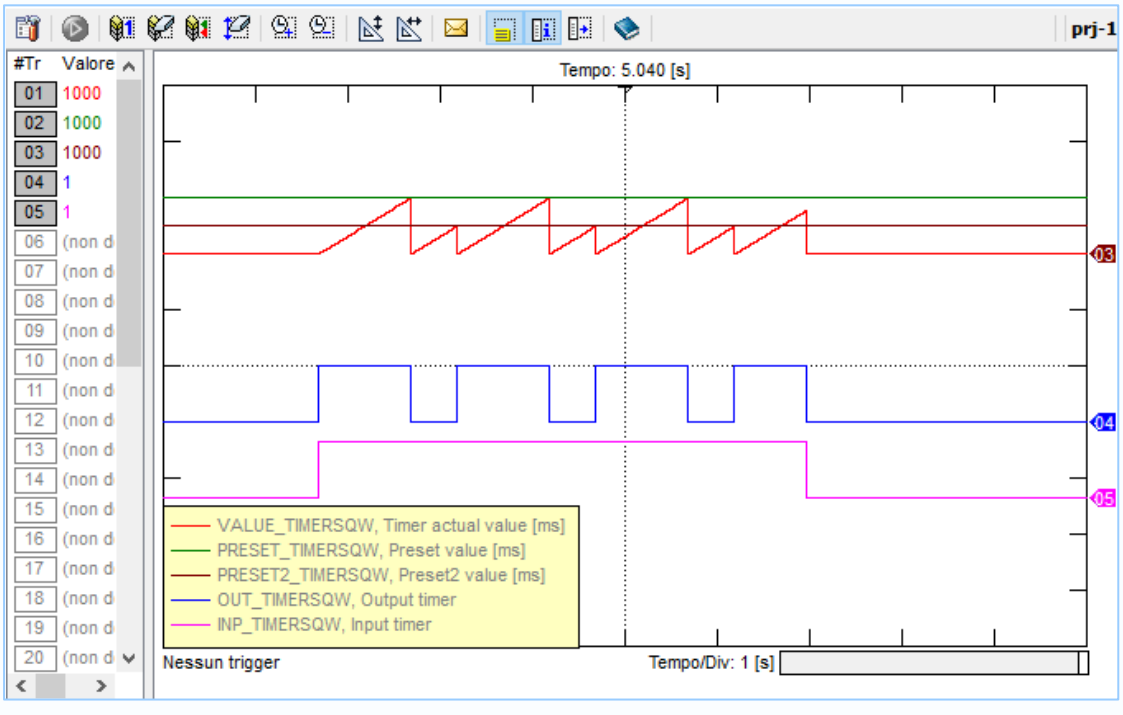
Diagnostica

- Genera allarme 9901 ill. fun. se il timer viene utilizzato precedentemente con un altro tipo di timer

Esempi

Esempio di utilizzo:

```
timer T1
tm_pt(T1, 1000)      ; settiamo il preset time ad 1 sec
tm_pt2(T1, 2000)     ; settiamo il secondo preset time a 2 sec
tm_sqw(T1, inp(1))   ; su salita input 1 (pulsante ADV su CPU) inizia
out(257) = tm_q(T1)  ; attiviamo per 1 sec l'uscita 257
et = tm_et(T1)       ; tempo trascorso
```



© 2025 Robox SpA

Istruzione tm_tp

Questa istruzione funge da timer per generare un impulso di durata predefinita.

Funzionamento

Sul fronte di salita dell'ingresso *start*, l'uscita Q rimane in stato di ON per il periodo di tempo preset specificato (PT) anche se l'ingresso dovesse subire un nuovo fronte di salita.

Il tempo trascorso (ET) inizia l'evoluzione non appena si ha un fronte di salita dell'ingresso *start* e continua ad incrementare fino al fronte di discesa dell'ingresso stesso, momento in cui viene imposto uguale a 0 (comunque non prima che il tempo (PT) sia terminato).

Sintassi	**tm_tp (timer name, bool start)**
name	Nome del timer di tipo timer 930.
start	Ingresso di avviamento del timer (fronte di salita). Può essere una qualsiasi condizione, anche un'espressione (es. cp(1) > 1000).
Validità	Rule, Task.
Note	I timer non sono ritentivi.

Diagnostica

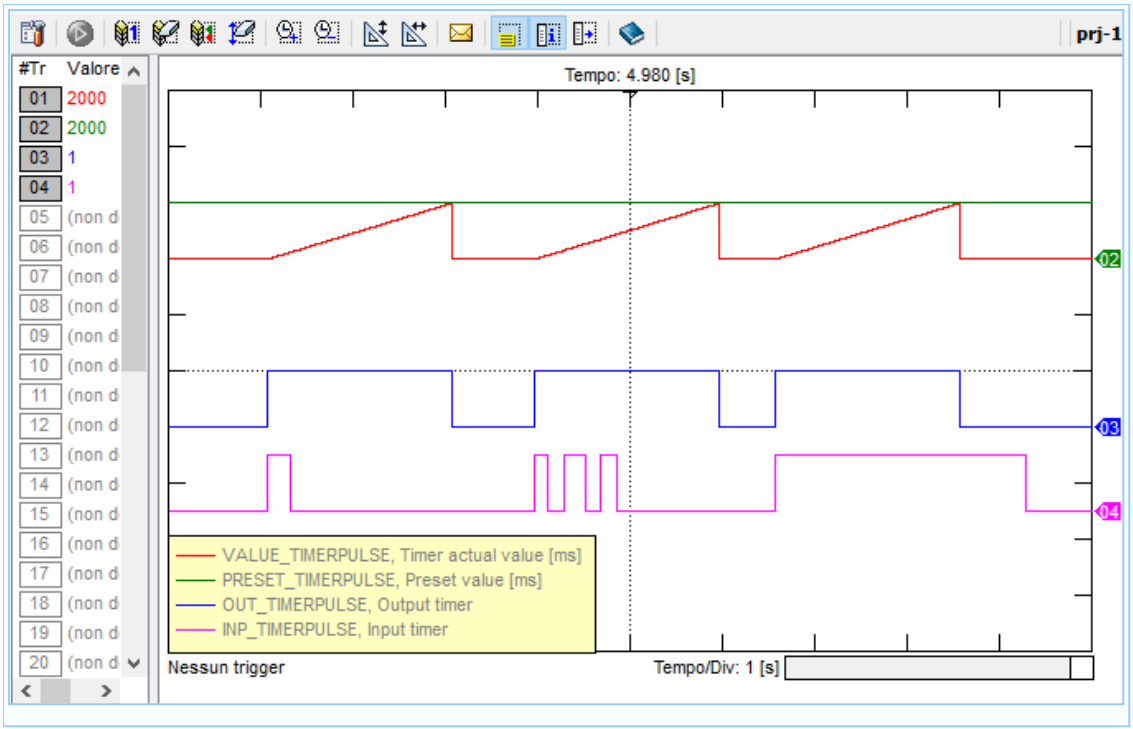
- Genera allarme 9901 ill. fun. se il timer viene utilizzato precedentemente con un altro tipo di timer

Esempi

Esempio di utilizzo:

```
timer 930 T1
tm_pt 643(T1, 1000)      ; settiamo il preset time ad 1 sec
tm_tp 786(T1, inp(1))    ; sullo stato alto dell' input 1 (pulsante ADV s
out(257) = tm_q 645(T1)  ; l'esito del timer sta in ON per 1 secondo
et = tm_et 779(T1)       ; tempo trascorso
```

© 2025 Robox SpA



Istruzione w0

Questa istruzione si usa per scrivere la prima word (a partire da quella meno significativa) dell'argomento.

Sintassi	**w0 (i16|i32|u16|u32|real arg) = u16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Nota	Questa istruzione non può essere usata per impostare il valore di un registro (es. w0(r(1))=val).

Istruzione w1

Questa istruzione si usa per scrivere la seconda word (a partire da quella meno significativa) dell'argomento.

Sintassi	**w1 (i32|u32|real arg) = u16 value**
arg	Argomento della word.
value	Valore da imporre.

© 2025 Robox SpA

Sintassi	**w1 (i32|u32|real arg) = u16 value**
Validità	Rule, Task.
Nota	Questa istruzione non può essere usata per impostare il valore di un registro (es. w1(r(1))=val).

Istruzione w2

Questa istruzione si usa per scrivere la terza word (a partire da quella meno significativa) dell'argomento.

Sintassi	**w2 (real arg) = u16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Nota	Questa istruzione non può essere usata per impostare il valore di un registro (es. w2(r(1))=val).

Istruzione w3

Questa istruzione si usa per scrivere la quarta word (a partire da quella meno significativa) dell'argomento.

Sintassi	**w3 (real arg) = u16 value**
arg	Argomento della word.
value	Valore da imporre.
Validità	Rule, Task.
Nota	Questa istruzione non può essere usata per impostare il valore di un registro (es. w3(r(1))=val).

© 2025 Robox SpA

Istruzione while

Questo costrutto permette di ripetere l'esecuzione del gruppo di istruzioni comprese fra while e end_while finchè la condizione definita nell'istruzione si mantiene vera.

Sintassi	**while (condizione)** `; istruzioni R3` ... **end_while**
condizione	Può essere una generica espressione matematica, il cui risultato viene assunto come falso se vale 0, altrimenti come vero.
Validità	Rule, Task.
Note	É consentito l'uso del break.

Esempi

Esempio di utilizzo con condizione:

```
; inizio del costrutto while ... end_while
; il ciclo viene ripetuto finché il valore di r(3) è minore di 10
while(r(3) < 10)
    ; istruzioni R
    ...
    r(3) = r(3) + 1
; fine del costrutto while ... end_while
end_while
```

Esempio di ciclo infinito:

```
; ciclo infinito
while(1)
    ; istruzioni R
end_while.
```

Esempio di uso con break:

```
while(1)
    ; istruzioni R
    ...
    if (MANUALE)
        break
    endif
    ; se manuale è vero [<> 0] qui non ci si passa mai!
    ...
end_while
```

© 2025 Robox SpA

Grammatica

Generalità grammatica

Un programma R3 è scritto in formato libero. Le linee di istruzioni possono essere incolonnate nel modo preferito, con un numero indefinito di spazi.

Ogni linea, separata da un'altra linea dai caratteri CR LF (tasto ENTER), deve contenere una sola istruzione. Un'istruzione può essere estesa su più linee, purché si inserisca al termine di ciascuna linea il carattere "&" (ampersand), per indicare al compilatore che l'istruzione non è terminata.

Un carattere "&" viene considerato come segno di prosecuzione dell'istruzione solo se si trova alla fine della riga (cioè se non è seguito da nessun carattere). In caso contrario "&" viene trattato come facente parte dell'istruzione.

Il carattere ";" (punto e virgola) significa commento: una riga preceduta da questo carattere rimane nel listato, ma viene ignorata dal compilatore R3.

Esempio:

```
; istruzione su una sola linea
r(27) = 123
```

Esempio:

```
; stessa istruzione estesa su due linee
r(27) = &
123
```

Un programma scritto in linguaggio R3 appare come una sequenza di frasi che devono essere scritte rispettando certe regole grammaticali e sintattiche.

In queste frasi figureranno delle parole riservate al linguaggio stesso (keyword), dei nomi di variabili, delle parole aventi significato solamente per l'utilizzatore ed evidentemente dei numeri.

Il set di caratteri che vengono accettati dal compilatore R3, per le keyword ed i nomi di variabili, è il seguente:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

0 1 2 3 4 5 6 7 8 9

Più alcuni caratteri speciali:

= . + - * / , : [ ] ( ) " $ _ ; & # spazio tabulatore

I seguenti caratteri speciali sono riservati agli sviluppi Robox:

@ % ! ^ ' ? | \ ~

Il compilatore non fa nessuna differenza tra lettere minuscole e maiuscole per le keyword e i nomi di variabili (ovvero definito case-insensitive).

Esempio:

```
; Queste tre parole sono corrette ed equivalenti
OUTPUT
Output
oUtPuT
```

In questo manuale, per ragioni di leggibilità, tutte le parole chiave del linguaggio R3 sono scritte in minuscolo per distinguerle dai tipi di dati, che invece vengono messi in maiuscolo.

© 2025 Robox SpA

Identificatori

Gli identificatori sono usati per nominare le variabili, le subroutine, le destinazioni di salto (label).

Il primo carattere di un identificatore deve essere alfabetico. I caratteri successivi possono essere alfabetici o numerici.

Inoltre è permesso l'uso del tratto basso (_). Non ci sono limiti particolari sul numero di caratteri utilizzabili.

Esempio:

```
; dichiarazione dell'identificatore PIPPO associato a una variabile inte
INT pippo
; dichiarazione dell'identificatore PLUTO associato a una label (nome su
pluto:
```

Non è possibile usare come identificatori le parole riservate del linguaggio R3.

Introduzione

Generalità linguaggio R3

L'automazione flessibile ha trovato nei linguaggi di programmazione il suo mezzo più potente di espressione. Questi linguaggi si sono affermati soprattutto per la programmazione di macchine utensili, di trasferte flessibili, di robot, macchine automatiche ecc.

Esiste però tutta una categoria di macchine, caratterizzate da molti assi che ruotano in continuo e che devono essere sincronizzati secondo leggi di movimento più o meno complicate, che ancora non ha potuto convenientemente godere della flessibilità caratteristica dell'automazione di oggi. Ci riferiamo, ad esempio, ai seguenti campi di applicazione:

- macchine per la stampa
- macchine tessili
- macchine per la lavorazione del metallo
- macchine per la lavorazione della carta
- macchine per la lavorazione del legno
- in generale controlli di sincronismo

In questi campi dominano ancora le soluzioni di tipo meccanico, come, ad esempio:

- camme di profilo opportuno per la memorizzazione di una certa legge di movimento
- tamburi con camme e micro per la memorizzazione di programmi
- sincronismi realizzati meccanicamente, ossia con alberi principali dal cui movimento, tramite ingranaggerie, vengono derivati tutti i movimenti per la macchina
- leverismi più o meno geniali
- ecc.

Le soluzioni elettroniche sono poi, normalmente, di tipo analogico, o derivate dall'analogico, a meno che la soluzione digitale risulti irrinunciabile; per esempio nel controllo in sincronismo di due motori asserviti.

Evidentemente, in queste condizioni, la flessibilità degli impianti, ossia la capacità di adeguarsi a nuove esigenze produttive, risulta nulla e, in ogni caso, legata ad operazioni

© 2025 Robox SpA

meccaniche lunghe e soprattutto costose. Costose in se stesse e costose soprattutto in termini di produzione mancata.

Analogamente impossibile risulta una conduzione centralizzata della produzione sia nel suo aspetto gestionale, sia nel suo aspetto diagnostico.

Obiettivi

La Robox SpA, forte dell'esperienza nel controllo assi acquisita in robotica, ha realizzato il linguaggio R3 per la programmazione dei cicli di lavoro di queste macchine.

Il linguaggio R3 realizza molteplici obiettivi, come ad esempio:

- rende utilizzabili le apparecchiature Robox di controllo assi, sviluppate per la robotica, nel campo più generale del controllo dei movimenti e dei sincronismi su assi che possono anche girare in continuo
- rende risolubili da software tutti i problemi
- rende possibile e facile l'inserimento delle macchine così controllate nella "fabbrica automatica".

Queste macchine, in generale, sono caratterizzate dal numero di assi controllati, intendendosi per "asse" una parte della macchina il cui moto può essere governato in modo indipendente mediante un attuatore.

L'attuatore sarà tipicamente un motore in CC, un motore brushless, un motore rotativo idraulico, ma anche un motore ad induzione, un cilindro idraulico, ecc.

Un esempio tipico è quello di un programma di trasformazione di coordinate di uno o più assi, il quale deve eseguire una serie di calcoli con cadenza fissa, ad es. ogni 2 ms, prendendo le informazioni dal processore master e restituendo i risultati in tempo reale.

Vantaggi dei linguaggi

Affrontare i problemi mediante programmazione, ossia scrivendo del software, offre innumerevoli vantaggi.

Ricordiamo i principali:

- la descrizione delle problematiche, in questo caso leggi di movimento e cicli operativi, è veloce, semplice e documentata
- i tempi intercorrenti fra l'inizio della progettazione di una macchina e la messa a punto vengono ridotti drasticamente
- le parti meccaniche delle macchine hanno un grado di standardizzazione molto più elevato
- vengono eliminati particolari meccanici di difficile lavorazione
- l'affidabilità della macchina aumenta, congiuntamente alla diminuzione delle complicazioni meccaniche (leonardismi)
- errori di progettazione vengono rapidamente e facilmente eliminati
- è assai facile ottimizzare i modi di funzionamento delle macchine e quindi migliorare il funzionamento o adattarsi a nuove esigenze
- è assai facile introdurre modifiche.

Praticamente inesistenti gli svantaggi. L'unico oggettivo svantaggio che può ancora esistere è la necessità di dover cambiare il modo di progettare e quindi di dover investire tempo nell'apprendere i nuovi strumenti di lavoro: computer, linguaggi di programmazione, editori, trasmissioni seriali, compilatori, ecc.; nonché di dover assimilare tutto un nuovo lessico fatto di software, listing, bugs, files, debug, ecc.

Si tratta comunque di un passo che, prima o dopo, deve essere fatto da tutti, e che quindi va fatto subito, dato che i tempi sono più che maturi.

© 2025 Robox SpA

Programmazione

Programmare in linguaggio R3 i movimenti delle macchine presenta molte analogie con il programmare un PLC. In entrambi i casi il progettista dovrà:

- scrivere un programma sorgente, cioè un elenco delle operazioni che la macchina deve svolgere, scritto in un opportuno linguaggio (linguaggio R3 nel caso Robox). È questo il momento della progettazione, cioè il momento creativo
- compilare il programma sorgente mediante l'apposito compilatore. Il compilatore, ricordiamo, traduce il programma sorgente, scritto in un linguaggio vicino all'uomo, in un programma oggetto, scritto in codici numerici comprensibili dall'apparecchiatura elettronica che controlla la macchina
- caricare il programma oggetto nell'apparecchiatura elettronica, operazione che normalmente è fatta utilizzando le porte di comunicazione seriali o Ethernet
- verificare il funzionamento della macchina ed eventualmente apportare modifiche al programma sorgente.

La differenza fondamentale fra il linguaggio R3 ed il linguaggio di programmazione ladder è che quest'ultimo è stato concepito per la programmazione di grandezze di tipo attacca/stacca, cioè, per esempio, tele-interruttori che si aprono e si chiudono secondo i segnali che provengono da pulsanti, micro interruttori, ecc. che, a loro volta, possono essere aperti o chiusi.

Il linguaggio R3, che è stato concepito per regolare grandezze continue come tipicamente posizioni, velocità, coppie, ma anche pressioni, temperature, ecc., accetta e gestisce equazioni che correlino variabili di questo tipo.

Praticamente non c'è limite alla sofisticazione delle operazioni: il linguaggio R3 gestisce integrali e derivate, moltiplicazioni, divisioni, seni, coseni, ecc.

Evidentemente il linguaggio R3 sarà anche in grado di governare fenomeni di tipo attacca/stacca come un PLC, cioè di risolvere equazioni di tipo booleano, ma questa è certamente un'attività secondaria, rispetto all'attività principale, che è volta a governare con precisione, dolcezza, rapidità e obbedendo a leggi di movimento più o meno complicate, le predette grandezze cinematiche e non.

L'estensione del file sorgente R3 deve essere .R3

L'estensione dei file di inclusione deve essere .I3

La programmazione con il linguaggio R3 avviene in tre passaggi.

Rules

Una prima fase di programmazione riguarda la descrizione delle leggi di movimento degli assi. Ad esempio, nel caso di un motore che debba ruotare in perfetto sincronismo con un secondo motore (asse elettrico), la descrizione della legge di movimento sarà semplicemente:

```
IP2 = IP1
```

cioè la posizione dell'albero del secondo motore deve essere identica a quella del primo motore (e conseguentemente le velocità coincideranno con errore assolutamente nullo).

Se, viceversa, due motori dovranno avere velocità identiche a meno di un fattore K, l'equazione sarà:

```
IV2 = IV1 * K
```

cioè la velocità IV2 dovrà coincidere con la velocità IV1 a meno del fattore K, e K potrà essere un qualunque numero reale.

Nel caso poi che si debba controllare un aspo avvolgente con un ballerino, si scriverà:

© 2025 Robox SpA

```
E = P - P0
INT_E = INT_E + E
IV1 = KP * E + KI * INT_E
```

L'equazione numero 3. dice che l'errore del ballerino è la differenza fra la posizione attuale (cioè P) e la posizione di mezzeria (cioè P0).

L'equazione numero 5. dice che la velocità dell'asse 1 deve essere uguale a KP volte l'errore del ballerino (KP = guadagno proporzionale) + KI volte l'integrale dell'errore del ballerino (KI = guadagno integrale).

Le equazioni 1, 2, 3, 4 e 5 vengono denominate, nel linguaggio R3, "rules" (regole) e dovranno essere verificate con frequenza sufficientemente alta nel tempo, cioè "in gergo" dovranno essere onorate con continuità da un "interrupt" a tempo. Le regole non saranno fisse, ma varieranno secondo i cicli operativi della macchina.

Rules asincrone

Sono inoltre gestite delle rules invocate in modo asincrono: l'invocazione può essere effettuata o tramite variazioni dello stato di un input digitale a frequenza programmabile oppure a predizione di posizione.

La gestione di queste rules e le rules stesse vengono eseguite in modo da non essere interrotte da nessun evento. Per il buon funzionamento del sistema stesso, l'utente dovrà avere la massima cura nello scrivere queste rules, in modo che la loro esecuzione sia il più breve possibile.

Cicli operativi (main)

Una seconda programmazione riguarda la descrizione dei cicli operativi e delle funzioni ausiliarie.

Per esempio, volendo programmare che, se la variabile booleana "avanti" è vera, venga eccitata la elettrovalvola EVa o, in caso contrario, l'elettrovalvola EVi si scriverà:

```
...
  if (avanti)
    out(EVa)
  else
    out(EVi)
  endif
...
```

iin questo caso, le istruzioni non dovranno essere eseguite a cadenza fissa, come nel caso delle rules, ma ogni qualvolta il flusso del programma le incontrerà.

Categorie di linguaggi

Il linguaggio R3, come già accennato, è un linguaggio compilato. Il programma sorgente viene tradotto dal compilatore in programma oggetto e il programma oggetto, a sua volta, deve essere immesso nella memoria dell'apparecchiatura.

I vantaggi del linguaggio compilato sono molteplici; per meglio chiarirli, ricordiamo che i linguaggi possono essere suddivisi in tre categorie:

- linguaggi interpretati
- linguaggi compilati

© 2025 Robox SpA

- linguaggi misti

Linguaggi interpretati

Nei linguaggi interpretati le varie istruzioni, scritte normalmente in formato AscII, vengono lette e decodificate al momento dell'esecuzione. L'esecuzione avverrà con la chiamata di opportune routines presenti nella memoria (normalmente firmware) dell'apparecchiatura.

Il vantaggio dei linguaggi interpretati è la maggior semplicità di uso, l'immediata eseguibilità e la maggiore facilità di debug.

Lo svantaggio è la maggiore lentezza di esecuzione, la più grande facilità di commettere errori, la mancanza di un "file" list.

Linguaggi compilati

Nei linguaggi compilati le varie istruzioni, scritte ancora in formato AscII, vengono tradotte in codici oggetto del microprocessore che governa l'apparecchiatura ed immesse nella sua memoria.

I vantaggi dei linguaggi compilati sono fra gli altri:

- permettono la massima rapidità di esecuzione del programma al momento dell'esecuzione. Questa caratteristica è estremamente importante nel caso R3 per la risoluzione in tempo reale di equazioni anche assai complicate
- generano un file avente estensione .LST (il file list) nel quale vengono individuati e documentati tutti gli errori formali e vengono date tutte le informazioni utili per la successiva messa a punto del programma
- assicurano una completa ed esaustiva documentazione del programma.

Linguaggi misti

Nei linguaggi misti le varie istruzioni vengono tradotte in codici intermedi, che non sono ancora i codici oggetto del microprocessore che governa l'apparecchiatura, ma che sono interpretati ed eseguiti dal software residente.

I linguaggi misti consentono, talvolta, di realizzare il compromesso ottimo, permettendo velocità di esecuzione, facilità di debug, compattezza del codice oggetto generato.

Compilazione

Come già detto in precedenza, il linguaggio R3 è un linguaggio compilato. Il programma sorgente viene tradotto dal compilatore in codici esadecimali eseguibili dal microprocessore. Si è preferita la versione compilata per privilegiare al massimo la rapidità di esecuzione necessaria per correttamente eseguire le istruzioni contenute nelle "rules".

Nel file sorgente è possibile introdurre apposite istruzioni, dette metacomandi, che permettono di variare, secondo particolari esigenze, il funzionamento del compilatore stesso.

La compilazione genera, oltre al file oggetto, anche un file .LST contenente informazioni di tipo generale, nonchè l'elenco degli errori grammaticali sintattici eventualmente contenuti nel file sorgente. Se questi errori sono di tipo grave, il file oggetto non viene generato.

Ambiente di sviluppo RDE (Robox Development Environment)

Allo scopo di agevolare l'utente in tutte queste operazioni, è stato sviluppato un ambiente di sviluppo denominato RDE. Questo ambiente di sviluppo, che di norma viene ospitato su Personal Computer Windows, pur non essendo indispensabile, crea l'ambiente ideale per poter correttamente operare sia con il linguaggio R3 che con i moduli della Famiglia ROBOX.

Due tipi di programmazione

© 2025 Robox SpA

Il linguaggio R3 prevede due tipi di programmazione.

Il primo tipo di programmazione riguarda la descrizione delle leggi di movimento della macchina (programmazione delle "rules").

Il secondo tipo di programmazione, a livello "main", riguarda la descrizione delle sequenze operative della macchina (programmazione delle istruzioni).

Le "rules" (regole) dovranno essere eseguite ad una frequenza adeguata al processo controllato. Programmando le rules è lecito solo una programmazione lineare, non devono essere usati dei loop bloccanti o attese di eventi.

Evidentemente le risorse dedicate dall'apparecchiatura al controllo dei movimenti della macchina (rules) dovranno, in generale, essere molto maggiori di quelle dedicate al controllo delle sequenze operative.

Programmazione a livello RULE

A questo livello di programmazione è affidato il governo vero e proprio dei movimenti della macchina, in altre parole:

- la chiusura di anelli di posizione o di velocità
- il controllo di eventuali ballerini
- la generazione di opportune traiettorie
- la sincronizzazione di più assi
- ecc.

Queste funzioni dovranno essere descritte nelle "rules" e dovranno essere ottemperate dal controllo "execution time" con frequenza sufficientemente alta rispetto alla larghezza di banda della dinamica del processo controllato.

I moduli della famiglia Robox SpA sono concepiti per lavorare con una frequenza di aggiornamento delle rules che va da 25 Hz a 2 KHz (default 100 Hz), che appare più che sufficiente per controllare processi con larghezza di banda fino a 200 Hz.

Come accennato, alla programmazione a livello rule è affidato il compito di descrivere le leggi di movimento della macchina.

In linea di massima una rule sarà relativa ad uno degli "n" assi controllati, ma è ammesso che una rule descriva le leggi di moto di più assi. Allo scopo di poter descrivere le leggi di movimento nei vari stati operativi, sarà sufficiente scrivere le diverse rules.

In generale in una rule sono individuabili tre campi:

- REF
- MOTION
- AUX

nel campo REF (opzionale) saranno scritte le equazioni che governano la generazione del riferimento agli azionamenti (tipicamente riferimento di velocità). Se questo campo viene omesso il controllo userà le formule standard.

Nel campo MOTION saranno descritte le leggi di movimento.

Nel campo AUX saranno descritte funzioni al contorno per le quali valgono le stesse esigenze temporali caratteristiche delle rule.

Ripetiamo che le equazioni descritte nelle rule vengono eseguite con una frequenza fissa normalmente pari a 100 Hz, cui corrisponde un periodo di "scansione" di 10 ms, con una frequenza massima di 2 KHz, cui corrisponde un periodo di 0,5 ms. Per ogni battuta, possono essere eseguite fino a 32 rules.

Programmazione a livello MAIN

© 2025 Robox SpA

A questo livello di programmazione è affidato il compito di descrivere l'operatività della macchina.

La programmazione a livello main è una programmazione di tipo classico, orientata all'uso dei diagrammi di flusso. L'utente dovrà definire il o i diagrammi di flusso in base alle sue esigenze e successivamente convertirli in istruzioni del linguaggio R3, operazione che normalmente non presenta difficoltà né concettuali, né pratiche.

Per facilitare il compito di chi scrive, il linguaggio R3 permette una programmazione a livello main "multitasking".

Possono essere descritti fino a 8 programmi evolventi in parallelo. Tutti i task godono del medesimo livello di priorità, nel senso che il tempo macchina disponibile verrà ripartito fra tutti i task attivi.

Al fine di facilitare la sincronizzazione dei differenti task, sono disponibili un numero adeguato di variabili "globali", ossia accessibili dai diversi tasks. In ogni task si potranno poi aggiungere variabili locali secondo le necessità.

All'accensione dell'unità di governo viene mandato automaticamente in esecuzione soltanto il task numero 1 (task autorun). Sarà poi il task n. 1 ad avviare gli altri task, se necessari e se previsto dal programma del task 1.

Durante lo svolgimento del programma è inoltre possibile disabilitare uno o più task, ponendoli in stato di attesa. Il sistema è inoltre dotato della capacità di gestire gli "interrupt", ossia eventi esterni a fronte dei quali il normale programma deve essere abbandonato per permettere di gestire con la dovuta rapidità detti eventi.

Il controllo di tale "evento" viene effettuato all'interno di una RULE, e quindi con un tempo di latenza massimo nominalmente di 20 microsec. La rule a sua volta forzerà l'esecuzione di una apposita procedura scritta dall'utente a livello main.

#Sistema operativo in linea RTE (Real time extended)

Il linguaggio R3 è supportato nei moduli Robox da un sistema operativo real time RTE il quale garantisce fra le altre le seguenti funzioni fondamentali:

- assegna la giusta partizione di tempo all'evoluzione dei programmi a livello rule
- assegna la giusta partizione di tempo all'evoluzione di programmi a livello main
- assicura i servizi fondamentali a livello governo assi
- assicura i servizi fondamentali relativi alle comunicazioni (RS232 RS422/485 ETHERNET)
- assicura la gestione (drivers) di una varietà di schede I/O come:

  - schede input output digitali

  - schede input output analogiche

  - ecc.
- assicura i servizi fondamentali a livello diagnostico
- permette l'esecuzione delle rules asincrone.

Conclusioni

Per programmare in R3, il progettista dovrà innanzitutto assimilarne le istruzioni. Le istruzioni del linguaggio R3 sono le classiche istruzioni dei linguaggi di programmazione strutturati, con l'aggiunta di alcune istruzioni specifiche, strettamente legate alle problematiche cui il linguaggio R3 è destinato.

Chiunque abbia un certo grado di familiarità con linguaggi di programmazione quali il Basic, il C o altri, impiegherà poche ore per assimilare il linguaggio R3.

Chiunque non abbia dimestichezza coi predetti linguaggi, ma abbia già una certa esperienza di programmazione di PLC impiegherà qualche giorno.

© 2025 Robox SpA

Infine chi non ha nessun tipo di esperienza potrà imparare a programmare in qualche settimana, a patto di essere dotato di senso logico e di essere predisposto per le scienze matematiche. Programmare in linguaggio R3 significa poi conoscere i computer e averne confidenza.

L'operatore dotato delle conoscenze sopra descritte si siederà a tavolino e inizierà a descrivere il programma applicativo per la macchina di interesse.

In questa scrittura avrà a disposizione i cosiddetti text editor, cioè programmi che permettono di scrivere testi, memorizzarli e correggerli. L'insieme di istruzioni scritte dal programmatore si chiamerà listato sorgente e sarà in generale facilmente comprensibile da chiunque. A questo punto il programma dovrà essere "compilato" cioè, come già accennato, tradotto in codici numerici comprensibili all'apparecchiatura elettronica.

Questo file si chiamerà file oggetto. L'apparecchiatura elettronica in cui dovrà essere travasato dovrà essere collegata al personal computer mediante la linea seriale RS232 o ETHERNET di cui ogni personal computer è dotato. Il programma oggetto così generato verrà immagazzinato nelle memorie dell'apparecchiatura elettronica di controllo.

La macchina è pronta per funzionare e il programmatore potrà verificare che il comportamento sia quello da lui previsto. In generale ciò non riesce al primo colpo: l'operatore dovrà quindi verificare il suo programma, trovare gli errori concettuali o formali e correggerli.

Il linguaggio R3 e l'ambiente di sviluppo RDE rendono particolarmente agevole questa operazione. Una volta chiarita la natura degli inconvenienti, lo sviluppatore provvederà a correggere i programmi sorgenti, a ricompilare il programma e a reinserirlo nelle memorie dell'unità.

Requisiti software

Sistema operativo Microsoft Windows 7 (x86), Windows 8 e 8.1 (x86), Windows 10 (x86/x64), Windows 11 (x86/x64).

Metacomandi

Metacomandi

I metacomandi sono degli identificatori destinati a condizionare la fase di compilazione. Non generano direttamente nessun codice ma specificano la modalità di elaborazione del programma sorgente R3.

I metacomandi si suddividono in quattro gruppi:

- metacomandi di definizione di modalità di programmazione diagramma tempo generale 798
- metacomandi di condizionamento del listato 799
- metacomandi per scopi generali 799
- pseudo istruzioni 800

Metacomandi di definizione di modalità di programmazione diagramma tempo generale

$rule	Definizione task 0 che contiene le rule 748
$task n	Definizione Task (1-:-8)
$rule_periodic	Definizione regola periodica rule_periodic 758

© 2025 Robox SpA

$rule_time	Definizione regole asincrone int_timer 728 , int_after 725
$rule_inp	Definizione regola asincrona int_inp 726

Metacomandi di condizionamento del listato

$define 800	Definisce un simbolo
$undef 802	Rimuove la definizione di un simbolo
$ifdef-$else-$endif 801	Se il simbolo è definito, allora il programma viene compilato
$ifndef-$else-$endif 801	Se il simbolo non è definito, allora il programma viene compilato
$expand	Se presente, in fase di compilazione viene gestita l'espansione dei LIT all'interno del file listato

Metacomandi per scopi generali

$include 802	Include un file esterno nella compilazione (*.i3). Utile per la definizione simbolica delle risorse hardware
$close_loop	Simula la chiusura dell'anello di posizione. Se presente, RTE simula i valori di cp e cv degli assi
$check_array	verifica, run-time, se vi è un accesso ad un elemento di un array inesistente. Ovviamente penalizza il tempo di esecuzione.
$write_on_extern	Abilita possibilità di scrivere le variabili definite come **extern** (public 814)
$use_rpe	Abilita l'estensione al linguaggio RPE (Robox Path Executor). Aggiunto in automatico usando il configuratore di progetto RDE (gruppi di assi)
$error "messaggio"	Da RC3E v33.1.21 Forza il compilatore a dare un errore di compilazione (utile ad es. all'interno di un blocco condizionato, per accorgersi se il blocco viene o meno compilato)
$step	Per ottenere il massimo delle prestazioni dalle rule interruptanti è stato eliminato il superfluo. Volendo riabilitare la 'marcatura' dei passi di un task di rule

© 2025 Robox SpA

	asincrone (RULE_PERIODIC incluso) al fine di poter mettere dei breakpont sull'esecuzione dei passi, bisogna aggiungere in testa al file (dopo la definizione del tipo di file) la linea **$step** Esempio: **$rule_periodic** **$step**
$warning "messaggio"	Da RC3E v33.1.21 Forza il compilatore a dare un warning di compilazione (utile ad es. all'interno di un blocco condizionato, per accorgersi se il blocco viene o meno compilato)

Pseudo istruzioni

Le variabili di tipo BIT_MASK permettono di definire e inizializzare particolari variabili intere (di tipo U32) da utilizzarsi come maschere di bit.

Tipicamente l'impostazione dei bit avviene in automatico tramite le scelte di configurazione che l'utente opera nel configuratore di progetto dci RDE.

Le variabili di tipo BIT_MASK possono comunque essere lette e modificate anche da sorgente R3

bit_mask_set nomevar valore	La variabile nomevar viene inizializzata a valore
bit_mask_add nomevar valore	In nomevar vengono posti a 1 i bit specificati da valore, senza alterare gli altri
bit_mask_remove nomevar valore	In nomevar vengono posti a 0 i bit specificati da valore, senza alterare gli altri

Annotazioni:

- anche l'istruzione lit 729 che permette di definire dei simbolici alle grandezze.

Metacomando $define

Permette di definire un simbolo che poi verrà utilizzato per condizionare la compilazione ($ifdef 801, $ifndef 801).

Sintassi	**$define symbol** **$define symbol = value**
symbol	Il nome del simbolo da definire
value	Il valore da associare al simbolo

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
; definisco il simbolo versione e gli assegno il valore 12
$define versione = 12
; definisco il simbolo versione_di_prova
$define versione_di_prova
; se versione è maggiore o uguale a 12 e versione_di_prova è falso...
$ifdef 801 versione >= 12 and not versione_di_prova
    ; allora questa parte di codice sarà compilata
    ...
$endif 801
```

Metacomando $ifdef/$ifndef

Esegue la condizione if in base alla definizione o meno di un simbolo.

Sintassi	**$ifdef <bool> [<op> <bool>[...]]** istruzioni... **$else** istruzioni... **$endif**
op	Operatore logico 809: and, and not, or, or not, xor
bool	Sono espressioni booleane (max. 5) composte come segue: simbolo <opr> cost dove <opr> è un operatore a scelta fra: => >= < <= <> e cost è una costante numerica
Note	Regole di precedenza: per primi vengono valutati <bool>, successivamente viene valutata la sequenza dei <bool> da sinistra a destra, senza applicazione di priorità fra i vari <op>

Esempi

Esempio di utilizzo:

```
; definisco il simbolo versione e gli assegno il valore 12
$define 800 versione = 12
; definisco il simbolo versione_di_prova
$define 800 versione_di_prova
; se versione è maggiore o uguale a 12 e versione_di_prova è falso...
$ifdef versione >= 12 and not versione_di_prova
    ; allora questa parte di codice sarà compilata
    ...
$else
    ; altrimenti questa parte di codice sarà compilata
    ...
```

© 2025 Robox SpA

```
$endif
```

Metacomando $include

Permette di inglobare nella compilazione il file esterno indicato.

Sintassi	**$include <filename>**
filename	Nome del file da includere. Il file è riferito alla cartella del file che contiene la direttiva $include, a meno che sia specificato con altro percorso
Note	Possono essere inclusi più file (max:400)

Esempi

Esempio di utilizzo:

```
; Nel main (programma principale)
$task 1
; definizione dei registri
$include litreg.i3
; definizione degli input/output
$include litio.i3
; definizione degli input/output con percorso relativo
$include ./filesIO/litio.i3
```

Esempio di contenuto del file litio.i3:

```
...
lit iAriaCompressa     inp(145)
lit iManuale           inp(146)
...
```

Esempio di contenuto del file litreg.i3:

```
...
lit rrVelMan          nvrr(1)
lit rrAccMan          nvrr(2)
...
```

Metacomando $undef

Permette di rimuovere la definizione di un simbolo precedentemente definito utilizzando $define 800.

Sintassi	**$undef symbol**
symbol	Nome del simbolo da rimuovere

Esempi

Esempio di utilizzo:

© 2025 Robox SpA

```
$define 800 PROX_ABRASIVI
$define 800 PONTE_2

...

$undef PROX_ABRASIVI
$undef PONTE_2
```

Operatori

Operatori

Gli operatori sono quei caratteri, o parole chiave, che permettono di mettere in relazione variabili, costanti e funzioni per generare espressioni più o meno complesse.

Gli operatori possono essere divisi in sei gruppi:

- operatori di assegnazione 808 (=)
- operatori aritmetici 804 (+ , - , * , /)
- operatori logici 809 (**and** , **or** , **not** , **xor**)
- operatori relazionali numerici 811 (> , < , = , <> , >= , <=)
- operatori binari su variabili intere 806 (**r_and** , **r_not** , **r_or** , **r_xor**)
- operatori di concatenamento stringhe 808 (#)

Le precedenze naturali di esecuzione degli operatori sono riportate nella tabella sottostante. In generale le precedenze sono nell'ordine:

- Unari
- Aritmetici
- Relazionali
- Binari su bit
- Binari logici
- Assegnazione

Per imporre l'ordine di esecuzione desiderato si devono usare coppie di parentesi. In generale si consiglia di aggiungere coppie di parentesi ogniqualvolta le espressioni possano fare nascere dubbi sulle precedenze.

La precedenza è secondo l'ordine di scrittura. I gruppi identificano lo stesso livello di priorità.

Meno unario	-	- var
Più unario	+	+ var
Not logico	**not**	**not** var
Not binario	**r_not**	**r_not** (var)
Moltiplicazione	*	var * var

© 2025 Robox SpA

Divisione	/	var / var
Addizione	+	var + var
Sottrazione	-	var - var
Maggiore	>	var > var
Maggiore uguale	> =	var > = var
Minore	<	var < var
Minore uguale	< =	var < = var
Uguale	=	var = var
Diverso	< >	var < > var
And binario	**r_and**	var **r_and** var
Or binario	**r_or**	var **r_or** var
Xor binario	**r_xor**	var **r_xor** var
And logico	**and**	var **and** var
Or logico	**or**	var **or** var
Assegnazione	=	var = var

Operatori aritmetici

Validità: **Rule, Task**

In generale questi operatori lavorano fra grandezze intere o reali. Se una grandezza reale viene assegnata a una variabile di tipo intero, le cifre decimali vengono troncate. Qualora il valore sia troppo grande per essere rappresentato nella variabile intera, il risultato sarà imprevedibile. Quando un operatore aritmetico unisce un reale con un intero, il risultato è convertito in reale.

Addizione

Sintassi	**valueA + valueB**
valueA	Primo valore addendo
valueB	Secondo valore addendo
Risultato	La somma dei due valori

Operatore di somma tra due grandezze numeriche.

Esempio di utilizzo:

```
; Il registro rr(12) conterrà la somma della posizione corrente dell'ass
; e del contenuto del registro rr(3)
rr(12) = cp(2) + rr(3)
; Può essere usato anche come operatore di segno (unario)
```

© 2025 Robox SpA

```
; Il registro r(12) conterrà il valore +3
r(12) = +3
```

Annotazioni:

- per sommare (concatenare) delle stringhe, usare l'operatore # 808

Sottrazione

Sintassi	**valueA - valueB**
valueA	Valore del minuendo
valueB	Valore del sottraendo
Risultato	La differenza tra minuendo e sottraendo

Operatore di sottrazione fra grandezze numeriche.

Esempio di utilizzo:

```
; Il registro rr(12) conterrà la differenza tra il contenuto del registr
; e della variabile "remainder"
rr(12) = r(2) - remainder
; Può essere usato anche come operatore di segno (unario)
; Il registro r(12) conterrà il valore -3
r(12) = -3
```

Moltiplicazione

Sintassi	**valueA * valueB**
valueA	Valore del moltiplicando
valueB	Valore del moltiplicatore
Risultato	La moltiplicazione tra moltiplicando e moltiplicatore

Operatore di moltiplicazione fra grandezze numeriche.

Esempio di utilizzo:

```
; Il registro rr(12) conterrà il prodotto tra la costante K_PI (Pigreco)
; e la variabile "diameter"
rr(12) = K_PI 17 * diameter
```

Divisione

Sintassi	**valueA / valueB**
valueA	Valore del dividendo
valueB	Valore del divisore
Risultato	La divisione tra dividendo e divisore

Operatore di divisione fra grandezze numeriche.

© 2025 Robox SpA

Annotazioni:

- Nel caso di divisione fra interi, il risultato è costituito dalla sola parte intera della divisione anche se la destinazione è una grandezza di tipo reale.
- Il valore del divisore deve essere diverso da 0.

Esempio di utilizzo:

```
; Il registro rr(12) conterrà il rapporto il valore 2.0 e la
; variabile "value" a sua volta diviso per la variabile "remainder"
REAL value = 10.0
REAL remainder = 0.5
rr(12) = (2.0 / value) / remainder
```

Operatori a bit (binari)

Validità: **Rule**, **Task**

L'uso di questa famiglia di operatori è riservato ai valori interi. In caso di utilizzo con valori reali, il compilatore evidenzierà un errore. Il risultato è ancora un valore di tipo intero.

AND binario

Sintassi	**valueA r_and valueB**
valueA	Primo valore
valueB	Secondo valore
Risultato	AND binario tra i due valori

Operazione di AND a bit (binario) tre due valori interi.

Esempio di utilizzo:

```
; Con i dati riportati in tabella, "value1" conterrà il valore 0xC0
value1 = inp_w(1) r_and 0c8h

; Esito binario dell'operazione
```

	B3								B2								B1								B0							
inp_w(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	1	1	0	1	0	1	0	0
0c8h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
value1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

```
; "value2" conterrà il valore del byte più basso di r(1)
value2 = r(1) r_and 0xFF
```

OR binario

Sintassi	**valueA r_or valueB**
valueA	Primo valore
valueB	Secondo valore

© 2025 Robox SpA

Sintassi	**valueA r_or valueB**
Risultato	OR binario tra i due valori

Operazione di OR bit a bit (binario) tra due valori interi.

Esempio di utilizzo:

```
; Con i dati riportati in tabella, "value1" conterrà il valore 0xF6DC
value1 = inp_w(1) r_or 0c8h

; Esito binario dell'operazione
```

	B3								B2								B1								B0							
inp_w(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	1	1	0	1	0	1	0	0
0c8h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
value1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	1	1	0	1	1	1	0	0

NOT binario

Sintassi	**r_not value**
value	Valore
Risultato	NOT binario del valore

Operazione di NOT bit (binario) per un valore intero.

Esempio di utilizzo:

```
; Con i dati riportati in tabella, "value1" conterrà il valore 0xFFFF092
value1 = r_not inp_w(1)

; Esito binario dell'operazione
```

	B3								B2								B1								B0							
inp_w(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	1	1	0	1	0	1	0	0
value1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	0	0	1	0	0	1	0	1	0	1	0

XOR binario

Sintassi	**valueA r_xor valueB**
valueA	Primo valore
valueB	Secondo valore
Risultato	XOR binario tra i due valori

Operazione di XOR a bit (binario) tra due valori interi.

Esempio di utilizzo:

```
; Con i dati riportati in tabella, "pippo" conterrà il valore 0xF61C
pippo = inp_w(1) R_XOR 0C8H

; Esito binario dell'operazione
```

© 2025 Robox SpA

	B3								B2								B1								B0							
inp_w(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	1	1	0	1	0	1	0	0
0c8h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
value1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	0	1	1	1	0	0

Operatori di assegnazione

Validità: **Rule, Task**

Sintassi	**var = valueS** **var = varS**
var	Variabile destinazione
valueS	Valore costante o espressione sorgente
varS	Variabile sorgente

Permette di assegnare ad una variabile il valore di una costante, o di una variabile o di una espressione.

NOTA: se si assegna ad una variabile intera un valore reale si avrà il troncamento della parte decimale; qualora il valore sia troppo grande per essere rappresentato nella variabile intera, il risultato sarà imprevedibile.

Esempio di utilizzo:

```
; Il registro r(100) conterrà il valore del prodotto 17*3
; meno il valore della variabile "value1"
r(100) = 17 * 3 - value1
; La variabile "value2" assumerà il valore del contenuto del registro r(
; moltiplicato per la costante K_PI (Pigreco)
value2 = r(100) * K_PI
; Il registro stringa sr(12) conterrà la stringa:
; Inserisci un desiderio..
sr(12) = "Inserisci un desiderio.."
```

Operatori di concatenamento stringhe

Validità: **Rule, Task**

Sintassi	**valueA # valueB**
valueA	Primo valore stringa
valueB	Secondo valore stringa
Risultato	La concatenazione tra le due stringhe

Operatore di concatenamento stringhe, può essere utilizzato solo con variabili di tipo stringa.

Esempio di utilizzo:

```
; In entrambi i casi il registro stringa sr(1) conterrà la stringa: "son
```

© 2025 Robox SpA

```
sr(1) = "sono" # " le otto"
sr(1) = "sono" # " " # "le otto"
```

Operatori logici

Validità: **Rule, Task**

Questi operatori operano fra grandezze booleane binarie e rendono un risultato di tipo booleano binario, cioè un risultato che può avere due soli valori: "vero" o "falso".

AND logico

Sintassi	**valueA and valueB**
valueA	Primo valore booleano
valueB	Secondo valore booleano

Risultato

valueA	valueB	Risultato
falso	falso	falso
vero	falso	falso
falso	vero	falso
vero	vero	vero

Congiunzione logica fra espressioni booleane.

Esempio d utilizzo:

```
; se il contenuto del registro rr(12) è uguale a zero e l'input(5) è att
; allora al registro stringa sr(1) sarà assegnata la stringa:
; "la variabile rr(12) è uguale a zero e l'input 5 è TRUE"
_if ((rr(12) = 0) and inp(5))
    sr(1) = "la variabile rr(12) è uguale a zero e l'input 5 è TRUE"
; se il contenuto dei registri r(12) e r(5) sono diversi da zero
; allora al registro stringa sr(1) sarà assegnata la stringa:
; "sia il registro r(12) che r(5) sono diversi da zero"
_if (r(12) and r(5))
    sr(1) = "sia il registro r(12) che r(5) sono diversi da zero"
```

NOT logico

Sintassi	**not value**
value	Valore booleano

Risultato

value	Risultato
falso	vero
vero	falso

© 2025 Robox SpA

Operatore di negazione logica di una espressione booleana.

Esempio di utilizzo:

```
; se è falso che il valore della variabile "value1"
; è uguale al contenuto del registro r(5)
; allora al registro stringa sr(3) sarà assegnata la stringa:
; "la variabile value1 è diversa dal contenuto di r(5)"
if (not (value = r(5)))
    sr(3)= "la variabile pippo è diversa dal contenuto di r(5)"
endif
```

OR logico

Sintassi	**valueA or valueB**
valueA	Primo valore booleano
valueB	Secondo valore booleano
Risultato	

valueA	valueB	Risultato
falso	falso	falso
vero	falso	vero
falso	vero	vero
vero	vero	vero

Disgiunzione logica fra espressioni booleane.

Esempio di utilizzo:

```
; se il valore della variabile "value1" è diverso da zero
; oppure l'input(5) è attivo (cioé diverso da zero)
; allora al registro stringa sr(3) sarà assegnata la stringa:
; "la variabile value1 è <> 0 o l'input 5 è TRUE"
if (value1 or inp(5))
    sr(3) = "la variabile pluto è <> 0 o l'input 5 è TRUE"
endif
```

XOR logico

Sintassi	**valueA xor valueB**
valueA	Primo valore booleano
valueB	Secondo valore booleano
Risultato	

valueA	valueB	Risultato
falso	falso	falso
vero	falso	vero

© 2025 Robox SpA

Sintassi	**valueA xor valueB**

valueA	valueB	Risultato
falso	vero	vero
vero	vero	falso

Disgiunzione logica esclusiva fra espressioni booleane.

Esempio di utilizzo:

```
; se l'input(1) e l'input(2) sono entrambi veri o entrambi falsi
; allora alla variabile booleana reso sarà assegnato il valore falso
; se uno dei due input è vero e l'altro è falso
; allora alla variabile booleana reso sarà assegnato il valore vero
reso = xor(inp(1), inp(2))

; se solo uno dei due input è vero e l'altro è falso
; allora il contenuto del registro r(1) viene incrementato di una unità
if (xor(inp(1), inp(2)))
    inc(r(1) ,1)
endif
```

Operatori relazionali numerici

Validità: **Rule**, **Task**

Questi operatori operano su grandezze di tipo numerico. L'uso su grandezze di tipo logico deve essere evitato mentre per confrontare delle stringhe è necessario utilizzare le apposite funzioni.

Operatore minore

Sintassi	**valueA < valueB**
valueA	Primo valore
valueB	Secondo valore
Risultato	Vero se valueA minore di valueB, falso altrimenti.

Esempio di utilizzo:

```
; se il contenuto del registro r(2) è minore del contenuto del registro
; allora saranno eseguite le istruzioni seguenti alla "if"
if (r(2) < rr(3))
    ...
endif

; Se la condizione è falsa (cioè il contenuto del registro nvr(1)
; non è minore del contenuto del registro nvr(2))
; allora a "reso" sarà assegnato il valore 0 (reso = 0)
; Se la condizione è vera (cioè il contenuto del registro nvr(1)
```

© 2025 Robox SpA

```
; è minore del contenuto del registro nvr(2))
; allora a "reso" sarà assegnato il valore 1 (reso = 1)
reso = (nvr(1) < nvr(2))
```

Operatore minore o uguale

Sintassi	**valueA <= valueB**
valueA	Primo valore
valueB	Secondo valore
Risultato	Vero se valueA minore o uguale a valueB, falso altrimenti.

Esempio di utilizzo:

```
; se la variabile "value1" è minore o uguale a 3.1415
; allora saranno eseguite le istruzioni seguenti alla "if"
if (value1 <= 3.1415)
    ...
endif

; Se la condizione è falsa (cioè il contenuto del registro nvr(1)
; non è minore o uguale al contenuto del registro nvr(2))
; allora a "reso" sarà assegnato il valore 0 (reso = 0)
; Se la condizione è vera (cioè il contenuto del registro nvr(1)
; è minore o uguale al contenuto del registro nvr(2))
; allora a "reso" sarà assegnato il valore 1 (reso = 1)
reso = (nvr(1) <= nvr(2))
```

Operatore maggiore

Sintassi	**valueA > valueB**
valueA	Primo valore
valueB	Secondo valore
Risultato	Vero se valueA maggiore di valueB, falso altrimenti.

Esempio di utilizzo:

```
; se il contenuto del registro rr(2) è maggiore del contenuto del regist
; allora saranno eseguite le istruzioni seguenti alla "if"
if (rr(2) > r(3))
    ...
endif

; Se la condizione è falsa (cioè il contenuto del registro nvr(1)
; non è maggiore del contenuto del registro nvr(2))
; allora a "reso" sarà assegnato il valore 0 (reso = 0)
; Se la condizione è vera (cioè il contenuto del registro nvr(1)
; è maggiore del contenuto del registro nvr(2))
```

© 2025 Robox SpA

```
; allora a "reso" sarà assegnato il valore 1 (reso = 1)
reso = (nvr(1) > nvr(2))
```

Operatore maggiore o uguale

Sintassi	**valueA >= valueB**
valueA	Primo valore
valueB	Secondo valore
Risultato	Vero se valueA maggiore o uguale a valueB, falso altrimenti.

Esempio di utilizzo:

```
; se la variabile "value1" è maggiore o uguale a 3.1415
; allora saranno eseguite le istruzioni seguenti alla "if"
if (value1 >= 3.1415)
    ...
endif

; Se la condizione è falsa (cioè il contenuto del registro nvr(1)
; non è maggiore o uguale al contenuto del registro nvr(2))
; allora a "reso" sarà assegnato il valore 0 (reso = 0)
; Se la condizione è vera (cioè il contenuto del registro nvr(1)
; è maggiore o uguale al contenuto del registro nvr(2))
; allora a "reso" sarà assegnato il valore 1 (reso = 1)
reso = (nvr(1) >= nvr(2))
```

Operatore di uguaglianza

Sintassi	**valueA = valueB**
valueA	Primo valore
valueB	Secondo valore
Risultato	Vero se valueA uguale a valueB, falso altrimenti.

Esempio di utilizzo:

```
; Se il contenuto del registro r(2) è uguale alla variabile "value1"
; allora saranno eseguite le istruzioni seguenti alla "if"
if (r(2) = value1)
    ...
endif

; Se il contenuto del registro rr(2) è uguale alla variabile "value2" us
; la funzione similar(), allora saranno eseguite le istruzioni seguenti
if (similar (rr(2), value1)
    ...
endif
```

© 2025 Robox SpA

```
; Se la condizione è falsa (cioè il contenuto del registro nvr(1)
; non è uguale al contenuto del registro nvr(2))
; allora a "reso" sarà assegnato il valore 0 (reso = 0)
; Se la condizione è vera (cioè il contenuto del registro nvr(1)
; è uguale al contenuto del registro nvr(2))
; allora a "reso" sarà assegnato il valore 1 (reso = 1)
reso = (nvr(1) = nvr(2))
```

Annotazioni:

- Si sconsiglia l'uso dell'operatore = per confrontare grandezze reali (o un reale con un intero): si consiglia invece, di usare l'apposita funzione similar601().

Operatore di disuguaglianza

Sintassi	**valueA <> valueB**
valueA	Primo valore
valueB	Secondo valore
Risultato	Vero se valueA diverso da valueB, falso altrimenti.

Esempio di utilizzo:

```
; se la variabile "value" è diversa da 3.1415
; allora saranno eseguite le istruzioni seguenti alla "if"
if (value1 <> 3.1415 )
    ...
endif

; Se la condizione è falsa (cioè il contenuto del registro nvr(1)
; non è diverso dal contenuto del registro nvr(2))
; allora a "reso" sarà assegnato il valore 0 (reso = 0)
; Se la condizione è vera (cioè il contenuto del registro nvr(1)
; è diverso dal contenuto del registro nvr(2))
; allora a "reso" sarà assegnato il valore 1 (reso = 1)
reso = (nvr(1) <> nvr(2))
```

Tipi di dato

Le variabili sono entità il cui valore non è noto durante la scrittura di un programma e normalmente varia durante l'esecuzione dello stesso.

I tipi di dato utilizzati dal linguaggio R3 sono:

Tipo	Descrizione
I8925	Variabile intera a 8 bit con segno
I16923	Variabile intera a 16 bit con segno
I32924	Variabile intera a 32 bit con segno

© 2025 Robox SpA

Tipo	Descrizione
U8 932	Variabile intera a 8 bit senza segno
U16 930	Variabile intera a 16 bit senza segno
U32 931	Variabile intera a 32 bit senza segno
INT 924	Variabile intera a 32 bit (è come I32)
FLOAT 922	Variabile floating point 4 Byte
REAL 928	Variabile floating point 8 Byte (double)
BOOL 919	Variabile booleana
CHAR 921	Variabile di tipo char (è come I8)
STRUCT 917	Variabile di tipo struttura
STRUCT_P 918	Variabile di tipo struttura packed (i dati sono contigui, occupa meno spazio)
STRING 929	Variabile di tipo stringa (127 char + terminatore)
TABSTR 919	Tabelle di costanti stringa
COUNTER 921	Variabile di tipo counter
TIMER 930	Variabile di tipo timer
POWER_SET 926	Variabile per la gestione della potenza
BCC_MSGUSER 820	Struttura di appoggio per funzioni bcc_send() 229, bcc_receive() 226, ..
STRU_BCC_TCP_CLIENT 830	Struttura di appoggio per funzioni bcc_get_tcp_clients_info() 222 e getTcpClientsInfo()
STRU_AAX 822	Struttura di appoggio per funzioni planar_comp_ip 519() e planar_comp_cp 518()
STRU_AXIO_PDI 828	Struttura di appoggio per funzioni axio_robj() 186 e axio_wobj() 199
STRU_CO_OBJ 832	Struttura di appoggio per funzioni co_robj() 258 e co_wobj() 273
STRU_COE_OBJ 834	Struttura di appoggio per funzioni co_robj() 258 e co_wobj() 273
STRU_COERRMSG 836	Struttura di appoggio per funzione co_rerr() 255
STRU_STAT 912	Struttura di appoggio per funzione statistics() 621
STRU_GROR 869	Struttura di appoggio per funzione group 721 order() 735

© 2025 Robox SpA

Tipo	Descrizione
STRU_FILEINFO 843	Struttura di appoggio per funzione di gestione f_dir() 321
STRU_MVITP 879	Struttura di appoggio per funzione di movimento mva_to_n() 496
STRU_MATCAM 875	Struttura di appoggio per funzione di movimento mv_cam() 451
STRU_MODBUS 876	Struttura di appoggio per funzioni di comunicazione modbus/tcp 138
STRU_CAM 831	Struttura di appoggio per funzione di movimento mv_table() 481
STRU_MVTO 881	Struttura di appoggio per funzione di movimento mv_to() 482
STRU_MVTOCJ 884	Struttura di appoggio per funzione di movimento mv_to_cj() 486
STRU_MVTOCJV 886	Struttura di appoggio per funzione di movimento mv_to_cjv() 487
STRU_MVTOCJV_INFO 888	Struttura di appoggio per funzione di ausilio alla mv_to_cjv_info() 489
STRU_MVTOCJVE 890	Struttura di appoggio per funzione di movimento mv_to_cjv() 487
STRU_MVZC 892	Struttura di appoggio per funzione di movimento mva_zc() 511
STRU_MVGENITP 877	Struttura di appoggio per funzione di movimento mva_to_n_v() 504
STRU_MVITPCJ 880	Struttura di appoggio per funzione di movimento mva_to_n_cj() 500
STRU_AH_D 823	Struttura di appoggio per funzione ah_get_d() 159
STRU_AH_D_EXT 824	Struttura di appoggio per funzione ah_get_d() 159 con parametri USV
STRU_AH_INFO 825	Struttura di appoggio per funzione ah_info() 162
STRU_AS_D 826	Struttura di appoggio per funzione as_get_d() 178
STRU_AS_D_EXT 827	Struttura di appoggio per funzione as_get_d() 178 con parametri USV
STRU_FILEHND 842	Struttura di appoggio per funzioni di gestione flash-card f_open() 327, f_close() 318, ..

© 2025 Robox SpA

Tipo	Descrizione
STRU_INT 870	Struttura di appoggio per istruzioni di gestione interrupt int_after() 725, int_inp() 726 e int_timer() 728
STRU_LARGE_INT 872	Struttura di appoggio per istruzioni get_i64() 719 e set_i64() 770
STRU_LARGE_UINT 873	Struttura di appoggio per istruzioni get_u64() 720 e set_u64() 774
STRU_CRIMPER 839	Struttura di appoggio per funzione di movimento mv_crimper() 453
STRU_ECAT_PROBE 841	Struttura di appoggio per funzione ecat_probe() 313
STRU_FOLLOW 844	Struttura di appoggio per funzione di movimento mv_follow() 454
STRU_FOLLOW2 458	Struttura di appoggio per funzione di movimento mv_follow2() 458
STRU_FOLLOW2_S 458	Struttura di appoggio per funzione di movimento mva_follow2() 458
STRU_FOLLOW2_SM 458	Struttura di appoggio per funzione di movimento mva_follow2_m() 458
STRU_FOLLOW2_INFO 854	Struttura di appoggio per funzione di movimento mv_follow2() 458
STRU_HW_INFO 870	Struttura di appoggio per istruzione get_hw_info() 351
STRU_PHADJ 898	Struttura di appoggio per funzione di movimento mv_phase_adj() 472
STRU_PHADJ2 901	Struttura di appoggio per funzione di movimento mv_phase_adj2() 474
STRU_PM_PD 904	Struttura di appoggio per funzione axio_pm_r() 183
STRU_REACHTARGET 906	Struttura di appoggio per funzione di movimento mv_reach_target() 477
STRU_SAFETY_PRJINFO 908	Struttura di appoggio per funzione safe_prj_info() 574
STRU_SOE_IDN 909	Struttura di appoggio per funzioni soe_ridn() 603 e soe_widn() 615
STRU_SYNCHRO 913	Struttura di appoggio per funzione di movimento mv_synchro() 479
STRU_TRACK 916	Struttura di appoggio per funzione tracking() 648

© 2025 Robox SpA

Tipo	Descrizione
STRU_MASTERF 874	Struttura di appoggio per funzione master_filter() 389

NOTA: è consentita la nidificazione delle strutture (strutture di strutture).

Le variabili possono avere i seguenti livelli di visibilità:

Visibilità	Descrizione
Globale	Sono le variabili predefinite dal linguaggio R3 o le locali dichiarate come pubbliche (vedi sotto). Sono accessibili da tutti i task e utilizzabili su tutti gli strumenti di debug
Locale	Variabili accessibili (R/W) dal solo task in cui sono dichiarate. Devono essere definite all'inizio del file sorgente prima delle istruzioni. `INT pluto`
Pubblica	Variabili accessibili (R/W) dal task a cui sono dichiarate e normalmente leggibili dagli altri task. (vedi esterne) Devono essere definite all'inizio del file sorgente prima delle istruzioni con il prefisso "PUBLIC". `PUBLIC INT pluto`
Esterna	Variabili provenienti da un task differente. Sono normalmente di sola lettura a meno che il task che le include non abbia il metacomando "$WRITE_ON_EXTERN 814" al proprio inizio. Devono essere definite all'inizio del file sorgente con il prefisso "EXTERN" a cui deve necessariamente corrispondere in un altro task l'equivalente PUBLIC. (vedi public) `EXTERN INT pluto`
Ritentiva	Struttura utente della quale si vuole la ritentività allo spegnimento. (memoria ritentiva) Ha le seguenti particolarità: • Dimensione massima della struttura è minore di quella dell'area utente di memoria ritentiva (vedere report) • Non può essere un array di struttura (ma nulla vieta di farlo al suo interno) • Dimensione massima del nome della struttura di 31 caratteri • Dimensione massima del nome della variabile di 31 caratteri • Per poter scrivere questa struttura deve essere presente il metacomando $ WRITE_ON_EXTERN 814 • L'associazione della struttura utente a quella ritentiva avviene solo in fase di caricamento se e solo se NON esistono associazioni precedenti

© 2025 Robox SpA

Visibilità	Descrizione

	• Se esiste già una struttura ritentiva allora, in fase di messa in esecuzione dei task, si controlla che il nome della struttura, il nome dell'istanza e la sua dimensione corrispondano con quelli precedentemente registrati
	• Questa struttura viene salvata, così come i registri, allo spegnimento del controllo
	Esempio
	`$WRITE_ON_EXTERN`
	`STRUCT ret_struct`
	`  INT integerData[20];`
	`  REAL realData[10];`
	`  STRING setName[64];`
	`END_STRUCT`
	`RETENTIVE ret_struct retentiveData`
	`retentiveData.integerData[0] = SECOND`
	`retentiveData.integerData[1] = MINUTE`
	`retentiveData.integerData[2] = HOUR`
	NOTA: per eliminare l'associazione della struttura utente a quella ritentiva utilizzare il comando dipositivo (direttiva ascii) REMOVE NV_USERSTRUCT.
	NOTA: per inizializzare la struttura in caso di perdita di memoria utilizzare il comando dispositivo (direttiva ascii) RESTORE NV_USERSTRUCT.
	NOTA: per allargare automaticamente la dimensione della struttura, a patto che il nome della struttura e dell'istanza siano uguali, utilizzare il bit 10 della variabile predefinita SYS_CFG.

Locale alle funzioni	Variabili che sono definite all'interno della function() 715 e che sono utilizzabili solo all'interno della stessa. Queste variabili perdono il loro contenuto ad ogni uscita dalla funzione. Nota: è possibile ridefinire il nome di variabili locali, vale a dire che le variabili interne possono avere lo stesso nome delle variabili locali. Tuttavia si consiglia di utilizzare nomi diversi per evitare di fare confusione. Attenzione: sono locazioni diverse, vale a dire che le variabili interne e quelle locali con lo stesso nome occupano locazioni di memoria fisica differente. La modifica di una variabile interna non ha alcun effetto su una variabile locale che ha lo stesso nome. È possibile dichiarare in funzioni diverse variabili con lo stesso nome. NON sono utilizzabili dagli strumenti di debug. Possono essere inserite anche nei campi motion 730 all'interno delle rule 748. Si comportano in modo analogo a quelle dichiarate nelle function.

È possibile dichiarare variabili locali nelle rules, purché all'interno dei blocchi REF / MOTION / AUX. Le variabili del blocco REF sono visibili esclusivamente nel blocco stesso, quelle di MOTION e AUX sono invece in comune tra loro.

È possibile dichiarare variabili locali entro il codice main dei task (la visibilità è limitata tra il punto della dichiarazione e la fine del main).

© 2025 Robox SpA

Esempi

Vari esempi di dichiarazione dati:

```
; "varname" è una stringa di caratteri che deve necessariamente iniziare
; La definizione di variabile ha la seguente struttura:
; definizione singola variabile di tipo 'tipoDato'
tipoDato   varname
; definizione di n variabili di tipo 'tipoDato'
tipoDato   varname1, varname2, ..., varnameN
; definizione di un array monodimensionale di n elementi di tipo 'tipoDa
; gli indici degli elementi vanno da 0 a n-1, quindi:
; il primo elemento è varname[0] e l'ultimo è varname[n-1]
tipoDato   varname [n]
; definizione di un array bidimensionale di n*m elementi di tipo 'tipoDa
; gli indici degli elementi vanno da 0 a n-1 e da 0 a m-1, quindi:
; il primo elemento è varname[0][0] e l'ultimo è varname[n-1][m-1]
tipoDato   varname [n][m]
; definizione di un array tridimensionale di n*m*o elementi di tipo 'tip
; gli indici degli elementi vanno da 0 a n-1, da 0 a m-1 e da 0 a o-1, q
; il primo elemento è varname[0][0][0] e l'ultimo è varname[n-1][m-1][o-
tipoDato   varname [n][m][o]
; definizione singola variabile inizializzata di tipo 'tipoDato'
; vale a dire che alla variabile viene assegnato il valore k
tipoDato   varname = k
; definizione di un array inizializzato di tipo 'tipoDato'
; ai singoli elementi sono assegnati i valori k1, k2,..., kn nella segue
; varname[0]=k1, varname[1]=k2, .. , varname[n-1]=kn,
tipoDato   varname[n] = {k1, k2,..,kn}
; definizione di una variabile visibile in lettura anche dagli altri tas
; che la dichiarano come extern (usare il metacomando $WRITE ON EXTERN 814
public     tipoDato     varname
; definizione di una variabile a sola lettura dichiarata in un altro tas
extern     tipoDato     varname
```

Struttura BCC_MSGUSER

Validità: **RC3E v33.3.1** (o successive)

Questa keyword permette di definire una variabile di tipo BCC_MSGUSER (STRUCT_P 918).

Questa variabile viene utilizzata dalle istruzioni di invio e ricezione messaggi bcc_send 229() e bcc_receive 226().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
resSys1	U32	Riservato
resSys2	U8	Riservato

© 2025 Robox SpA

Campo	Tipo	Descrizione
bccID	U8	Codice identificativo del protocollo BCC
dst	U8	Destinazione
dch	U8	ID canale di destinazione
src	U8	ID sorgente
sch	U8	ID canale sorgente
msg	U16	Codice messaggio
lendata	U8	Dimensione dati messaggio in Byte
pid	U8	ID protocollo
dlc	U8	Contatore dead loop
data[255]	STRING	buffer dati

Esempi

Esempio di dichiarazione:

```
; definizione della variabile miomessaggio di tipo BCC_MSGUSER
BCC_MSGUSER miomessaggio
```

Esempio di utilizzo:

```
; utilizzo del dato contenuto in un campo della struttura
; definizione della variabile miomessaggio di tipo BCC_MSGUSER
BCC_MSGUSER miomessaggio
; definisco una struttura "MIA_STRUCT" di tipo STRUCT_P
struct_p MIA_STRUCT
    I32 d1
    I32 d2
    REAL d3
end_struct
; definisco la variabile my_data di tipo MIA_STRUCT
MIA_STRUCT my_data
; trasferisco il contenuto del campo "data" della struttura miomessagg
; nella struttura my_data
move_byte (miomessaggio.data, my_data, sizeof(my_data))
; eseguo delle operazioni sui dati letti
if (my_data.d1 = 27)
...
endif
```

© 2025 Robox SpA

Struttura STRU_AAX

Questa keyword permette di definire una variabile di tipo STRU_AAX. Questa struttura contiene le posizioni degli assi compensate.

Può essere utilizzata nelle funzioni planar_comp_ip519() e planar_comp_cp518().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
axp [1-:-32]	REAL	Array per le posizioni degli assi

Esempi

Esempio di utilizzo:

```
; SAMPLE CODE
; Global Storage definition
;---------------------------
STRU_aax in_cp, out_cp
STRU_aax in_ip, out_ip


in_ip.axp[1] = IP(1)
in_ip.axp[2] = IP(2)
in_ip.axp[3] = IP(3)
; quando il compilatore sarà pronto si scrive:
; esito = PLANAR_COMP_IP (in_ip, out_ip);
; until the compiler it's not ready... using the right files:
; C:\Robox\Rc3e\33.5.16\include\parnames.def
; C:\Robox\Rc3e\33.5.16\include\prdf_str.inr
; C:\Robox\Rc3e\33.5.16\include\struser.h
; C:\Robox\Rc3e\33.5.16\cpu-ppc-p2020\include\rrt\interrt.h
; C:\Robox\Rc3e\33.5.16\cpu-ppc-g2\include\rrt\interrt.h
; you can use the function in the following way
$LIN_STEP planar_comp_ip (&in_ip, &out_ip);
rr(120+1) = out_ip.axp[1]
rr(120+2) = out_ip.axp[2]
rr(120+3) = out_ip.axp[3]
...
...
in_cp.axp[1] = CP(1)
in_cp.axp[2] = CP(2)
in_cp.axp[3] = CP(3)
; quando il compilatore sara'pronto si scrive:
; esito = PLANAR_COMP_CP (in_cp, out_cp);
$LIN_STEP planar_comp_cp (&in_cp, &out_cp)
rr(160+1) = out_cp.axp[1]
rr(160+2) = out_cp.axp[2]
rr(160+3) = out_cp.axp[3]
```

© 2025 Robox SpA

Struttura STRU_AH_D

Questa keyword permette di definire una variabile di tipo STRU_AH_D (STRUCT_P 918).

Viene utilizzata dall'istruzione di gestione allarmi ah_get_d() 159.

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
Id	I32	Identificativo dell'allarme
hour	I32	Ora in cui si è verificato l'allarme
min	I32	Minuto in cui si è verificato l'allarme
sec	I32	Secondo in cui si è verificato l'allarme
day	I32	Giorno in cui si è verificato l'allarme
mon	I32	Mese in cui si è verificato l'allarme
year	I32	Anno in cui si è verificato l'allarme
alnum	I32	Numero dell'allarme
alaxis	I32	Asse a cui si riferisce l'allarme
p1	REAL	Eventuale primo parametro dell'allarme
p2	REAL	Eventuale secondo parametro dell'allarme
p3	REAL	Eventuale terzo parametro dell'allarme

Esempi

Esempio di dichiarazione:

```
; definizione di variabile okkio di tipo STRU_AH_D
STRU_AH_D okkio
```

Esempio di utilizzo:

```
; utilizzo del dato contenuto in un campo della struttura
; definizione di variabile okkio di tipo STRU_AH_D
STRU_AH_D okkio
; definisco le variabili
I32 numero_allarme
I32 reso
I32 indice_allarme = 2
```

© 2025 Robox SpA

```
; leggo un allarme dallo storico allarmi
reso = ah_get_d(159)(indice_all,okkio)
; leggo il valore del campo "Numero dell'allarme" alnum
; e lo utilizzo come variabile di selezione di un blocco SELECT(764) ...
numero_allarme = okkio.alnum
select (numero_allarme)
    case 1
        sr(1) = "ATTENZIONE! ALLARME GRAVE"
        break
    case 2
        sr(1) = "Allarme non bloccante"
        break
    case 3
        sr(1) = "Malfunzionamento"
        break
    default
        sr(1) = "Nessun allarme"
end_select
```

696

Struttura STRU_AH_D_EXT

Questa keyword permette di definire una variabile di tipo STRU_AH_D_EXT (STRUCT_P(918)).

Viene utilizzata dall'istruzione di gestione allarmi ah_get_d()(159).

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
Id	I32	Identificativo dell'allarme
hour	I32	Ora in cui si è verificato l'allarme
min	I32	Minuto in cui si è verificato l'allarme
sec	I32	Secondo in cui si è verificato l'allarme
day	I32	Giorno in cui si è verificato l'allarme
mon	I32	Mese in cui si è verificato l'allarme
year	I32	Anno in cui si è verificato l'allarme
alnum	I32	Numero dell'allarme
alaxis	I32	Asse a cui si riferisce l'allarme o 0
p1	REAL	Eventuale primo parametro numerico dell'allarme

© 2025 Robox SpA

Campo	Tipo	Descrizione
p2	REAL	Eventuale secondo parametro numerico dell'allarme
p3	REAL	Eventuale terzo parametro numerico dell'allarme
us1	STRING[64]	Primo parametro stringa utente USV
us2	STRING[64]	Secondo parametro stringa utente USV
us3	STRING[64]	Terzo parametro stringa utente USV

Esempi

Esempio di utilizzo:

```
stru_ah_info ahi
stru_ah_d_ext ah
int result
int ahidcnt
...
; get alarm history information
ah_info(ahi);
ahidcnt = ahi.oldid
while (ahidcnt <= ahi.newcnt)
    result = ah_get_d(ahIdcnt, ah)
    inc (ahidcnt)
    ...
    ...
end_while
...
```

Struttura STRU_AH_INFO

Questa keyword permette di definire una variabile di tipo STRU_AH_INFO (STRUCT_P 918).

Viene utilizzata dall'istruzione di gestione allarmi ah_info() 162.

E' composta dalle seguenti informazioni:

Campo	Tipo	Description
oldId	I32	Numero progressivo più vecchio degli allarmi
newId	I32	Numero progressivo più nuovo degli allarmi
nal	I32	Numero allarme

© 2025 Robox SpA

Campo	Tipo	Description
ahSize	I32	Dimensione allarme

Esempi

Esempio di utilizzo:

```
BOOL get_ah_info
INT result_ah_info
STRU_AH_INFO info_alarm_histrory
STRING rep_msg

; ...

if (get_ah_info)

    get_ah_info = false
    result_ah_info = ah_info(info_alarm_histrory)

    if (not result_ah_info)
        inf_report(0x20, "Alarm histrory info:")
        str_format(rep_msg, "AH old id: %i", info_alarm_histrory.oldId
        inf_report(0x20, rep_msg)
        str_format(rep_msg, "AH new id: %i", info_alarm_histrory.newId
        inf_report(0x20, rep_msg)
        str_format(rep_msg, "AH num al: %i", info_alarm_histrory.nal)
        inf_report(0x20, rep_msg)
        str_format(rep_msg, "AH size: %i", info_alarm_histrory.ahSize)
        inf_report(0x20, rep_msg)
    else
        inf_report(0x80, "### Alarm history not enabled! ###")
    endif
endif
```

Struttura STRU_AS_D

Validità: **RC3E v33.3.4** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_AS_D.

Viene utilizzata dall'istruzione di gestione allarmi as_get_d().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
alnum	I32	Numero allarme
alaxis	I32	Asse che ha generato l'allarme
p1	REAL	Primo parametro dell'allarme

© 2025 Robox SpA

Campo	Tipo	Descrizione
p2	REAL	Secondo parametro dell'allarme
p3	REAL	Terzo parametro dell'allarme

Esempi

Esempio di utilizzo:

```
; definizione di variabile s_al di tipo STRU_AS_D
STRU_AS_D s_al

; leggo le informazioni sull'eventuale allarme presente in stack
as_get_d(1,s_al)

; eseguo delle operazioni sui dati letti

; verifico se nella posizione letta ci sia allarme 2000 con il primo p
if ((s_al.alnum = 2000) and (s_al.p1 = 23))
...
endif
```

Struttura STRU_AS_D_EXT

Validità: **RTE v34.30.0** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_AS_D_EXT.

Viene utilizzata dall'istruzione di gestione allarmi as_get_d().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
alnum	I32	Numero allarme
alaxis	I32	Asse che ha generato l'allarme o 0
p1	REAL	Primo parametro numerico dell'allarme
p2	REAL	Secondo parametro numerico dell'allarme
p3	REAL	Terzo parametro numerico dell'allarme
us1	STRING[64]	Primo parametro stringa utente USV
us2	STRING[64]	Secondo parametro stringa utente USV

© 2025 Robox SpA

Campo	Tipo	Descrizione
us3	STRING[64]	Terzo parametro stringa utente USV

Esempi

Esempio di utilizzo:

```
; definizione di variabile s_al di tipo STRU_AS_D
STRU_AS_D_EXT s_al

; leggo le informazioni sull'eventuale allarme presente in stack
as_get_d(1, s_al)

; eseguo delle operazioni sui dati letti

; verifico se nella posizione letta ci sia allarme 2000 con il primo p
if ((s_al.alnum = 2000) and (s_al.us1 = "USV"))
...
endif
```

Struttura STRU_AXIO_PDI

Questa keyword permette di definire una variabile di tipo STRU_AXIO_PDI

Questa variabile viene utilizzata dalle funzioni AXIO_ROBJ e AXIO_WOBJ per la lettura/scrittura degli ogetti fieldbus

É composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
rid	INT	Robox ID associato al dispositivo 0 per AXIOLINE su RPx >0 per dispositivo definito AXIOLINE remoto
slot	INT	numero di slot del modulo in questione. 1 è il primo modulo dopo Rpx o BK EC.
index	INT	indice oggetto da leggere o scrivere
subIndex	INT	subindex oggetto da leggere o scrivere
datatype	INT	tipo di dato da leggere o scrivere (vedi tabella codici) Se 0 il tipo di dato viene desunto direttamente dall'oggetto in questione.

© 2025 Robox SpA

Campo	Tipo	Descrizione
		Qesta variabile viene scritta da istruzione con il tipo di dati utilizzato.
datalen	INT	se usata per AXIO_ROBJ: dimensione massima in byte della area utilizzata per memorizzare la risposta. se usata per AXIO_WOBJ: dimensione in byte dei dati da scrivere in caso di scrittura di OCTECT_STRING o VISIBLE_STRING
phase	INT	fase di lavoro/ modalitá di esecuzione: 1--> inizializzazione esecuzione passante (**) -1--> inizializzazione esecuzione bloccante(**) Se AXIO_ROBJ: 2-:-10 --> reserved RTE 11--> operazione eseguita con successo Se AXIO_WOBJ: 2-:-6 --> reserved RTE 7--> operazione eseguita con successo 0--> non inizializzato o errore altro--> valore illegale solo i valori contrassegnati con (**) SONO I SOLI scrivibili in questa variabile da parte dell'utente.
nric	INT	numero di caratteri ricevuti
res_1	INT	reserved
res_2	INT	reserved
res_3	INT	reserved
res_4[256]	U8	reserved

Tabella codici dataType:

Codice	Tipo Dato
0	AUTO-SELECTION (default)

© 2025 Robox SpA

Codice	Tipo Dato
2	INTEGER8
3	INTEGER16
4	INTEGER32
5	UNSIGNED8
6	UNSIGNED16
7	UNSIGNED32
9	VISIBLE_STRING
10	OCTET_STRING
21	INTEGER64
27	UNSIGNED64
-2	ARRAY INTEGER8
-3	ARRAY INTEGER16
-4	ARRAY INTEGER32
-5	ARRAY UNSIGNED8
-6	ARRAY UNSIGNED16
-7	ARRAY UNSIGNED32
-10	OCTET_STRING (usato per strutture speciali (es. tabella parametri CNT2/INC2)

Esempi

Esempio di dichiarazione:

```
; definizione della variabile s di tipo STRU_COE_OBJ
STRU_AXIO_PDI s
```

Struttura STRU_BCC_TCP_CLIENT

Validità: **RC3E v33.9.1-17** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_BCC_TCP_CLIENT.

Questa variabile viene utilizzata dalla funzione R3 bcc_get_tcp_clients_info() per ottenere informazioni riguardanti la lista dei client registrati e sapere se il flusso di messaggi BCC dal client specifico sia ancora attivo o meno.

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
ip_addr	U32	Indirizzo IP del client

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td>port</td><td>U16</td><td>Porta del client</td></tr>
<tr><td>status</td><td>U32</td><td>Stato logico della connessione.
<table>
<tr><th>Bit</th><th>Mask</th><th>Significato</th></tr>
<tr><td>0</td><td>0x00000001</td><td>Il client è connesso</td></tr>
<tr><td>1</td><td>0x00000002</td><td>Flusso di comunicazione attivo</td></tr>
<tr><td>2</td><td>0x00000004</td><td>Flusso di comunicazione mancante (latched)</td></tr>
</table></td></tr>
<tr><td>description</td><td>STRING [66]</td><td>Descrizione del client</td></tr>
<tr><td>last_access</td><td>REAL</td><td>Ultimo tempo valido per ricevere un messaggio BCC prima di impostare "flusso mancante" [ms]</td></tr>
<tr><td>timeout</td><td>U32</td><td>Timeout programmato [ms]</td></tr>
<tr><td>flags</td><td>U32</td><td>Riservato</td></tr>
</table>

Esempi

Esempio di dichiarazione:

```
; definizione della variabile miavariabile di tipo STRU_BCC_TCP_CLIENT
STRU_BCC_TCP_CLIENT miavariabile
```

Esempio di utilizzo:

```
...
; leggo indirizzo ip del client i e lo salvo in un array
cl_ip[i] = miavariabile[i].ip_addr
...
```

Struttura STRU_CAM

Questa keyword permette di definire una variabile di tipo STRU_CAM (STRUCT_P 918).

Viene utilizzata dall'istruzione mv_table 481(), che permette di eseguire una camma per punti.

È composta dalle seguenti informazioni:

© 2025 Robox SpA

Campo	Tipo	Descrizione
x	REAL	coordinata x
y	REAL	coordinata y

Esempi

Esempio di dichiarazione:

```
; definizione di variabile profilo23 di tipo STRU_CAM
STRU_CAM profilo23
```

Esempio di utilizzo:

```
; utilizzo della struttura per un movimento a camma
; definizione di variabile profilo23 di tipo STRU_CAM con sette coppie
STRU_CAM profilo23[7]
; inserimento dei dati nella struttura
if (not (r(100)))
    profilo23[0].x = 0
    profilo23[1].x = 90
    profilo23[2].x = 135
    profilo23[3].x = 180
    profilo23[4].x = 270
    profilo23[5].x = 330
    profilo23[6].x = 360
    profilo23[0].y = 10
    profilo23[1].y = 90
    profilo23[2].y = 250
    profilo23[3].y = 300
    profilo23[4].y = 270
    profilo23[5].y = 100
    profilo23[6].y = 10
    r(100) = 1
endif
...
; esecuzione del movimento a camma
ip(2) = mv_table[481](ip(1) , 7 , profilo23)
```

Struttura STRU_CO_OBJ

Questa keyword permette di definire una variabile di tipo STRU_CO_OBJ.

Questa variabile viene utilizzata dalle funzioni CO_ROBJ[258] e CO_WOBJ[273] per la lettura/scrittura degli ogetti fieldbus.

E' composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
rid	INT	Robox ID associato al dispositivo

© 2025 Robox SpA

Campo	Tipo	Descrizione
objCode	INT	numero oggetto interessato
subIndex	INT	numero subindex interessato
type	INT	tipo di dato secondo la codifica usata per la CO_SEND_SDO 266 /CO_ASW_SDO 239 ossia: 0 = No object 1 = Unsigned 8 bits 2 = Unsigned 16 bits 3 = Unsigned 24 bits 4 = Unsigned 32 bits 5 = Real 32 bit -1 = Signed 8 bits -2 = Signed 16 bits -3 = Signed 24 bits -4 = Signed 32 bits
data	REAL	dato ricevuto / inviato
phase	INT	fase di lavoro/ modalitá di esecuzione: 1--> inizializzazione esecuzione passante (**) -1--> inizializzazione esecuzione bloccante(**) 2--> reserved RTE 3--> reserved RTE 4--> operazione eseguita con successo 0--> non inizializzato o errore altro--> valore illegale solo i valori contrassegnati con (**) SONO I SOLI scrivibili in questa variabile da parte dell'utente.
res_1	INT	

Esempi

Esempio di dichiarazione:

```
; definizione della variabile s di tipo STRU_CO_OBJ
STRU_CO_OBJ s
```

© 2025 Robox SpA

Struttura STRU_COE_OBJ

Validità: **RC3E v33.5.18-2** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_COE_OBJ.

Questa variabile viene utilizzata dalle funzioni COE_ROBJ 279 e COE_WOBJ 284 per la lettura/scrittura degli oggetti fieldbus.

E' composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
rid	INT	Robox ID associato al dispositivo
objCode	INT	numero oggetto interessato
subIndex	INT	numero subindex interessato
datatype	INT	tipo di dato. Vedi tabella sotto.
datalen	INT	se usata per COE_ROBJ: dimensione massima in byte della area per memorizzare la risposta se usata per COE_WOBJ: dimensione in byte dei dati da scrivere.
phase	INT	fase di lavoro/ modalitá di esecuzione: 1--> inizializzazione esecuzione passante (**) -1--> inizializzazione esecuzione bloccante(**) 2--> reserved RTE 3--> reserved RTE 4--> operazione eseguita con successo 0--> non inizializzato o errore altro--> valore illegale solo i valori contrassegnati con (**) SONO I SOLI scrivibili in questa variabile da parte dell'utente.
res_1	INT	reserved (numero di caratteri ricevuti)
res_2	INT	reserved
res_3	INT	reserved

© 2025 Robox SpA

Codici tipi di dato (dataType)

Codice	Tipo di dato
1	BOOLEAN
2	INTEGER8
3	INTEGER16
4	INTEGER32
5	UNSIGNED8
6	UNSIGNED16
7	UNSIGNED32
8	REAL32
9	VISIBLE_STRING
10	OCTET_STRING
11	UNICODE_STRING
12	TIME_OF_DAY
13	TIME_DIFFERENCE
15	DOMAIN
16	INTEGER24
17	REAL64
18	INTEGER40
19	INTEGER48
20	INTEGER56
21	INTEGER64
22	UNSIGNED24
24	UNSIGNED40
25	UNSIGNED48
26	UNSIGNED56
27	UNSIGNED64
30	BYTE
45	BITARR8
46	BITARR16
47	BITARR32

© 2025 Robox SpA

Esempi

Esempio di dichiarazione:

```
; definizione della variabile s di tipo STRU_COE_OBJ
STRU_COE_OBJ s
```

Struttura STRU_COERRMSG

Questa keyword permette di definire una variabile di tipo STRU_COERRMSG (STRUCT_P 918).

Questa variabile viene utilizzata dall'istruzione co_rerr 255() per la gestione dei messaggi CanOpen.

E' composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
errCode	U16	emergency error code ()
errReg	U8	error register
mse[5]	U8	manufacturer specific error field byte 1,2,3,4,5

Esempi

Esempio di definizione:

```
; definizione della variabile analisiCan di tipo STRU_COERRMSG
STRU_COERRMSG analisiCan
```

Esempio di utilizzo:

```
; utilizzo dei dati contenuti nella struttura
; definizione della variabile analisiCan di tipo STRU_COERRMSG
STRU_COERRMSG analisiCan
; definisco una struttura "MIA_STRUCT" di tipo STRUCT_P 918
STRUCT_P MIA_STRUCT
U16 codice_emg
U8 reg_errore
U8 errore_spec[5]
END_STRUCT
; definisco la variabile can_err di tipo MIA_STRUCT
MIA_STRUCT can_err
; definisco la variabile reso e la variabile can_wks
I32 reso
I32 can_wks
; leggo le informazioni sull'eventuale allarme della stazione can
can_wks = 1
reso = co_rerr 255(can_wks, analisiCan)
; trasferisco i contenuto dei campi della struttura analisiCan
; nella struttura can_err
move_byte 731(analisiCan, can_err, SIZEOF(can_err))
; eseguo delle operazioni sui dati letti
```

© 2025 Robox SpA

```
if (can_err.codice_emg = 0x1000)
    sr(1) = "ERRORE GENERICO"
endif
```

Specifiche CiA Draft Standard

Qui di seguito sono riportate le specifiche CiA Draft Standard:

9.2.5.2 Emergency Object Data

The Emergency Telegram consists of 8 bytes with the data as shown in Figure 34: Emergency Object Data.

Byte	0	1	2	3	4	5	6	7
Content	Emergency Error Code (see Table 21)		Error register (Object 1001H)	Manufacturer specific Error Field				

Figure 34: Emergency Object Data

© 2025 Robox SpA

Table 21: Emergency Error Codes

Error Code (hex)	Meaning
00xx	Error Reset or No Error
10xx	Generic Error
20xx	Current
21xx	Current, device input side
22xx	Current inside the device
23xx	Current, device output side
30xx	Voltage
31xx	Mains Voltage
32xx	Voltage inside the device
33xx	Output Voltage
40xx	Temperature
41xx	Ambient Temperature
42xx	Device Temperature
50xx	Device Hardware
60xx	Device Software
61xx	Internal Software
62xx	User Software
63xx	Data Set
70xx	Additional Modules
80xx	Monitoring
81xx	Communication
8110	CAN Overrun (Objects lost)
8120	CAN in Error Passive Mode
8130	Life Guard Error or Heartbeat Error
8140	recovered from bus off
8150	Transmit COB-ID collision
82xx	Protocol Error
8210	PDO not processed due to length error
8220	PDO length exceeded
90xx	External Error
F0xx	Additional Functions
FFxx	Device specific

© 2025 Robox SpA

Table 48: Structure of the Error Register

Bit	M/O	Meaning
0	M	generic error
1	O	current
2	O	voltage
3	O	temperature
4	O	communication error (overrun, error state)
5	O	device profile specific
6	O	Reserved (always 0)
7	O	manufacturer specific

Struttura STRU_CRIMPER

Questa keyword permette di definire una variabile di tipo STRU_CRIMPER (STRUCT_P 918).

Viene utilizzata dall'istruzione di movimento mv_crimper 453().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
mastercontactlength	REAL	Durata tratto in aggancio [0--masterCycle] in unità del master
mastercycle	REAL	Lunghezza ciclo master
qMaster	REAL	Quota attuale del master - Da aggiornare prima di ogni chiamata alla funzione -
qSlave	REAL	Quota attuale dello slave - Da aggiornare prima di ogni chiamata alla funzione -
reservedCrimper1	REAL	riservato
reservedCrimper2	REAL	riservato
status	I32	Stato movimento M_INITIALIZING alla prima chiamata della funzione l'utente deve impostare lo stato a M_INITIALIZING M_HOMING se è in corso la missione di recupero M_SYNCHRONIZED se lo slave è agganciato al master Per i relativi codici, fare riferimento ai codici di reso 63

© 2025 Robox SpA

Campo	Tipo	Descrizione
slavedirection	REAL	Direzione asse slave
slavephase	REAL	Fase dello slave - unità del master - Se da 10 si passa a 20 la differenza (+10) viene aggiunta a mastercycle (solo per un ciclo). Con slavephase = 0, si ha un legame tale per cui ogni volta che un nuovo ciclo master inizia, l'asse crimper si trova all'inizio della sua zona di aggancio.
slavecycle	REAL	Lunghezza ciclo slave - in unità slave
kslavemaster	REAL	K di adattamento unità di grandezza master/slave Se unità master in mm e unità slave in mm K=1 Se unità master in mm e unità slave in gradi , ad esempio, se ogni 500 mm di avanzamento del master il coltello deve fare 360 gradi allora kslavemaster = 2 * K_PI * raggio coltello / 360

Esempi

Esempio di dichiarazione:

```
; definizione di variabile parGan1 di tipo STRU_CRIMPER
STRU_CRIMPER parGan1
```

Esempio di utilizzo:

```
; utilizzo dei dati contenuti nella struttura
; definizione di variabile parGan1 di tipo STRU_CRIMPER
STRU_CRIMPER parGan1
; definizione variabili accessorie
I32 esito
REAL reso
; inserimento dei dati nella struttura
if (not (r(100)))
    parGan1.mastercontactlength = 500
    parGan1.mastercycle = 2000
    parGan1.status = 1                          ; 1 è il codice di M_INIT
    parGan1.slavephase = 25
    parGan1.slavecycle = 800
    parGan1.kslavemaster = 2
```

© 2025 Robox SpA

```
    r(100) = 1
endif
; prima di chiamare la funzione mv_crimper()
; aggiorno le quote del master e dello slave
parGan1.qMaster = cp(1)
parGan1.qSlave = cp(2)
; eseguo la funzione mv_crimper()
reso = mv_crimper ( esito , parGan1 )
; valuto la variabile esito
if ( esito = 8 )                          ; 8 è il codice di M_SYNCHRON
    sr(1) = "Sono sincronizzato"
endif
```

Struttura STRU_ECAT_PROBE

Validità: **RC3E v33.7-1-beta0** (o successive)

Questa keyword definisce una variabile di tipo STRU_ECAT_PROBE.

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
phase	U32	fase di lavoro interna va settata a 0 alla partenza
err	I32	codice errore interno: • -14 (-EFAULT) pointer nullo o fase illegale • -5 (-EIO) cavo non collegato • -2 (-ENOENT) nessun dispositivo trovato • -19 (-ENODEV) il dispositivo selezionato non esiste
n_pos	U32	posizione ordinale nella linea ethercat (1 -: 1024)
how_many	U32	numero di dispositivi presenti sulla linea ethercat
al_status	U16	ritorna: • 0x01 Init • 0x02 Pre-Operational • 0x03 Bootstrap • 0x04 Safe-Operational • 0x08 Operational • 0x11 Init + err • 0x12 Pre-Operational + err

© 2025 Robox SpA

Campo	Tipo	Descrizione
		• 0x13 Bootstrap + err • 0x14 Safe-Operational + err • 0x18 Operational + err
al_status_code	U16	AL status code del dispositivo
config_addr	U16	indirizzo configurato del dispositivo
alias_addr	U16	indirizzo configurato del dispositivo
vendor_id	U32	vendor ID
product_code	U32	product code del dispositivo
revision_number	U32	revision number del dispositivo in EEP
serial_number	U32	serial number del dispositivo in EEP

Esempi

Esempio di dichiarazione:

```
; definizione di variabile handle di tipo STRU_ECAT_PROBE
STRU_ECAT_PROBE sP
```

Struttura STRU_FILEHND

Questa keyword permette di definire una variabile di tipo STRU_FILEHND (STRUCT_P 918).

Viene utilizzata dalle istruzioni di gestione flash f_open 327(), f_close 318(), f_read 330(), f_gets 324(), ...

E' composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
h	I32	handle

Esempi

Esempio di dichiarazione:

```
; definizione di variabile handle di tipo STRU_FILEHND
STRU_FILEHND handle
```

Esempio di utilizzo:

```
; utilizzo dei dati contenuti nella struttura con la funzione f_open()
; definizione di variabile handle di tipo STRU_FILEHND
```

© 2025 Robox SpA

```
STRU_FILEHND handle
; definizione delle veriabile ausiliarie
STRING nome_file                        ; è il nome del file da aprire
STRING modo_open                        ; è la modalità di apertura
I32 car_max = 0                         ; deve essere zero
I32 reso                                ; risultato dell'operazione
; assegno un valore alle variabili
nome_file = sr(1)
modo_open = sr(2)
; eseguo la funzione di apertura file
reso = f_open(nome_file, modo_open, car_max, handle)
; eseguo delle operazioni sui dati letti
if (reso)
    sr(3) = "Ho aperto il file " # nome_file
else
    sr(3) = "Il file " # nome_file # " non esiste!"
endif
```

Struttura STRU_FILEINFO

Questa keyword permette di definire una variabile di tipo STRU_FILEINFO (STRUCT_P).

Viene utilizzata dall'istruzione f_dir() per la lettura delle informazioni sui file contenuti nelle memorie flash.

E' composto dalle seguenti informazioni:

Campo	Tipo	Descrizione
reserved[16]	STRING	Riservato
len	U32	Lunghezza del file
name[128]	STRING	Nome del file

Esempi

Esempio di definizione:

```
; definizione di variabile archivio di tipo STRU_FILEINFO
STRU_FILEINFO archivio
```

Esempio

```
; utilizzo dei dati contenuti nella struttura
; definizione di variabile archivio di tipo STRU_FILEINFO
STRU_FILEINFO archivio
; definizione delle veriabile ausiliarie
STRING banco_f                          ; è il nome del banco della flas
I32 num_file                            ; è il numero dei file da legger
I32 reso                                ; risultato dell'operazione
; assegno un valore alle variabili
```

© 2025 Robox SpA

```
banco_f = "/FA/*.*"
; uso il valore salvato nel registro 220
num_file = r(220)
; eseguo la funzione di apertura file
reso = f_dir(banco_f, num_file, archivio)
; eseguo delle operazioni sui dati letti
if (reso)
    sr(3) = "Ci sono " # str_i(reso, 1, 10) # " file nella flash"
else
    sr(3) = "La flash è vuota!"
endif
```

Struttura STRU_FOLLOW

Questa keyword permette di definire una variabile di tipo STRU_FOLLOW (STRUCT_P).

Viene utilizzata dall'istruzione di movimento mv_follow().

È composto dalle seguenti informazioni:

Campo	Tipo	Descrizione
masterDirection	REAL	1 se l'asse master evolve verso quote positive -1 se l'asse master evolve verso quote negative
qSyncMaster	REAL	Quota di sincronismo dell'asse master
masterCycle	REAL	Passo di incremento della quota qSyncMaster per gestire l'inseguimento ciclico ripetuto (senza segno).
qMaster	REAL	Posizione della grandezza master. Deve essere aggiornata prima di ogni chiamata di funzione.
velMaster	REAL	Velocità della grandezza master. Deve essere aggiornata prima di ogni chiamata di funzione.
si	REAL	Tempo di campionamento si
slaveDirection	REAL	1 se l'asse slave evolve verso quote positive -1 se l'asse slave evolve verso quote negative
qSyncSlave	REAL	Quota inizio sincronismo dell'asse slave

© 2025 Robox SpA

Campo	Tipo	Descrizione
qSyncOut	REAL	Quota fine sincronismo dell'asse slave
qParkSlave	REAL	Quota di parcheggio dell'asse slave
slaveCycle	REAL	Passo di incremento della quota qSyncSlave per gestire l'inseguimento ciclico ripetuto (senza segno)
velMaxSlave	REAL	È il limite superiore di velocità ammessa per il movimento dell'asse slave. Nel caso la formula di correzione richiedesse di salire al di sopra della velocità massima, il valore reso dalla funzione sarebbe pari a VelMaxSlave.
accMaxSlave	REAL	È il valore di accelerazione massima accettato per l'asse slave.
kSlaveMaster	REAL	K di adattamento master/slave es: se un coltello lungo 250 mm kslavemaster=K_DUEPI 17 *0.25/360
qSlave	REAL	Posizione della grandezza slave. Deve essere aggiornata prima di ogni chiamata di funzione.
velSlave	REAL	Velocità della grandezza slave. Deve essere aggiornata prima di ogni chiamata di funzione.
status	I32	M_POSITIONING: L'asse slave si sta portando alla posizione di parcheggio N.B. Alla prima chiamata della funzione l'utente deve impostare status a M_POSITIONING. MV_WAITING: L'asse slave sta aspettando di partire M_WAITING_NEXT_CYCLE: Non ci sono le condizioni per partire; aspetto il prossimo appuntamento M_REACHED: L'asse slave si è agganciato Per i relativi codici, fare riferimento ai codici di reso 63

© 2025 Robox SpA

Esempi

Esempio di dichiarazione:

```
; definizione di variabile sega1 di tipo STRU_FOLLOW
STRU_FOLLOW sega1
```

Esempio di utilizzo:

```
; utilizzo dei dati contenuti nella struttura
; definizione di variabile sega1 di tipo STRU_FOLLOW
STRU_FOLLOW sega1
; definizione variabili accessorie
I32 esito
REAL reso
; inserimento dei dati nella struttura
if (not(r(100)))
    sega1.masterDirection = 1
    sega1.qSyncMaster = 400
    sega1.masterCycle = 50
    sega1.si = si
    sega1.slaveDirection = 1
    sega1.qSyncSlave = 80
    sega1.qSyncOut = 300
    sega1.qParkSlave = 250
    sega1.slaveCycle = 30
    sega1.velMaxSlave = 60
    sega1.accMaxSlave = 10
    sega1.kSlaveMaster = 2
    sega1.status = 2                                ; 2 è il codice di
    r(100) = 1
endif
; prima di chiamare la funzione mv_follow()
; aggiorno le quote e la velocità del master e dello slave
sega1.qMaster = cp(1)
sega1.velMaster = cv(1)
sega1.qSlave = cp(2)
sega1.velSlave = cv(2)
; eseguo la funzione mv_follow()
reso = mv_follow( esito, sega1 )
; valuto la variabile esito
if (esito = 6)                                  ; 6 è il codice di M_REACH
    sr(1) = "Asse agganciato"
endif
```

Struttura STRU_FOLLOW2

Validità: **RC3E v33.5.8-7** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_FOLLOW2 (STRUCT_P).

Viene utilizzata dall'istruzione di movimento mv_follow2().

È composto dalle seguenti informazioni:

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td>command</td><td>U32</td><td>Parola di comando e configurazione:
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td colspan="3">Comandi configurabili runtime:</td></tr>
<tr><td>0</td><td>0x00000001</td><td>Se impostato a 1 si ha l'arresto dello slave a qParkSlave durante il ciclo successivo (il ciclo successivo non viene eseguito)
(attivo su livello)</td></tr>
<tr><td>1</td><td>0x00000002</td><td>Se impostato a 1 si ha il posizionamento iniziale a qParkSlave
(autoresettato da RTE)</td></tr>
<tr><td>2</td><td>0x00000004</td><td>Se impostato a 1 si ha il posizionamento iniziale rispetto al ciclo master
(autoresettato da RTE)</td></tr>
<tr><td>3</td><td>0x00000008</td><td>Forzamento fine tratto di sincronismo
(autoresettato da RTE)</td></tr>
<tr><td>4</td><td>0x00000010</td><td>Se slave di tipo lineare, forza l'arresto alla fine del</td></tr>
</table></td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>ciclo di andata fino a che tale flag è presente
(attivo su livello)</td></tr>
<tr><td>5</td><td>0x00000020</td><td>Indica che la quota di sync master fornita non è ancora quella definitiva (solo presunta). Utile se utilizzata con OB ELEMENT LOCATION. In tal caso il prossimo aggancio non viene eseguito e lo slave rimane fermo
(attivo su livello)</td></tr>
<tr><td colspan="3">Flag di configurazione:</td></tr>
<tr><td>16</td><td>0x00010000</td><td>• 0, master a ciclo fisso (masterCycle = lunghezza ciclo, qSyncMaster = quota di sync relativa al ciclo)
• 1, master a ciclo variabile</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>(masterCycle = lunghezza minima ciclo, qSyncMaster = quota di sync assoluta aggiornata ad ogni ciclo, ad esempio sulla lettura di una fotocellula)</td></tr>
<tr><td>17</td><td>0x00020000</td><td>Se impostato a 1 si ha la disabilitazione del test su maxSpe</td></tr>
<tr><td>18</td><td>0x00040000</td><td>Se impostato a 1 si ha la disabilitazione del test su maxAcc</td></tr>
<tr><td>19</td><td>0x00080000</td><td>• 0, qSyncMaster allineata a qSyncSlave
• 1, qSyncMaster allineata al punto medio tra qSyncSlave e qOutSlave</td></tr>
<tr><td>20</td><td>0x00100000</td><td>Valido solo per slave ciclico</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>(slaveCycle >0).
1, abilita fermata slave a qParkSlave ad ogni ciclo per la durata spaceWaitInPark
In caso di slave lineare (slaveCycle =0) la fermata a quota di parcheggio è implicita</td></tr>
<tr><td>21</td><td>0x00200000</td><td>Valido solo per slave ciclico (slaveCycle >0).
0, lo slave può muoversi anche all'indietro (durante il posizionamento)
1, lo slave può muoversi solo in avanti</td></tr>
<tr><td>22</td><td>0x00400000</td><td>Valido solo per slave ciclico (slaveCycle >0), master a ciclo variabile (BIT 16=1) e fermata a</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td><table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>qParkSlave disabilitata (BIT 20=0)
1, abilita recupero del sincronismo al volo durante la rampa di frenata</td></tr>
</table></td></tr>
<tr><td>errorCode</td><td>U32</td><td>Codici di errore.
Vedi tabella sotto.</td></tr>
<tr><td></td><td></td><td>//Dati Master//</td></tr>
<tr><td>masterCycle</td><td>REAL</td><td>Passo di ciclo master se master a ciclo fisso (B16= 0).
Passo minimo di ciclo master se master a ciclo variabile (B16=1).
Nota: se master a ciclo variabile può anche essere lasciato a 0, il tal caso però non vengono fatti controlli sulla velocità di recupero.</td></tr>
<tr><td>qSyncMaster</td><td>REAL</td><td>Quota master usata come riferimento di sincronismo.
Se master a ciclo fisso (B16=0) viene intesa come quota relativa al ciclo e va impostata una tantum, mentre se master a ciclo variabile (B16=1) è una quota assoluta che va reimpostata ad ogni ciclo. Ad esempio sulla fotocellula</td></tr>
<tr><td>qMaster</td><td>REAL</td><td>Posizione del master. Deve essere aggiornata sempre, prima di ogni chiamata di funzione. Se asse corrisponde alla IP</td></tr>
<tr><td>velMaster</td><td>REAL</td><td>Velocità del master. Deve essere aggiornata sempre, prima di ogni chiamata di funzione. Se asse corrisponde alla IV</td></tr>
<tr><td>accMaster</td><td>REAL</td><td>Accelerazione del master. Deve essere aggiornata sempre, prima</td></tr>
</table>

© 2025 Robox SpA

Campo	Tipo	Descrizione
		di ogni chiamata di funzione. Se asse corrisponde alla IA Nota: questo parametro serve solo per calcolare in modo corretto accSlave.
		//DATI SLAVE//
slaveCycle	REAL	Passo di incremento della quota qSyncSlave per gestire l'inseguimento ciclico ripetuto. (se slave lineare va lasciato a 0).
qSyncSlave	REAL	Quota inizio sincronismo dell'asse slave
qOutSlave	REAL	Quota fine sincronismo dell'asse slave
qParkSlave	REAL	Quota di parcheggio dell'asse slave
spaceStopSlave	REAL	Spazio di arresto dello slave a partire da qOutSlave. Valido solo se slave lineare (slaveCycle = 0).
spaceWaitInPark	REAL	Spazio di avanzamento master durante il quale l'asse slave resta fermo alla quota di parcheggio qParkSlave ad ogni ciclo prima di iniziare l'inseguimento successivo. Nota: questo parametro ha validità solo se lo slave è lineare (slaveCycle = 0) oppure se lo slave è ciclico ed è abilitata la fermata a quota qParkSlave ad ogni ciclo (B20=1).
velPositSlave	REAL	Velocità posizionamento iniziale su qParkSlave
accPositSlave	REAL	Accelerazione posizionamento iniziale su qParkSlave
jerkPositSlave	REAL	Jerk posizionamento iniziale su qParkSlave
kSlaveMaster	REAL	Costante di conversione da velocità master a velocità slave usata durante il tratto sincronizzato (velSlave = velMaster * kSlaveMaster) Nota: serve ad uniformare le unità di misura nel caso in cui siano differenti, oppure a differenziare le velocità durante il tratto di sincronizzato.

© 2025 Robox SpA

Campo	Tipo	Descrizione
qSlave	REAL	Quota attuale slave aggiornata runtime dalla funzione (la prima volta che si usa la funzione va inizializzata al valore reale a cui si trova l'asse slave).
vSlave	REAL	Velocità attuale slave aggiornata runtime dalla funzione
aSlave	REAL	Accelerazione attuale slave aggiornata runtime dalla funzione. Nota: il suo valore è corretto solo se è stato impostato correttamente il parametro accMaster.
maxSpe	REAL	È il valore di velocità massima per il movimento della grandezza slave. Utilizzata se abilitato test relativo (B17=0)
maxAcc	REAL	È il valore di accelerazione massima per il movimento della grandezza slave. Utilizzata se abilitato test relativo (B18=0)
		//STORAGE INTERNO//
resDouble[14]	REAL	Reserved
resInt[2]	REAL	Reserved

Tabella codici di errore

Valore	Significato
< 0	Errore interno (contattare Robox)
0	Nessun errore
1	*maxAcc* non valida
2	Spazio slave di partenza o di arresto insufficiente
3	*kSlaveMaster* non valido
4	Spazio master di partenza + frenata insufficiente
5	Spazio master di recupero insufficiente
6	Ciclo master non impostato con master a ciclo fisso

© 2025 Robox SpA

Valore	Significato
7	Impossibile effettuare il posizionamento (parametri cinematici di posizionamento invalidi)
8	Superamento massima velocità in fase di inseguimento
9	Superamento massima accelerazione in fase di inseguimento
10	Superamento massima velocità in fase di sincronismo
11	Superamento massima accelerazione in fase di sincronismo
12	Superamento massima velocità in fase di arresto
13	Superamento massima accelerazione in fase di arresto
14	Superamento massima velocità in fase di homing
15	Superamento massima accelerazione in fase di homing

Esempi

Esempio di utilizzo:

```
; definizione di variabile sega1 di tipo STRU_FOLLOW2
STRU_FOLLOW2 sega1
result = mv_follow2(sega1)
set_kinematics(ax1, sega1.qSlave, sega1.velSlave, sega1.accSlave)
```

Struttura STRU_FOLLOW2_INFO

Validità: **RC3E v33.5.8-7** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_FOLLOW2_INFO (STRUCT_P).

Viene utilizzata dall'istruzione di movimento mv_follow2().

E' composta dalle seguenti informazioni in sola lettura:

Campo	Tipo	Descrizione
qmastertostart	REAL	Spazio mancante al master per raggiungere la quota di inizio inseguimento.

© 2025 Robox SpA

Campo	Tipo	Descrizione
qmastertosync	REAL	Spazio mancante al master per raggiungere la quota di inizio sincronismo.
qmastertoout	REAL	Spazio mancante al master per raggiungere la quota di inizio fine sincronismo.

Esempi

Esempio di utilizzo:

```
; definizione di variabile sega1 di tipo STRU_FOLLOW2
STRU_FOLLOW2_INFO infoShear
timetoStart = infoShear.qmastertostart / IV(master)
```

Struttura STRU_FOLLOW2_S

Validità: **RC3E v33.5.8-7** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_FOLLOW2_S (STRUCT_P 918).

Viene utilizzata dall'istruzione di movimento mva_follow2 458 ().

È composto dalle seguenti informazioni:

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td>nAxS</td><td>U32</td><td>Numero asse slave</td></tr>
<tr><td>command</td><td>U32</td><td>Parola di comando e configurazione:
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td colspan="3">Comandi configurabili runtime:</td></tr>
<tr><td>0</td><td>0x00000001</td><td>Se impostato a 1 si ha l'arresto dello slave a qParkSlave durante il ciclo successivo (il ciclo successivo non viene eseguito)
(attivo su livello)</td></tr>
</table></td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td>1</td><td>0x00000002</td><td>Se impostato a 1 si ha il posizionamento iniziale a qParkSlave
(autoresettato da RTE)</td></tr>
<tr><td>2</td><td>0x00000004</td><td>Se impostato a 1 si ha il posizionamento iniziale rispetto al ciclo master
(autoresettato da RTE)</td></tr>
<tr><td>3</td><td>0x00000008</td><td>Forzamento fine tratto di sincronismo
(autoresettato da RTE)</td></tr>
<tr><td>4</td><td>0x00000010</td><td>Se slave di tipo lineare, forza l'arresto alla fine del ciclo di andata fino a che tale flag è presente
(attivo su livello)</td></tr>
<tr><td>5</td><td>0x00000020</td><td>Indica che la quota di sync master fornita non è ancora quella definitiva (solo presunta). Utile se utilizzata</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>con OB ELEMENT LOCATION. In tal caso il prossimo aggancio non viene eseguito e lo slave rimane fermo
(attivo su livello)</td></tr>
<tr><td colspan="3">Flag di configurazione:</td></tr>
<tr><td>16</td><td>0x00010000</td><td>• 0, master a ciclo fisso (masterCycle = lunghezza ciclo, qSyncMaster = quota di sync relativa al ciclo)
• 1, master a ciclo variabile (masterCycle = lunghezza minima ciclo, qSyncMaster = quota di sync assoluta aggiornata ad ogni ciclo, ad esempio sulla lettura di una fotocellula)</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td>17</td><td>0x00020000</td><td>Se impostato a 1 si ha la disabilitazione del test su maxSpe</td></tr>
<tr><td>18</td><td>0x00040000</td><td>Se impostato a 1 si ha la disabilitazione del test su maxAcc</td></tr>
<tr><td>19</td><td>0x00080000</td><td>• 0, qSyncMaster allineata a qSyncSlave
• 1, qSyncMaster allineata al punto medio tra qSyncSlave e qOutSlave</td></tr>
<tr><td>20</td><td>0x00100000</td><td>Valido solo per slave ciclico (slaveCycle >0).
• 1, abilita fermata slave a qParkSlave ad ogni ciclo per la durata spaceWaitInPark
In caso di slave lineare (slaveCycle =0) la fermata a quota di parcheggio è implicita</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td>21</td><td>0x00200000</td><td>Valido solo per slave ciclico (slaveCycle >0).
0, lo slave può muoversi anche all'indietro (durante il posizionamento)
1, lo slave può muoversi solo in avanti</td></tr>
<tr><td>22</td><td>0x00400000</td><td>Valido solo per slave ciclico (slaveCycle >0), master a ciclo variabile (BIT 16=1) e fermata a qParkSlave disabilitata (BIT 20=0)
1, abilita recupero del sincronismo al volo durante la rampa di frenata</td></tr>
</table>
</td></tr>
<tr><td>errorCode</td><td>U32</td><td>Codici di errore.
Vedi tabella codici di errore sotto.</td></tr>
<tr><td></td><td></td><td>//Dati Master//</td></tr>
<tr><td>masterCycle</td><td>REAL</td><td>Passo di ciclo master se master a ciclo fisso (B16= 0).</td></tr>
</table>

© 2025 Robox SpA

Campo	Tipo	Descrizione
		Passo minimo di ciclo master se master a ciclo variabile (B16=1). Nota: se master a ciclo variabile può anche essere lasciato a 0, il tal caso però non vengono fatti controlli sulla velocità di recupero.
qSyncMaster	REAL	Quota master usata come riferimento di sincronismo. Se master a ciclo fisso (B16=0) viene intesa come quota relativa al ciclo e va impostata una tantum, mentre se master a ciclo variabile (B16=1) è una quota assoluta che va reimpostata ad ogni ciclo. Ad esempio sulla fotocellula
qMaster	REAL	Posizione del master. Deve essere aggiornata sempre, prima di ogni chiamata di funzione. Se asse corrisponde alla IP
velMaster	REAL	Velocità del master. Deve essere aggiornata sempre, prima di ogni chiamata di funzione. Se asse corrisponde alla IV
accMaster	REAL	Accelerazione del master. Deve essere aggiornata sempre, prima di ogni chiamata di funzione. Se asse corrisponde alla IA Nota: questo parametro serve solo per calcolare in modo corretto accSlave.
		//DATI SLAVE//
slaveCycle	REAL	Passo di incremento della quota qSyncSlave per gestire l'inseguimento ciclico ripetuto. (se slave lineare va lasciato a 0).
qSyncSlave	REAL	Quota inizio sincronismo dell'asse slave
qOutSlave	REAL	Quota fine sincronismo dell'asse slave
qParkSlave	REAL	Quota di parcheggio dell'asse slave
spaceStopSlave	REAL	Spazio di arresto dello slave a partire da qOutSlave. Valido solo se slave lineare (slaveCycle = 0).

© 2025 Robox SpA

Campo	Tipo	Descrizione
spaceWaitInPark	REAL	Spazio di avanzamento master durante il quale l'asse slave resta fermo alla quota di parcheggio qParkSlave ad ogni ciclo prima di iniziare l'inseguimento successivo. Nota: questo parametro ha validità solo se lo slave è lineare (slaveCycle = 0) oppure se lo slave è ciclico ed è abilitata la fermata a quota qParkSlave ad ogni ciclo (B20=1).
velPositSlave	REAL	Velocità posizionamento iniziale su qParkSlave
accPositSlave	REAL	Accelerazione posizionamento iniziale su qParkSlave
jerkPositSlave	REAL	Jerk posizionamento iniziale su qParkSlave
kSlaveMaster	REAL	Costante di conversione da velocità master a velocità slave usata durante il tratto sincronizzato (velSlave = velMaster * kSlaveMaster) Nota: serve ad uniformare le unità di misura nel caso in cui siano differenti, oppure a differenziare le velocità durante il tratto di sincronizzato.

Tabella codici di errore

Valore	Significato
< 0	Errore interno (contattare Robox)
0	Nessun errore
1	*maxAcc* non valida
2	Spazio slave di partenza o di arresto insufficiente
3	*kSlaveMaster* non valido
4	Spazio master di partenza + frenata insufficiente
5	Spazio master di recupero insufficiente
6	Ciclo master non impostato con master a ciclo fisso

© 2025 Robox SpA

Valore	Significato
7	Impossibile effettuare il posizionamento (parametri cinematici di posizionamento invalidi)
8	Superamento massima velocità in fase di inseguimento
9	Superamento massima accelerazione in fase di inseguimento
10	Superamento massima velocità in fase di sincronismo
11	Superamento massima accelerazione in fase di sincronismo
12	Superamento massima velocità in fase di arresto
13	Superamento massima accelerazione in fase di arresto
14	Superamento massima velocità in fase di homing
15	Superamento massima accelerazione in fase di homing

Esempi

Esempio di utilizzo:

```
; definizione di variabile sega1 di tipo STRU_FOLLOW2_S
STRU_FOLLOW2_S mf2
result = mva_follow2 458 (mf2)
```

Struttura STRU_FOLLOW2_SM

Validità: **RC3E v33.5.8-7** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_FOLLOW2_SM (STRUCT_P 918).

Viene utilizzata dall'istruzione di movimento mva_follow2_m 458().

È composto dalle seguenti informazioni:

Campo	Tipo	Descrizione
nAxM	U32	Numero asse master
nAxS	U32	Numero asse slave
command	U32	Parola di comando e configurazione:

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td colspan="3">Comandi configurabili runtime:</td></tr>
<tr><td>0</td><td>0x00000001</td><td>Se impostato a 1 si ha l'arresto dello slave a qParkSlave durante il ciclo successivo (il ciclo successivo non viene eseguito)
(attivo su livello)</td></tr>
<tr><td>1</td><td>0x00000002</td><td>Se impostato a 1 si ha il posizionamento iniziale a qParkSlave
(autoresettato da RTE)</td></tr>
<tr><td>2</td><td>0x00000004</td><td>Se impostato a 1 si ha il posizionamento iniziale rispetto al ciclo master
(autoresettato da RTE)</td></tr>
<tr><td>3</td><td>0x00000008</td><td>Forzamento fine tratto di sincronismo
(autoresettato da RTE)</td></tr>
<tr><td>4</td><td>0x00000010</td><td>Se slave di tipo lineare, forza l'arresto alla fine del ciclo di andata fino</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>a che tale flag è presente
(attivo su livello)</td></tr>
<tr><td>5</td><td>0x00000020</td><td>Indica che la quota di sync master fornita non è ancora quella definitiva (solo presunta). Utile se utilizzata con OB ELEMENT LOCATION. In tal caso il prossimo aggancio non viene eseguito e lo slave rimane fermo
(attivo su livello)</td></tr>
<tr><td colspan="3">Flag di configurazione:</td></tr>
<tr><td>16</td><td>0x00010000</td><td>• 0, master a ciclo fisso (masterCycle = lunghezza ciclo, qSyncMaster = quota di sync relativa al ciclo)
• 1, master a ciclo variabile (masterCycle =</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>lunghezza minima ciclo, qSyncMaster = quota di sync assoluta aggiornata ad ogni ciclo, ad esempio sulla lettura di una fotocellula)</td></tr>
<tr><td>17</td><td>0x00020000</td><td>Se impostato a 1 si ha la disabilitazione del test su maxSpe</td></tr>
<tr><td>18</td><td>0x00040000</td><td>Se impostato a 1 si ha la disabilitazione del test su maxAcc</td></tr>
<tr><td>19</td><td>0x00080000</td><td>• 0, qSyncMaster allineata a qSyncSlave
• 1, qSyncMaster allineata al punto medio tra qSyncSlave e qOutSlave</td></tr>
<tr><td>20</td><td>0x00100000</td><td>Valido solo per slave ciclico (slaveCycle >0).</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>• 1, abilita fermata slave a qParkSlave ad ogni ciclo per la durata spaceWaitInPark

In caso di slave lineare (slaveCycle =0) la fermata a quota di parcheggio è implicita</td></tr>
<tr><td>21</td><td>0x00200000</td><td>Valido solo per slave ciclico (slaveCycle >0).

• 0, lo slave può muoversi anche all'indietro (durante il posizionamento)

• 1, lo slave può muoversi solo in avanti</td></tr>
<tr><td>22</td><td>0x00400000</td><td>Valido solo per slave ciclico (slaveCycle >0), master a ciclo variabile (BIT 16=1) e fermata a qParkSlave</td></tr>
</table>
</td></tr>
</table>

© 2025 Robox SpA

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td><table>
<tr><th>Bit</th><th>Valore esadecimale</th><th>Descrizione</th></tr>
<tr><td></td><td></td><td>disabilitata (BIT 20=0)
1, abilita recupero del sincronismo al volo durante la rampa di frenata</td></tr>
</table></td></tr>
<tr><td>errorCode</td><td>U32</td><td>Codici di errore.
Vedi tabella codici di errore sotto.</td></tr>
<tr><td></td><td></td><td>//Dati Master//</td></tr>
<tr><td>masterCycle</td><td>REAL</td><td>Passo di ciclo master se master a ciclo fisso (B16= 0).
Passo minimo di ciclo master se master a ciclo variabile (B16=1).
Nota: se master a ciclo variabile può anche essere lasciato a 0, il tal caso però non vengono fatti controlli sulla velocità di recupero.</td></tr>
<tr><td>qSyncMaster</td><td>REAL</td><td>Quota master usata come riferimento di sincronismo.
Se master a ciclo fisso (B16=0) viene intesa come quota relativa al ciclo e va impostata una tantum, mentre se master a ciclo variabile (B16=1) è una quota assoluta che va reimpostata ad ogni ciclo. Ad esempio sulla fotocellula</td></tr>
<tr><td></td><td></td><td>//DATI SLAVE//</td></tr>
<tr><td>slaveCycle</td><td>REAL</td><td>Passo di incremento della quota qSyncSlave per gestire l'inseguimento ciclico ripetuto. (se slave lineare va lasciato a 0).</td></tr>
<tr><td>qSyncSlave</td><td>REAL</td><td>Quota inizio sincronismo dell'asse slave</td></tr>
<tr><td>qOutSlave</td><td>REAL</td><td>Quota fine sincronismo dell'asse slave</td></tr>
</table>

© 2025 Robox SpA

Campo	Tipo	Descrizione
qParkSlave	REAL	Quota di parcheggio dell'asse slave
spaceStopSlave	REAL	Spazio di arresto dello slave a partire da qOutSlave. Valido solo se slave lineare (slaveCycle = 0).
spaceWaitInPark	REAL	Spazio di avanzamento master durante il quale l'asse slave resta fermo alla quota di parcheggio qParkSlave ad ogni ciclo prima di iniziare l'inseguimento successivo. Nota: questo parametro ha validità solo se lo slave è lineare (slaveCycle = 0) oppure se lo slave è ciclico ed è abilitata la fermata a quota qParkSlave ad ogni ciclo (B20=1).
velPositSlave	REAL	Velocità posizionamento iniziale su qParkSlave
accPositSlave	REAL	Accelerazione posizionamento iniziale su qParkSlave
jerkPositSlave	REAL	Jerk posizionamento iniziale su qParkSlave
kSlaveMaster	REAL	Costante di conversione da velocità master a velocità slave usata durante il tratto sincronizzato (velSlave = velMaster * kSlaveMaster) Nota: serve ad uniformare le unità di misura nel caso in cui siano differenti, oppure a differenziare le velocità durante il tratto di sincronizzato.

Tabella codici di errore

Valore	Significato
< 0	Errore interno (contattare Robox)
0	Nessun errore
1	*maxAcc* non valida
2	Spazio slave di partenza o di arresto insufficiente
3	*kSlaveMaster* non valido
4	Spazio master di partenza + frenata insufficiente

© 2025 Robox SpA

Valore	Significato
5	Spazio master di recupero insufficiente
6	Ciclo master non impostato con master a ciclo fisso
7	Impossibile effettuare il posizionamento (parametri cinematici di posizionamento invalidi)
8	Superamento massima velocità in fase di inseguimento
9	Superamento massima accelerazione in fase di inseguimento
10	Superamento massima velocità in fase di sincronismo
11	Superamento massima accelerazione in fase di sincronismo
12	Superamento massima velocità in fase di arresto
13	Superamento massima accelerazione in fase di arresto
14	Superamento massima velocità in fase di homing
15	Superamento massima accelerazione in fase di homing

Esempi

Esempio di utilizzo:

```
; definizione di variabile sega1 di tipo STRU_FOLLOW2_SM
STRU_FOLLOW2_SM myShear
result = mva_follow2_m 458(myShear)
```

Struttura STRU_GROR

Questa keyword permette di definire una variabile di tipo STRU_GROR (STRUCT_P 918).

Viene utilizzata dalle istruzioni group 721 e order 735, che servono per attivare gli esecutori delle regole (RULE) e definirne l'ordine.

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
idx[1;32]	I32	Numero della Rule

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

```
; definizione della variabile attivaRule di tipo STRU_GROR
STRU_GROR attivaRule
; rc(1) verrà eseguito per primo
attivaRule.idx[1] = 1
; rc(2) verrà eseguito per secondo
attivaRule.idx[2] = 2
; rc da 3 a 32 saranno disabilitati
attivaRule.idx[3] = -1
; assegnare l'ordine
order(attivaRule)
```

Struttura STRU_HW_INFO

Validità: **RC3E v33.5.23-3** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_HW_INFO (STRUCT_P).

Viene utilizzata dalla funzione get_hw_info().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
board_code_l	U32	Codice basso della scheda
board_code_h	U32	Codice alto della scheda
iwow	INT	numero della prima input word e della prima output word associata alla scheda
name	STRING[64]	nome della scheda

Esempi

Esempio di utilizzo:

```
; utilizzando l'istruzione get_hw_info con nSlot=0 otteniamo:
name = "CPURBX"
iwow = 1
board_code_l = 0x13
board_code_h = 0x0
```

Struttura STRU_INT

Questa keyword permette di definire una variabile di tipo STRU_INT (STRUCT_P).

Viene utilizzata dalle istruzioni di gestione interrupt int_timer(), int_after(), int_inp().

È composta dalle seguenti informazioni:

© 2025 Robox SpA

Campo	Tipo	Descrizione	Attributi
cntr	I32	BIT 0: Se impostato a 0 si ha l'esecuzione continua dell'interrupt. Se impostato a 1 si ha l'esecuzione one-shot BIT 1: Se impostato a 0 usa STRU_INT.count come contatore di interrupt. Se impostato a 1 usa STRU_INT.count come flag mettendolo a 1 ogni volta che si verifica un interrupt. È a cura dell'utente azzerare STRU_INT.count	Read/write
count	I32	Contatore di interrupt	Read/write
intTime	REAL	Tempo [sec] relativo al verificarsi dell'interrupt. Questo tempo va confrontato con il parametro di sistema synch_hw_time [microsec] che corrisponde al tempo dell'ultimo synch di sistema. Attenzione: intTime [sec] mentre synch_hw_time [microsec]. Per calcolare a che distanza temporale si è verificato l'interrupt rispetto all'ultimo synch di sistema usare la formula: delta_sec = intTime - (synch_hw_time * 1e-6). Tale formula va usata all'interno della rule chiamata dall'interrupt, in tal caso varrà sempre: 0 < delta_sec < si	Read
maskPhoto	I32	Maschera degli assi dei quali si vuole la foto sul verificarsi dell'interrupt	Read/write

© 2025 Robox SpA

Campo	Tipo	Descrizione	Attributi
nrule	I32	Numero di rule da attivare. (Essa deve essere contenuta nel file $ rule_time e/o nel file $ rule_inp	Read/write
pos[1-32]	REAL	Posizione fotografata degli assi (abilitati da maskPhoto) su interrupt	Read
sts	I32	1 significa che la struttura dati è utilizzata dalle istruzioni int_timer , int_inp , int_after 2 significa che la struttura dati è utilizzata dalla istruzione Rule_periodic	Read

Esempi

Esempio di definizione:

```
; definizione di variabile box di tipo STRU_INT
STRU_INT box
```

Esempio di utilizzo:

```
; utilizzo dei dati contenuti nella struttura
; definizione di variabile box di tipo STRU_INT
STRU_INT box
; scrivo i dati della struttura
box.cntr.0 = 0                          ; esecuzione continua
box.cntr.1 = 0                          ; il totalizzatore è abilita
box.maskPhoto = 0x3                     ; abilito le foto dei primi
box.nrule = 100                         ; attivo la rule numero 100
box.sts = 1                             ; indico che la struttura è
```

Struttura STRU_LARGE_INT

Questa keyword permette di definire una variabile di tipo STRU_LARGE_INT. Questa struttura è da usare come sostituto del tipo i64 non presente nel linguaggio R3.

Può essere utilizzata nelle istruzioni get_i64() 719 e set_i64() 770.

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
high	I32	Parte alta del i64

© 2025 Robox SpA

Campo	Tipo	Descrizione
low	U32	Parte bassa del i64

Esempi

Esempio di utilizzo:

```
U8 buffer[128]
I32 val_high
U32 val_low
STRU_LARGE_INT set_val
STRU_LARGE_INT my_data

; -------- loop begin --------
__main_loop__

    ; range check
    if (offset >= 0 and (offset + sizeof(my_data)) < 128)
        set_val.high = val_high
        set_val.low  = val_low
        set_i64(buffer, offset, set_val)
        get_i64(buffer, offset, my_data)
    endif

end_main_loop
```

Struttura STRU_LARGE_UINT

Questa keyword permette di definire una variabile di tipo STRU_LARGE_UINT. Questa struttura è da usare come sostituto del tipo u64 non presente nel linguaggio R3.

Può essere utilizzata nelle istruzioni get_u64() e set_u64().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
high	U32	Parte alta del u64
low	U32	Parte bassa del u64

Esempi

Esempio di utilizzo:

```
U8 buffer[128]
U32 val_high
U32 val_low
STRU_LARGE_UINT set_val
STRU_LARGE_UINT my_data

; -------- loop begin --------
```

© 2025 Robox SpA

```
__main_loop__

    ; range check
    if (offset >= 0 and (offset + sizeof(my_data)) < 128)
        set_val.high = val_high
        set_val.low  = val_low
        set_u64(buffer, offset, set_val)
        get_u64(buffer, offset, my_data)
    endif

end_main_loop
```

Struttura STRU_MASTERF

Validità: **RC3E v33.5.8-7** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_MASTERF (STRUCT_P).

Viene utilizzata dall'istruzione master_filter().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
velFilterTime	REAL	costante di tempo filtraggio velocità limitata a essere >= SI
errRecoveryTime		costante di tempo di recupero errore limitata a essere >= SI
filteredVel		velocità filtrata (storage della funzione, da inizializzare tipicamente a 0)
posErr		errore di posizione (storage della funzione, da inizializzare tipicamente a 0)
masterPos		posizione reale del master (da impostare solo se NAX = 0)
mastervel		velocità reale del master (da impostare solo se NAX = 0)
pos		posizione rigenerata (Deve essere inizializzato a posizione attuale. Successivamente sola lettura)
vel		velocità rigenerata (Deve essere inizializzato a 0. Successivamente sola lettura)
acc		(sola lettura)

© 2025 Robox SpA

Esempi

Esempio di utilizzo:

```
; definizione di variabile sega1 di tipo STRU_FOLLOW2
STRU_MASTERF filMas
result = master_filter(filMas, 3)
velocitaFiltrata = filMas.vel
```

Struttura STRU_MATCAM

Questa keyword permette di definire una variabile di tipo STRU_MATCAM (STRUCT_P).

Viene utilizzata dall'istruzione mv_cam(), che permette di eseguire una polinomiale del terzo ordine.

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
xInit	REAL	x inizio camma
yInit	REAL	y inizio camma
c	REAL	coefficiente c
b	REAL	coefficiente b
a	REAL	coefficiente a

Esempi

Esempio di dichiarazione:

```
; definizione di un array muoviCamma di tipo STRU_MATCAM
STRU_MATCAM muoviCamma[3]
```

Esempio

```
; utilizzo della struttura per un movimento a camma
; definizione di un array muoviCamma di tipo STRU_MATCAM
STRU_MATCAM muoviCamma[3]
; definizione di variabile accessoria per la mv_cam
I32 numero_tratti = 3
; inserimento dei dati nella struttura
; (deve essere un array con un numero
; di punti uguale a numero_tratti)
if (not(r(100)))
    muoviCamma[0].xInit = r(220)
    muoviCamma[0].yInit = r(221)
    muoviCamma[0].c = r(222)
    muoviCamma[0].b = r(223)
    muoviCamma[0].a = r(224)
```

© 2025 Robox SpA

```
    muoviCamma[1].xInit = r(230)
    muoviCamma[1].yInit = r(231)
    muoviCamma[1].c = r(232)
    muoviCamma[1].b = r(233)
    muoviCamma[1].a = r(234)
    muoviCamma[2].xInit = r(240)
    muoviCamma[2].yInit = r(241)
    muoviCamma[2].c = r(242)
    muoviCamma[2].b = r(243)
    muoviCamma[2].a = r(244)
    r(100) = 1
endif
...
; esecuzione della camma
ip(2) = mv_cam(ip(1), numero_tratti, muoviCamma)
```

Struttura STRU_MODBUS

Questa keyword permette di definire una variabile di tipo STRU_MODBUS.

Viene utilizzata dalle istruzioni di gestione modbus 138.

E' composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
status	INT	va imposto = 0 per inizializzazione in uscita contiene codice errore o 1 se in attesa di fine compimento operazione (come il risultato delle funzioni che utilizzano questa struttura)
address	INT	indice del primo indirizzo da leggere/scrivere
quantity	INT	numero di indirizzo da leggere/scrivere
endian	INT	tipo di endian da usare: • EM_B_16 = 0 Big Endian nella singola word (default) • EM_L_16 = 1 Little Endian nella singola word • EM_B_32 = 2 Big Endian in 32 bit • EM_L_32 = 3 Little Endian in 32 bit • EM_MB_32 = 4 Middle Big Endian in 32 bit

© 2025 Robox SpA

Campo	Tipo	Descrizione
		• EM_ML_32 = 5 Middle little Endian in 32 bit • EM_B_64 = 6 Big Endian in 64 bit • EM_L_64 = 7 Little Endian in 64 bit • EM_MB_64 = 8 Middle Big Endian in 64 bit • EM_ML_64 = 9 Middle little Endian in 64 bit
res_1	INT	trans_id
unitId	INT	unit_id
res_3	REAL	Time_stamp

Esempi

Esempio di utilizzo:

```
; definizione di variabile box di tipo STRU_INT
STRU_MODBUS myStru
INT valueRead[10]
INT idSlave = 0
INT return
...
; Preparo la struttura di appoggio per eseguire il comando. Questo ser
; Voglio leggere gli holding register dall'indice 1 all'indice 10 e in
myStru.status = 0
myStru.address = 1
myStru.quantity = 10
myStru.unitId = idSlave
myStru.endian = EM_B_32
...
; chiamo la funzione per leggere gli holding register (mbc è l'handle
return = mbc_read_hold_reg_s(mbc, myStru, valueRead)
; Nota: Il risultato della chiamata a funzione e' disponibile anche ne
```

Struttura STRU_MVGENITP

Questa keyword permette di definire una variabile di tipo STRU_MVGENITP (STRUCT_P 918).

Viene utilizzata dall'istruzione di movimento mva_to_n_v 504().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
nAx	I32	Numero dell'asse

© 2025 Robox SpA

Campo	Tipo	Descrizione
target	REAL	Target da raggiungere (posizione). Se il target è distante meno di pos_thr / 10, RTE considera lo spazio nullo. Quindi nel caso l'utente desideri effettuare dei micro avanzamenti (<POS_THR/10) dovrà impostare opportunamente la variabile POS_THR
v	REAL	Velocità di regime per portarsi al target. Nota: In caso di interpolazione di più assi, la velocità utilizzata potrebbe essere minore di quella impostata (solo l'asse più lento a raggiungere il target la utilizza, gli altri useranno una velocità più bassa).
vf	REAL	Velocità finale sul target.
accI	REAL	Accelerazione/decelerazione massima iniziale per portarsi a v
accF	REAL	Accelerazione/decelerazione massima finale per portarsi sul target (sul target avrò accelerazione nulla)
accH	REAL	Accelerazione in caso di hold

Esempi

Esempio di dichiarazione a un asse:

```
; definizione di variabile mov1 di tipo STRU_MVGENITP per la movimenta
STRU_MVGENITP mov1
```

Esempio di dichiarazione a tre assi:

```
; definizione di variabile mov_ax di tipo STRU_MVGENITP per la movimen
STRU_MVGENITP mov_ax[3]
```

Esempio di utilizzo:

```
; utilizzo della struttura per una movimentazione
; definizione di variabile m1 di tipo STRU_MVGENITP per la movimentazi
STRU_MVGENITP m1
; definizione di variabili accessorie
I32 esito
```

© 2025 Robox SpA

```
I32 restartFlag
; inserimento dei dati nella struttura
m1.nAx = 1                                  ;numero asse
m1.target = rr(1000)                        ;posizione desiderata [units
m1.v = rr(1001)                             ;velocità di regime [units/s
m1.vf = rr(1002)                            ;Velocità finale [units/sec]
m1.acci = rr(1003)                          ;accelerazione iniziale [uni
m1.accf = rr(1004)                          ;accelerazione finale [units
m1.accH = max_acc(1)                        ;valore della decelerazione
; flag di restart
restartFlag = false
; eseguo la movimentazione
esito = mva_to_n_v(restartFlag, m1 , 1, hold, fr)
; controllo l'esito
if (esito = M_REACHED)
    sr(1) = "HO FINITO!"
endif
```

NOTA: per l'accesso ricordarsi che gli indici partono da 0.

Struttura STRU_MVITP

Questa keyword permette di definire una variabile di tipo STRU_MVITP (STRUCT_P).

Viene utilizzata dall'istruzione di movimento assi mva_to_n().

E' composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
nAx	I32	Numero dell'asse
target	REAL	Posizione da raggiungere [units]
v	REAL	Velocità di regime [units/sec]
acc	REAL	Accelerazione [units/sec²]
dec	REAL	Decelerazione [units/sec²]
decH	REAL	Decelerazione in caso di hold [units/sec²]

Esempi

Esempio di definizione a un asse:

```
; definizione variabile muoviTrapezioidale di tipo STRU_MVITP (coinvol
STRU_MVITP muoviTrapezioidale
```

Esempio di definizione a tre assi:

© 2025 Robox SpA

```
; definizione variabile muoviTrapezioidale di tipo STRU_MVITP (coinvol
STRU_MVITP muoviTrapezioidale[3]
```

Esempio di definizione a trentadue assi:

```
; definizione variabile muoviTrapezioidale di tipo STRU_MVITP (coinvol
STRU_MVITP muoviTrapezioidale[32]
```

Esempio di utilizzo:

```
; utilizzo della struttura per una movimentazione
; definizione variabile m1 di tipo STRU_MVITP (coinvolge un singolo as
STRU_MVITP m1
; definizione di variabili accessorie
I32 esito
I32 restartFlag
; inserimento dei dati nella struttura
m1.nAx = 1                                  ;numero asse
m1.target = rr(1000)                        ;posizione desiderata [units
m1.v = rr(1001)                             ;velocità di regime [units/s
m1.acc = rr(1002)                           ;accelerazione [units/sec^2]
m1.dec = rr(1003)                           ;decelerazione [units/sec^2]
m1.decH = max_acc(1)                        ;valore della decelerazione
; flag di restart
restartFlag = false
; eseguo la movimentazione
esito = mva_to_n(restartFlag, m1 , 1, hold, fr)
; controllo l'esito
if (esito = M_REACHED)
    sr(1) = "HO FINITO!"
endif
```

NOTA: Per l'accesso ricordarsi che gli indici partono da 0.

Struttura STRU_MVITPCJ

Questa keyword permette di definire una variabile di tipo STRU_MVITPCJ (STRUCT_P).

Viene utilizzata dall'istruzione di movimento mva_to_n_cj().

E' composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
nAx	I32	Numero dell'asse
target	REAL	Posizione da raggiungere [units]
v	REAL	Velocità di regime [units/sec]
acc	REAL	Accelerazione/decelerazione [units/sec²]
jerk	REAL	Jerk [units/sec³]

© 2025 Robox SpA

Campo	Tipo	Descrizione
decH	REAL	Decelerazione in caso di hold [units/sec²]

Esempi

Esempio di dichiarazione a un asse:

```
; definizione di variabile muoviJerkControllato di tipo STRU_MVITPCJ
STRU_MVITPCJ muoviJerkControllato
```

Esempio di dichiarazione a tre assi:

```
; definizione di un array di tre variabili di tipo STRU_MVITPCJ (tre a
STRU_MVITPCJ mov[3]
```

Esempio di accesso al membro v:

```
; si accede al membro v della prima variabile struttura
mov[0].v
```

Esempio di utilizzo:

```
; utilizzo della struttura per una movimentazione
; definizione di variabile m1 di tipo STRU_MVITPCJ
STRU_MVITPCJ m1
; definizione di variabili accessorie
I32 esito
I32 restartFlag
; inserimento dei dati nella struttura
m1.nAx = 1                                  ;numero asse
m1.target = rr(1000)                        ;posizione desiderata [units
m1.v = rr(1001)                             ;velocità di regime [units/s
m1.acc = rr(1002)                           ;accelerazione/decelerazione
m1.jerk = rr(1003)                          ;jerk [units/sec^3]
m1.decH = max_acc(1)                        ;valore della decelerazione
; flag di restart
restartFlag = false
; eseguo la movimentazione
esito = mva_to_n_cj(restartFlag, m1 , 1, hold, fr)
; controllo l'esito
if (esito = M_REACHED)
    sr(1) = "HO FINITO!"
endif
```

NOTA: Per l'accesso ricordarsi che gli indici partono da 0.

Struttura STRU_MVTO

Questa keyword permette di definire una variabile di tipo STRU_MVTO (STRUCT_P).

Viene utilizzata dall'istruzione di movimento mv_to().

E' composta dalle seguenti informazioni:

© 2025 Robox SpA

Campo	Tipo	Descrizione
cPos	REAL	Posizione attuale [units]. Da aggiornare la prima volta che la funzione viene chiamata. Nel caso venga pilotata la posizione di un asse, l'assegnazione da effettuare è: stru_mvto.cPos = IP()
cSpeed	REAL	velocità attuale [units/sec]. Da aggiornare la prima volta che viene chiamata la funzione. Nel caso venga pilotata la posizione di un asse IP, la relativa IV (velocità) viene automaticamente calcolata da RTE. L'assegnazione da eseguire risulta quindi: stru_mvto.cspeed = IV() Nel caso venga pilotato il valore di un registro o di una variabile (quindi non la IP), il calcolo della cSpeed (velocità attuale) deve essere effettuato dall'utente. Il sistema più semplice è calcolare la velocità di variazione della grandezza pilotata come segue: in rule_init 757 oldValue = rr(1000) in rule_prologue 759 velGrandezza = (rr(1000)-oldValue) / si oldValue = rr(1000) ; nelle rules 748 viene manipolato il valore della grandezza pilotata ; prima della chiamata alla MV_TO stru_mvto.cpos = rr(1000) stru_mvto.cSpeed = velGrandezza .... rr(1000)= MV_TO(.....)
dPos	REAL	Posizione desiderata [units]

© 2025 Robox SpA

Campo	Tipo	Descrizione
dSpeed	REAL	Velocità desiderata [units/sec]
acc	REAL	Accelerazione [units/sec²]
dec	REAL	Decelerazione [units/sec²]
delta	REAL	Incremento temporale dall'ultima esecuzione della funzione. Se usata nelle rule allora corrisponde al valore di si Non è obbligatorio che il valore sia riferito ad un tempo. Nulla vieta che sia ad esempio uno spazio di un asse master.

Esempi

Esempio di definizione:

```
; definizione di variabile muoviTrap di tipo STRU_MVTO
STRU_MVTO muoviTrap
```

Esempio:

```
; utilizzo della struttura per una movimentazione
; definizione di variabile m1 di tipo STRU_MVTO
STRU_MVTO m1
; definizione di variabili accessorie
I32 esito
; inserimento dei dati nella struttura
m1.cPos = ip(1)                          ;posizione attuale [units]
m1.cSpeed = iv(1)                        ;velocità attuale [units/sec
m1.dPos = rr(1002)                       ;posizione desiderata [units
m1.dSpeed = rr(1003)                     ;velocità desiderata [units/
m1.acc = rr(1004)                        ;accelerazione [units/sec^2]
m1.dec = rr(1005)                        ;decelerazione [units/sec^2]
m1.delta = si                            ;incremento temporale
; eseguo la movimentazione
ip(1) = mv_to(esito, m1 , 0)
; controllo l'esito
if (esito = M_REACHED)
    sr(1) = "HO FINITO!"
endif
```

NOTA: Per l'accesso ricordarsi che gli indici partono da 0.

© 2025 Robox SpA

Struttura STRU_MVTOCJ

Questa keyword permette di definire una variabile di tipo STRU_MVTOCJ (STRUCT_P 918).

Viene utilizzata dall'istruzione di movimento mv_to_cj 486().

E' composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
cPos	REAL	Posizione attuale [units]. Da aggiornare la prima volta che la funzione viene chiamata. Nel caso venga pilotata la posizione di un asse, l'assegnazione da effettuare è: stru_mvto.cPos = IP()
cSpeed	REAL	velocità attuale [units/sec]. Da aggiornare la prima volta che viene chiamata la funzione. Nel caso venga pilotata la posizione di un asse IP, la relativa IV (velocità) viene automaticamente calcolata da RTE. L'assegnazione da eseguire risulta quindi: stru_mvto.cspeed = IV() Nel caso venga pilotato il valore di un registro o di una variabile (quindi non la IP), il calcolo della cSpeed (velocità attuale) deve essere effettuato dall'utente. Il sistema più semplice è calcolare la velocità di variazione della grandezza pilotata come segue: in rule_init 757 oldValue = rr(1000) in rule_prologue 759 velGrandezza = (rr(1000)-oldValue) / si oldValue = rr(1000) ; nelle rules 748 viene manipolato il valore della grandezza pilotata ; prima della chiamata alla MV_TO stru_mvto.cpos = rr(1000) stru_mvto.cSpeed = velGrandezza

© 2025 Robox SpA

Campo	Tipo	Descrizione
		.... rr(1000)= MV_TO(.....)
cAcc	REAL	Accelerazione attuale [units/sec²]. Da aggiornare la prima volta che viene chiamata la funzione. Nel caso venga pilotata la posizione di un asse IP, la relativa IA (accelerazione) viene automaticamente calcolata da RTE. L'assegnazione da eseguire risulta quindi: stru_mvto.cAcc = IA() Altrimenti vedi cSpeed!
dPos	REAL	Posizione desiderata [units]
dSpeed	REAL	Velocità desiderata [units/sec]
dAcc	REAL	Accelerazione desiderata [units/sec²]
dJerk	REAL	Jerk [units/sec³]
delta	REAL	Incremento temporale dall'ultima esecuzione della funzione. Se usata nelle rule allora corrisponde al valore di si Non è obbligatorio che il valore sia riferito ad un tempo. Nulla vieta che sia ad esempio uno spazio di un asse master.

Esempi

Esempio di dichiarazione:

```
; definizione di variabile stk di tipo STRU_MVTOCJ
STRU_MVTOCJ stk
```

Esempio di utilizzo

```
; utilizzo della struttura per una movimentazione
; definizione di variabile m1 di tipo STRU_MVTOCJ
STRU_MVTOCJ m1
; definizione di variabili accessorie
I32 esito
I32 restartFlag
; inserimento dei dati nella struttura
m1.cPos = ip(1)                          ;posizione attuale [units]
```

© 2025 Robox SpA

```
m1.cSpeed = iv(1)                              ;velocità attuale [units/sec
m1.cAcc = ia(1)                                ;accelerazione attuale [unit
m1.dPos = rr(1002)                             ;posizione desiderata [units
m1.dSpeed = rr(1003)                           ;velocità desiderata [units/
m1.dAcc = rr(1004)                             ;accelerazione desiderata [u
m1.dJerk = rr(1005)                            ;jerk [units/sec^3]
m1.delta = si                                  ;incremento temporale
; eseguo la movimentazione
ip(1) = mv_to_cj(esito, restartFlag, m1 , hold, fr)
; controllo l'esito
if (esito = M_REACHED)
    sr(1) = "HO FINITO!"
endif
```

Struttura STRU_MVTOCJV

Questa keyword permette di definire una variabile di tipo STRU_MVTOCJV (STRUCT_P).

Viene utilizzata dall'istruzione di movimento mv_to_cjv().

E' composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
cPos	REAL	Posizione attuale [units]. Da aggiornare la prima volta che la funzione viene chiamata. Nel caso venga pilotata la posizione di un asse, l'assegnazione da effettuare è: stru_mvto.cPos = IP()
cSpeed	REAL	velocità attuale [units/sec]. Da aggiornare la prima volta che viene chiamata la funzione. Nel caso venga pilotata la posizione di un asse IP, la relativa IV (velocità) viene automaticamente calcolata da RTE. L'assegnazione da eseguire risulta quindi: stru_mvto.cspeed = IV() Nel caso venga pilotato il valore di un registro o di una variabile (quindi non la IP), il calcolo della cSpeed (velocità attuale) deve essere effettuato dall'utente. Il sistema più semplice è calcolare la velocità di

© 2025 Robox SpA

Campo	Tipo	Descrizione
		variazione della grandezza pilotata come segue: in rule_init 757 oldValue = rr(1000) in rule_prologue 759 velGrandezza = (rr(1000)-oldValue) / si oldValue = rr(1000) ; nelle rules 748 viene manipolato il valore della grandezza pilotata ; prima della chiamata alla MV_TO stru_mvto.cpos = rr(1000) stru_mvto.cSpeed = velGrandezza .... rr(1000)= MV_TO(.....)
cAcc	REAL	Accelerazione attuale [units/sec²]. Da aggiornare la prima volta che viene chiamata la funzione. Nel caso venga pilotata la posizione di un asse IP, la relativa IA (accelerazione) viene automaticamente calcolata da RTE. L'assegnazione da eseguire risulta quindi: stru_mvto.cAcc = IA() Altrimenti vedi cSpeed!
dPos	REAL	Posizione desiderata [units]
dSpeed	REAL	Velocità desiderata [units/sec]
dAcc	REAL	Accelerazione [units/sec²]
dJerk	REAL	Jerk [units/sec³]
delta	REAL	Incremento temporale dall'ultima esecuzione della funzione. Se usata nelle rule allora corrisponde al valore di si Non è obbligatorio che il valore sia riferito ad un tempo. Nulla

© 2025 Robox SpA

Campo	Tipo	Descrizione
		vieta che sia ad esempio uno spazio di un asse master.

Esempi

Esempio di dichiarazione:

```
; definizione di variabile stk di tipo STRU_MVTOCJ
STRU_MVTOCJV stk
```

Esempio di utilizzo:

```
; utilizzo della struttura per una movimentazione
; definizione di variabile m1 di tipo STRU_MVTOCJ (coinvolge un singol
STRU_MVTOCJV m1
; definizione di variabili accessorie
I32 esito
; inserimento dei dati nella struttura
m1.cPos = ip(1)                              ;posizione attuale [units]
m1.cSpeed = iv(1)                            ;velocità attuale [units/sec
m1.cAcc = ia(1)                              ;accelerazione attuale [unit
m1.dPos = rr(1002)                           ;posizione desiderata [units
m1.dSpeed = rr(1003)                         ;velocità desiderata [units/
m1.dAcc = rr(1004)                           ;accelerazione desiderata [u
m1.dJerk = rr(1005)                          ;jerk [units/sec^3]
m1.delta = si                                ;incremento temporale
; eseguo la movimentazione
esito = mv_to_cjv 487(m1 , hold, fr)
; imposto i valori restituiti di cPos, cSpeed e cAcc
set_kinematics(1, m1.cPos, m1.cSpeed, m1.cAcc)
; controllo l'esito
if (esito = M_REACHED)
    sr(1) = "HO FINITO!"
endif
```

Struttura STRU_MVTOCJV_INFO

Questa keyword permette di definire una variabile di tipo STRU_MVTOCJV_INFO (STRUCT_P 918).

Viene utilizzata dall'istruzione di movimento mv_to_cjv_info 489().

E' composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
targetSpeed	REAL	Valore target di velocità [units/sec]
dAcc	REAL	Valore di accelerazione per raggiungere il target [units/sec²]

© 2025 Robox SpA

Campo	Tipo	Descrizione
dJerk	REAL	Valore del jerk per raggiungere il target[units/sec³]
cPos	REAL	Posizione attuale [units]. Da aggiornare la prima volta che la funzione viene chiamata.
cSpeed	REAL	velocità attuale [units/sec]. Da aggiornare la prima volta che viene chiamata la funzione.
cAcc	REAL	Accelerazione attuale [units/sec²]. Da aggiornare la prima volta che viene chiamata la funzione.
posInfo	REAL	Posizione raggiunta [units]
timeInfo	REAL	tempo necessario [sec]

Esempi

Esempio di dichiarazione:

```
; definizione di variabile mvinfo di tipo STRU_MVTOCJ_INFO
STRU_MVTOCJV_INFO mvinfo
```

Esempio di utilizzo:

```
; utilizzo della struttura
; definizione di variabile m1 di tipo STRU_MVTOCJ_INFO
STRU_MVTOCJV_INFO m1
; definizione di variabili accessorie
I32 esito
; inserimento dei dati nella struttura
m1.TargetSpeed = rr(1000)                ;valore target di velocità [
m1.dAcc = rr(1001)                       ;accelerazione desiderata [u
m1.dJerk = rr(1002)                      ;jerk [units/sec^3]
m1.cPos = ip(1)                          ;posizione attuale [units]
m1.cSpeed = iv(1)                        ;velocità attuale [units/sec
m1.cAcc = ia(1)                          ;accelerazione attuale [unit

; eseguo la movimentazione
esito = mv_to_cjv_info(m1)
; controllo l'esito
; se è ok, memorizzo i valori di posizione raggiunta e tempo necessari
if (esito)
    sr(1) = "HO FINITO!"
```

© 2025 Robox SpA

```
    rr(1100) = m1.PosInfo                    ;posizione raggiunta
    rr(1101) = m1.TimeInfo                   ;tempo necessario
endif
```

Struttura STRU_MVTOCJVE

Questa keyword permette di definire una variabile di tipo STRU_MVTOCJVE (STRUCT_P 918).

Viene utilizzata dall'istruzione di movimento mv_to_cjv 487().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
stopPosition	REAL	SOLA LETTURA. In caso di imposizione del campo "dSpeed" a zero, viene scritta automaticamente la posizione di arresto.
stopTime	REAL	SOLA LETTURA. In caso di imposizione del campo "dSpeed" a zero, viene scritto automaticamente il tempo relativo necessario per l'arresto.
cPos	REAL	Posizione attuale [units]. Da aggiornare la prima volta che la funzione viene chiamata. Nel caso venga pilotata la posizione di un asse, l'assegnazione da effettuare è: stru_mvto.cPos = IP()
cSpeed	REAL	velocità attuale [units/sec]. Da aggiornare la prima volta che viene chiamata la funzione. Nel caso venga pilotata la posizione di un asse IP, la relativa IV (velocità) viene automaticamente calcolata da RTE. L'assegnazione da eseguire risulta quindi: stru_mvto.cspeed = IV() Nel caso venga pilotato il valore di un registro o di una variabile (quindi non la IP), il calcolo della cSpeed (velocità attuale) deve essere effettuato dall'utente. Il sistema più semplice è calcolare la velocità di

© 2025 Robox SpA

Campo	Tipo	Descrizione
		variazione della grandezza pilotata come segue: in rule_init 757 oldValue = rr(1000) in rule_prologue 759 velGrandezza = (rr(1000)-oldValue) / si oldValue = rr(1000) ; nelle rules 748 viene manipolato il valore della grandezza pilotata ; prima della chiamata alla MV_TO stru_mvto.cpos = rr(1000) stru_mvto.cSpeed = velGrandezza .... rr(1000)= MV_TO(.....)
cAcc	REAL	Accelerazione attuale [units/sec²]. Da aggiornare la prima volta che viene chiamata la funzione. Nel caso venga pilotata la posizione di un asse IP, la relativa IA (accelerazione) viene automaticamente calcolata da RTE. L'assegnazione da eseguire risulta quindi: stru_mvto.cAcc = IA()
dPos	REAL	Posizione desiderata [units]
dSpeed	REAL	Velocità desiderata [units/sec]
dAcc	REAL	Accelerazione [units/sec²]
dJerk	REAL	Jerk [units/sec³]
delta	REAL	Incremento temporale dall'ultima esecuzione della funzione. Se usata nelle rule allora corrisponde al valore di si Non è obbligatorio che il valore sia riferito ad un tempo. Nulla vieta che sia ad esempio uno spazio di un asse master.

© 2025 Robox SpA

Esempi

Esempio di definizione

```
; definizione di variabile stk i tipo STRU_MVTOCJE
STRU_MVTOCJVE stk
```

Esempio di utilizzo:

```
; utilizzo della struttura per una movimentazione
; definizione di variabile stk i tipo STRU_MVTOCJE
STRU_MVTOCJVE m1
; definizione di variabili accessorie
I32 esito
; inserimento dei dati nella struttura
m1.cPos = ip(1)                              ;posizione attuale [units]
m1.cSpeed = iv(1)                            ;velocità attuale [units/sec
m1.cAcc = ia(1)                              ;accelerazione attuale [unit
m1.dPos = rr(1002)                           ;posizione desiderata [units
m1.dSpeed = rr(1003)                         ;velocità desiderata [units/
m1.dAcc = rr(1004)                           ;accelerazione desiderata [u
m1.dJerk = rr(1005)                          ;jerk [units/sec^3]
m1.delta = si                                ;incremento temporale
; eseguo la movimentazione
esito = mv_to_cjv 487(m1 , hold, fr)
; imposto i valori restituiti di cPos, cSpeed e cAcc
set_kinematics 595(1, m1.cPos, m1.cSpeed, m1.cAcc)
; controllo l'esito
if (esito = M_REACHED)
   sr(1) = "HO FINITO!"
else
    if (m1.dSpeed = 0)                       ; in questo caso è stat
        rr(1100) = m1.stopPosition            ; quindi memorizzo i v
        rr(1101) = m1.stopTime                ; e di m1.stopTime
    endif
endif
```

Struttura STRU_MVZC

Questa keyword permette di definire una variabile di tipo STRU_MVZC (STRUCT_P 918).

Viene utilizzata dall'istruzione mva_zc 511(), che esegue il ciclo di zero dell'asse.

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
nAx	I32	Numero dell'asse
preset	REAL	Quota di preselezione [units]

© 2025 Robox SpA

Campo	Tipo	Descrizione
velPos	REAL	Velocità di movimentazione preliminare al ciclo0 vero e proprio [units/sec] N.B. Se selezionato ciclo di zero a paracarro rappresenta l'errore epos(nAx) in unit per considerare raggiunto il paracarro
velZc	REAL	Velocità durante il ciclo0 vero e proprio (movimentazione finale) [units/sec]
acc	REAL	Accelerazione [units/sec²]
jerk	REAL	Jerk [units/sec³]
spSch	REAL	Spazio di movimentazione su movimento finale [units]
filter	REAL	Filtro temporale in secondi per filtrare i rimbalzi del micro. (il micro deve rimanere nello stesso stato per il tempo specificato affinchè sia assunto valido) N.B. Se è selezionato ciclo di zero a paracarro e si dispone di drive in CANOPEN/EtherCAT presente nella lista di selezione trasduttore del configuratore di progetto, rappresenta il valore di riduzione di coppia in percentuale [0-100]
zcFlag.0 [bit 0]	U32	Segno velocità: - con movimentazione che non prevede micro (zcFlag.3=1) o monodirezionale (zcFlag.4=1): 0 --> velocità negativa 1 --> velocità positiva - con movimentazione che utilizza micro (zcFlag.3=0): 0 --> velocità negativa se stato del micro in 1, positiva se stato del micro in 0 1 --> velocità positiva se stato del micro in 1, negativa se stato del micro in 0

© 2025 Robox SpA

Campo	Tipo	Descrizione
zcFlag.1 [bit 1]	U32	Selezione fronte di commutazione del micro - con movimentazioni che utilizzano solo micro (zcFlag.3=0 zcFlag.2=0) il fronte del micro è il punto di assegnazione della Quota di preselezione (Preset) 0 --> si seleziona il fronte di discesa del micro 1 --> si seleziona il fronte di salita del micro - con movimentazioni che utilizzano micro + full-precision (zcFlag.3=0 zcFlag.2=1) il punto di assegnazione della Quota di preselezione (Preset) è quello identificato dalla full-precision col micro nello stato indicato da questo bit: 0 --> si seleziona lo stato logico zero del micro 1 --> si seleziona lo stato logico uno del micro - non viene considerato con movimentazioni che non utilizzano micro (zcFlag.3=1)
zcFlag.2 [bit 2]	U32	Abilitazione/disabilitazione full-precision. 0 --> disabilita full precision 1 --> abilita full precision N.B. La full-precision è la massima precisione possibile ottenibile dal trasduttore dell'asse e la possibilità di utilizzo dipende dal tipo di trasduttore stesso. Esempi: - il CH0 se trasduttore di tipo enc_inc - la quota di giro elettrico del motore se trasduttore di tipo Fieldbus (se il dispositivo è gestito e se è disponibile)
zcFlag.3 [bit 3]	U32	Selezione modalità di utilizzo del micro 0 --> abilita utilizzo del micro

© 2025 Robox SpA

Campo	Tipo	Descrizione
		1 --> disabilita utilizzo del micro (N.B: deve essere abilitata la full-precision (zcFlag.2=1))
zcFlag.4 [bit 4]	U32	Selezione della possibilità di invertire la direzione durante la movimentazione: 0 --> l'esecuzione della movimentazione è bidirezionale 1 --> l'esecuzione della movimentazione è monodirezionale
zcFlag.5 [bit 5]	U32	Selezione della possibilità di limitare l'escursione della movimentazione preliminare allo spazio equivalente alla differenza dei due paracarri massimo e minimo dell'asse (max_str e min_str): 0 --> lo spazio percorribile durante la movimentazione preliminare è illimitato 1 --> lo spazio percorribile durante la movimentazione preliminare è limitato dal valore max_str(asse) - min_str(asse). Nel caso venga superato questo valore verrà segnalata diagnostica (#1)
zcFlag.6 [bit 6]	U32	- se movimentazione bidirezionale (zcFlag.4=0) o micro non usato (zcFlag.3=1) viene ignorato - se movimentazione monodirezionale (zcFlag.4=1) e micro usato (zcFlag.3=0) Abilitazione/disabilitazione delle movimentazioni preliminari (ricerca variazione micro) (#1) 0 --> abilitato 1 --> disabilitato
zcFlag.7 [bit 7]	U32	Selezione del ciclo0 a paracarro (#1) 0 --> disabilitato 1 --> abilitato

© 2025 Robox SpA

Campo	Tipo	Descrizione
zcFlag.8 [bit 8]	U32	Selezione del ciclo di zero ON_POSITION. La posizione attuale dell'asse diventa il valore di preset indicato. RTE provvede anche ad alzare il relativo bit nella variabile predefinita c0_done
zcFlag.9 [bit 9]	U32	Flag abilitazione al riposizionamento sulla quota di preset. La velocità utilizzata è quella indicata nel parametro velZc. Questo bit non può essere impostato a 1 se anche il bit 10 è impostato a 1. Se i bit 9 e 10 sono entrambi a 1 verrà generato l'allarme 9900. N.B. In caso di selezione cicli di zero monodirezionali questo flag viene ignorato
zcFlag.10 [bit 10]	U32	Flag di abilitazione riposizionamento all'interno dei paracarri software (min_str e max_str). Nel caso la quota di preset sia all'esterno dei fine corsa software e zcFlag.10 = 1 RTE provvede a movimentare l'asse verso il finecorsa più vicino. L'asse viene arrestato (con rampa) non appena RTE rileva che la sua quota è all'interno dei paracarri. La velocità utilizzata è quella indicata nel parametro velPos. Questo bit non può essere impostato a 1 se anche il bit 9 è impostato a 1. Se i bit 9 e 10 sono entrambi a 1 verrà generato l'allarme 9900
zcFlag.11 [bit 11]	U32	Il bit è utilizzato solo se si è selezionato il ciclo di 0 di tipo mono-direzionale, con il micro di zero e con la ricerca della variazione iniziale. In questo caso se il bit è in 1, dopo che si trova la variazione del micro di zero la velocità di posizionamento "velPos" viene mantenuta per uno spazio pari al parametro "spSch". Indi si passa alla velocità "velZc" per effettuare

© 2025 Robox SpA

Campo	Tipo	Descrizione
		la fase finale. In pratica di allunga la fase di posizionamento iniziale (dopo che si è trovato il micro di zero) di "spSch"
zcFlag.12 [bit 12]	U32	Si utilizza solo per ciclo di 0 a paracarro. Abilita, nel momento dello schianto l'arresto dell'evoluzione della quota ideale in modo controllato e ritorno sempre controllato della ip sulla posizione dello schianto
zcFlag.13 [bit 13]	U32	Da RTE 34.24.10 Si utilizza solo per ciclo di 0 a paracarro. Disabilita il controllo sulla velocità (abs(ip-cp) > zc.velPos) durante la ricerca del paracarro
zcFlag.14 [bit 14]	U32	Da RTE 34.28.0 Si utilizza per riallineare RAWIP e RAWCP. Viene utilizzato solo nell'homing di tipo 'On Position' con drive digitali pilotati in posizione per riallineare l'offset utilizzato per il calcolo della RAWIP. Il ciclo di homing deve essere eseguito in 'missing power'. Detto in altre parole riallinea in 'missing power' (con anello di posizione aperto) la RAWIP con la RAWCP

(#1) Disponibile a partire dalle versioni RTE v32.x.x per processori INTEL, v33.3.3 per processori PPC

Legenda:

- Ciclo0 o ciclo di zero: viene definito Ciclo0 la missione finale (ultima fase) delle varie movimentazioni eventualmente effettuate dalla mva_zc, ovvero quella in cui si assegna la quota di preselezione alla cp dell'asse
- Movimentazione preliminare al ciclo0: viene definita in questo modo tutta quella serie di movimentazioni (fasi preliminari) eventualmente effettuate dalla mva_zc che precedono il Ciclo0

Esempi

Esempio di dichiarazione per un asse:

```
; definizione di variabile home di tipo STRU_MVZC per la movimentazion
```

© 2025 Robox SpA

```
STRU_MVZC home
```

Esempio di dichiarazione per tre assi:

```
; definizione di variabile home di tipo STRU_MVZC per la movimentazion
STRU_MVZC home[3]
```

Struttura STRU_PHADJ

Questa keyword permette di definire una variabile di tipo STRU_PHADJ (STRUCT_P 918).

Viene utilizzata dall'istruzione di movimento mv_phase_adj 474().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
error	REAL	È lo scostamento dell'asse slave rispetto alla posizione desiderata di fase (positivo se occorre accelerare per raggiungere la fase, negativo se occorre rallentare). Una volta che la funzione sta lavorando, cambiando il valore di error e azzerando anche lo spazio percorso dal master deltaMaster al volo (ad esempio sul fronte di salita di un input) si otterrà l'esecuzione di una nuova correzione.
masterCycle	REAL	È lo spazio dell'asse master disponibile per effettuare la correzione di fase.
velMinSlave	REAL	È il limite inferiore di velocità ammessa per il movimento dell'asse slave. Nel caso la formula di correzione richiedesse di scendere al di sotto della velocità minima, il valore reso dalla funzione sarebbe pari a velMinSlave. In caso di intervento della limitazione, la correzione dell'errore di fase risulterà solo parziale. In caso di valore negativo viene limitato a 0
velMaxSlave	REAL	È il limite superiore di velocità ammessa per il movimento dell'asse slave. Nel caso la formula di correzione richiedesse di salire al di

© 2025 Robox SpA

Campo	Tipo	Descrizione
		sopra della velocità massima, il valore reso dalla funzione sarebbe pari a velMaxSlave. In caso di intervento della limitazione, la correzione dell'errore di fase risulterà solo parziale.
accMaxSlave	REAL	È il valore di accelerazione massima accettato per l'asse slave. La quantità di errore correggibile viene automaticamente ridotta in modo da non superare il valore di accelerazione impostato. Il parametro error fornito dall'utente viene automaticamente modificato a cura della funzione.
vRef	REAL	È la velocità base dell'asse slave, cioè la velocità da tenere in assenza di correzioni. Questo parametro deve essere aggiornato a cura del software applicativo prima di ogni chiamata alla funzione.
deltaMaster	REAL	È lo spazio master trascorso dall'inizio della chiamata della funzione. La correzione avviene nell'intervallo di deltaMaster compreso fra zero e masterCycle. Per valori di deltaMaster superiori a masterCycle la funzione restituisce la velocità base vRef e codice di reso M_SYNCHRONIZED Questo parametro deve essere aggiornato a cura del software applicativo prima di ogni chiamata alla funzione.

Esempi

Esempio di definizione:

```
; definizione di variabile nastro1 di tipo STRU_PHADJ
STRU_PHADJ nastro1
```

Esempio di utilizzo:

© 2025 Robox SpA

```
; utilizzo della struttura per una movimentazione
; definizione di variabile m1 di tipo STRU_PHADJ
STRU_PHADJ m1
; definizione di variabili accessorie
I32 esito
I32 fase
I32 time
I32 posMasterIn
I32 VelAxSlave
; inserimento dei dati nella struttura
; prima esecuzione
if (first_time[346]())
    m1.masterCycle = rr(1001)                          ;spazio asse m
    m1.velMinSlave = rr(1002)                          ;velocità mini
    m1.velMaxSlave = rr(1003)                          ;velocità mass
    m1.accMaxSlave = rr(1004)                          ;accelerazione
endif

; eseguo le operazioni in base alla variabile "fase"
select (fase)
    case 0                                             ;fase iniziale (0)
        m1.error = rr(1000)                            ;scostamento asse
        posMasterIn = ip(1)                            ;memorizzo la posi
        fase = 1                                       ;passo alla fase s
    case 1                                             ;fase 1
        m1.vRef = iv(1)                                ;velocità base sla
                                                       ;in questo caso vi
                                                       ;alla velocità del
        m1.deltaMaster = ip(1) - posMasterIn           ;delta master [uni
                                                       ;viene posta ugual
                                                       ;la posizione attu
        VelAxSlave = mv_phase_adj[474](esito, m1)        ;eseguo la funzi
        iv(2) = VelAxSlave                             ;assegno all'asse
        if (esito = M_SYNCHRONIZED)                    ;se sono sincroniz
            fase = 2
            time = TFB                                 ;memorizzo l'istan
        endif
        break
    case 2                                             ;fase 2
        _if ((TFB - time) > 1)                         ;aspetto un second
            fase = 0                                   ;poi ritorno alla
        break

end_select
```

© 2025 Robox SpA

Struttura STRU_PHADJ2

Validità: **RC3E v33.5.8-7** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_PHADJ2 (STRUCT_P 918).

Viene utilizzata dall'istruzione di movimento mv_phase_adj2 474().

È composta dalle seguenti informazioni:

<table>
<tr><th>Campo</th><th>Tipo</th><th>Descrizione</th></tr>
<tr><td>corrType</td><td>I32</td><td>Tipo di profilo di correzione :
Tabella di comparazione dei profili di correzione prendendo come riferimento il profilo triangolare:
<table>
<tr><th></th><th>corrType</th><th>Vmax</th><th>Amax</th><th>Jerk max</th></tr>
<tr><td>0</td><td>Triangolare</td><td>100</td><td>100</td><td>infinito in 3 punti</td></tr>
<tr><td>1</td><td>Parabolica</td><td>75</td><td>150</td><td>infinito in 2 punti</td></tr>
<tr><td>2</td><td>Sinusoidale</td><td>78.5</td><td>123.25</td><td>infinito in 2 punti</td></tr>
<tr><td>3</td><td>Jerkata</td><td>100</td><td>200</td><td>100</td></tr>
<tr><td>4</td><td>Cosenorializzato</td><td>100</td><td>157</td><td>123</td></tr>
<tr><td>5</td><td>Pentica</td><td>100</td><td>150</td><td>150</td></tr>
</table></td></tr>
<tr><td>error</td><td>REAL</td><td>È lo scostamento dell'asse slave rispetto alla posizione desiderata di fase (positivo se occorre accelerare per raggiungere la fase, negativo se occorre rallentare).
Una volta che la funzione sta lavorando, cambiando il valore di error e azzerando anche lo spazio percorso dal master deltaMaster al volo (ad esempio sul fronte di salita di un input) si otterrà l'esecuzione di una nuova correzione.</td></tr>
<tr><td>masterCycle</td><td>REAL</td><td>È lo spazio dell'asse master disponibile per effettuare la correzione di fase.</td></tr>
</table>

© 2025 Robox SpA

Campo	Tipo	Descrizione
velMinSlave	REAL	È il limite inferiore di velocità ammessa per il movimento dell'asse slave. Nel caso la formula di correzione richiedesse di scendere al di sotto della velocità minima, il valore reso dalla funzione sarebbe pari a velMinSlave. In caso di intervento della limitazione, la correzione dell'errore di fase risulterà solo parziale e la variabile di stato assumerà valore M_SPEED_LIMITED In caso di valore negativo viene limitato a 0
velMaxSlave	REAL	È il limite superiore di velocità ammessa per il movimento dell'asse slave. Nel caso la formula di correzione richiedesse di salire al di sopra della velocità massima, il valore reso dalla funzione sarebbe pari a velMaxSlave. In caso di intervento della limitazione, la correzione dell'errore di fase risulterà solo parziale e la variabile di stato assumerà valore M_SPEED_LIMITED
accMaxSlave	REAL	È il valore di accelerazione massima accettato per l'asse slave. In caso di intervento della limitazione, la correzione dell'errore di fase risulterà solo parziale e la variabile di stato assumerà valore M_ACCEL_LIMITED
vRef	REAL	È la velocità base dell'asse slave, cioè la velocità da tenere in assenza di correzioni. Questo parametro deve essere aggiornato a cura del software applicativo prima di ogni chiamata alla funzione.
deltaMaster	REAL	È lo spazio master trascorso dall'inizio della chiamata della funzione. La correzione avviene nell'intervallo di deltaMaster compreso fra zero e masterCycle.

© 2025 Robox SpA

Campo	Tipo	Descrizione
		Per valori di deltaMaster superiori a masterCycle la funzione restituisce la velocità base vRef e codice di reso M_SYNCHRONIZED Questo parametro deve essere aggiornato a cura del software applicativo prima di ogni chiamata alla funzione.
vRefMaster	REAL	Velocità target del master (se non specificata lasciare 0 o IV(master))
vSlave	REAL	Velocità attuale slave Questo parametro deve essere aggiornato a cura del software applicativo prima di ogni chiamata alla funzione.
si	REAL	Intervallo di campionamento
kTrap	REAL	Rapporto base minore/base maggiore trapezio (eventualmente ricalcolato a inizio correzione ed azzerato a fine correzione)
camValue	REAL	Valore camma di correzione attuale (solo lettura). Varia tra 0 ed 1

Esempi

Esempio di dichiarazione:

```
; definizione di variabile nastro1 di tipo STRU_PHADJ
STRU_PHADJ2 nastro1
```

Esempio di utilizzo in rule:

```
RULE R_AUTO_2
axes(AX_2)

MOTION
    if (first_time346())
        mioAggancio.corrType = 0 ; triangolare
        mioAggancio.masterCycle = rrPassoMaster         ; spazio master
        mioAggancio.velMinSlave = 0                     ; velocità mini
        mioAggancio.velMaxSlave = MAX_SPE(AX_2)         ; velocità mass
        mioAggancio.accMaxSlave = MAX_ACC(AX_2)         ; accelerazione
        mioAggancio.si = SI
        mioAggancio.error = 0
        mv_status = 0
    endif
    if (rise562(inp(1))) ; pulsantino rosso ADV
        mioAggancio.vRefMaster = rrVelMaster ; velocità a regime del m
```

© 2025 Robox SpA

```
        mioAggancio.error = slave_error
        mv_status = M_INITIALIZING
        posMasterIniziale = IP(AX_1)
    endif
    ;
    ; velocità che deve tenere l'asse slave in assenza di correzioni
    ;
    mioAggancio.vRef = iv(AX_1)
    ;
    ; velocità attuale dello slave
    ;
    mioAggancio.vSlave = iv(AX_2)
    ;
    ; spazio percorso dal master a partire dall'inizio della correzion
    ;
    mioAggancio.deltaMaster = ip(AX_1) - posMasterIniziale
    ;
    ; Muovo l'asse
    ;
    iv(AX_2) = mv_phase_adj2(mv_status, mioAggancio)

    if(rise(mv_status = M_SYNCHRONIZED))
        ; correzione terminata
    endif

END_MOTION

END_RULE
```

Struttura STRU_PM_PD

Validità: **RC3E v33.9.1-9** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_PM_PD.

Viene utilizzata dall'istruzione di comunicazione axioline axio_pm_r().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
L1_voltage	I32	Tensione in mV sulla fase L1 (*)(***)
L2_voltage	I32	Tensione in mV sulla fase L2 (*)(***)
L3_voltage	I32	Tensione in mV sulla fase L3 (*)(***)
L1_current	I32	Corrente in mA sulla fase L1 (*)

© 2025 Robox SpA

Campo	Tipo	Descrizione
L2_current	I32	Corrente in mA sulla fase L2 (*)
L3_current	I32	Corrente in mA sulla fase L3 (*)
NeutralConductor_current	I32	Corrente in mA sul neutro (*)
L1_Real_PE	I32	Potenza/Energia attiva in W/Wh sulla fase L1 (**)
L2_Real_PE	I32	Potenza/Energia attiva in W/Wh sulla fase L2 (**)
L3_Real_PE	I32	Potenza/Energia attiva in W/Wh sulla fase L3 (**)
L1_Reactive_PE	I32	Potenza/Energia reattiva in VAr/VArh sulla fase L1 (**)
L2_Reactive_PE	I32	Potenza/Energia reattiva in VAr/VArh sulla fase L2 (**)
L3_Reactive_PE	I32	Potenza/Energia reattiva in VAr/VArh sulla fase L3 (**)
L1_Apparent_PE	I32	Potenza/Energia apparente in VA/VAh sulla fase L1 (**)
L2_Apparent_PE	I32	Potenza/Energia apparente in VA/VAh sulla fase L2 (**)
L3_Apparent_PE	I32	Potenza/Energia apparente in VA/VAh sulla fase L3 (**)

Nota: (*) Se nell'oggetto 0x0E11 è salvato il valore 0x0 (default) allora in questi campi ci sarà il valore quadratico medio, se invece è salvato il valore 0x1 il valore istantaneo.

Nota: (**) Se nell'oggetto 0x0E12 è salvato il valore 0x0 (default) allora in questi campi ci sarà la potenza, se invece è salvato il valore 0x1 l'energia.

Nota: (***) Se nell'oggetto 0x0E13 è salvato il valore 0x0 (default) allora in questi campi ci sarà tensione di linea, se invece è salvato il valore 0x1 la tensione fase-fase.

Esempi

Esempio di utilizzo

```
INT pm_result
INT rid, slot           ; rid and slot of device AXl_SE_IOL4
STRU_PM_PD  pm          ; buffer used to store PM process data

; get now the information from the module
pm_result = axio_pm_r(rid, slot, pm, sizeof(pm));
if (pm_result < 0)
    ; write here your diagnostic
else
```

© 2025 Robox SpA

```
    ; test if all the lines have a voltage bigger than 100V
    if ((pm.L1_voltage >  100000) and (pm.L2_voltage > 100000 ) and (p
        ;do something
    endif
endif
```

Struttura STRU_REACHTARGET

Struttura STRU_REACHTARGET

Vedi: Tipi di dato 814

Questa keyword permette di definire una variabile di tipo STRU_REACHTARGET (STRUCT_P 918).

Viene utilizzata dall'istruzione di movimento mv_reach_target 477().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
velSlave	REAL	Velocità corrente slave, deve essere aggiornata a cura del software applicativo
velMinSlave	REAL	Velocità minima consentita durante l'aggancio
velMaxSlave	REAL	Velocità massima consentita durante l'aggancio
accMAxSlave	REAL	Accelerazione massima consentita durante l'aggancio
vTarget	REAL	Velocità desiderata dello slave nella posizione di sincronismo
ds	REAL	Spazio mancante allo slave al punto di sincronismo
tm	REAL	Tempo mancante al master al punto di sincronismo
si	REAL	sample interval

Esempi

Esempio di dichiarazione:

```
; definizione di variabile mov1 di tipo STRU_REACH_TARGET
STRU_REACHTARGET mov1
```

Esempio di utilizzo:

```
; utilizzo della struttura per una movimentazione
; definizione di variabile m1 di tipo STRU_REACH_TARGET
```

© 2025 Robox SpA

```
STRU_REACHTARGET m1
; definizione di variabili accessorie
I32 esito
I32 fase
REAL targetMaster
REAL targetSlave
REAL rrPassoMaster
REAL rrPassoSlave
REAL rrAccSlave
; inserimento dei dati nella struttura
; prima esecuzione
if (first_time())
    m1.velMinSlave = rr(1002)                    ;velocità minima slave
    m1.velMaxSlave = rr(1003)                    ;velocità massima slave
    m1.accMaxSlave = rr(1004)                    ;accelerazione massima
    m1.si = si                                   ;tempo di campionamento
    fase = 0
endif
; eseguo le operazioni in base alla variabile "fase"
select (phase)
    case 0  ;=== fase iniziale ===
        ; con l'attivazione dell'input 100 faccio partire l'inseguimen
        if (rise(inp(100)))
            iv(2) = 0
            targetMaster = ip(1) + rrPassoMaster
            targetSlave  = ip(2) + rrPassoSlave
            fase = 1
        endif
        break
    case 1  ;=== fase 1 ===
        ; Calcolo tempo che manca al master per arrivare alla quota ta
        m1.tm = (targetMaster - ip(1)) / iv(1)
        ; Calcolo spazio che manca allo slave
        m1.ds = targetSlave - ip(2)
        ; Aggiorno velocità del master
        m1.vtarget = iv(1)
        ; Aggiorno velocità dello slave
        m1.velSlave = iv(2)
        ; Muovo l'asse
        iv(2) = mv_reach_target (esito, m1)
        ; se ho raggiunto il target passo alla fase 2
        if (esito = M_REACHED)
            fase = 2
        endif
        break
    case 2  ;=== fase 2 ===
        ; continuo alberato al master per 100 unit
        ; poi passo alla fase 3
        iv(2) = iv(1)
        _if ((ip(ax2) - targetSlave) > 100 )
             fase = 3
```

© 2025 Robox SpA

```
        break
    case 3  ;=== fase 3 ===
        ; fermo l'asse slave
        iv(2) = ramp(iv(2) , 0 , rrAccSlave)
        ; poi torno alla fase 0
        if ((iv(2) < K_EPSILON))
            fase = 0
        endif
        break
end_select
```

Struttura STRU_SAFETY_PRJINFO

Validità: **RC3E v33.9.1-3** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_SAFETY_PRJINFO.

Viene utilizzata dalla funzione safe_prj_info 574().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
desc	STRING	Descrizione customer
prjName	STRING	Nome del progetto
ver	STRING	Versione
ct	STRING	Data e ora compilazione
isle	U8	Numero dell'isola
reserved_0	U8	Riservato
HB_crc	U32	Project header CRC
LB_crc	U32	Logical block CRC
AB_crc	U32	Address block CRC

Esempi

Esempio di utilizzo:

```
INT esito_spi
STRU_SAFETY_PRJINFO spi

esito_spi = safe_prj_info 574(1, spi)
if (esito_spi < 0)
    ; write here your diagnostic
else
    ; set value in registers for HMI
    SR(10) = spi.prjName
    R(10)  = spi.HB_crc
```

© 2025 Robox SpA

```
    R(11)   = spi.LB_crc
    R(12)   = spi.AB_crc
endif
```

Struttura STRU_SOE_IDN

Validità: **RC3E v33.5.19-3** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_SOE_IDN.

Questa variabile viene utilizzata dalle funzioni soe_ridn 603() e soe_widn 615() per la lettura/scrittura dei parametri SOE.

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
rid	INT	Robox ID associato al dispositivo
idn	INT	numero del parametro S o P da leggere o scrivere
element	INT	solo se lettura - elemento da leggere 0 è assunto come DATA () (vedi tabella codici element)
cmd	INT	Solo se scrittura - gestione per comandi - 0 (non comando) 1,2,3 (vedi tabella codici) Se lettura impostare a 0
isp	INT	selezione tipo di parametro : • se 0 si gestisce S-0-idn • se 1 si gestisce P-0-idn
datatype	INT	tipo di dato da leggere /scrivere (vedi tabella codici)
datalen	INT	Se usata per SOE_RIDN: dimensione massima in byte della area per memorizzare la risposta Se usata per SOE_WIDN: dimensione in byte dei dati da scrivere.
phase	INT	fase di lavoro/ modalitá di esecuzione: 1--> inizializzazione esecuzione passante (**) -1--> inizializzazione esecuzione bloccante(**)

© 2025 Robox SpA

Campo	Tipo	Descrizione
		2--> reserved RTE 3--> reserved RTE 4--> reserved RTE 5--> reserved RTE 6--> operazione eseguita con successo 0--> non inizializzato o errore altro--> valore illegale solo i valori contrassegnati con (**) SONO I SOLI scrivibili in questa variabile da parte dell'utente. Il valore bloccante (-1) viene cambiato dall'istruzione, in caso di esecuzioni consecutive delle funzioni SOE_RIDN 603 o SOE_WIDN 615 il valore -1 DEVE essere reimpostato prima dell'esecuzione della stessa.
nric	INT	numero di caratteri ricevuti. Utile in caso di array per dedurre quanti elementi si sono ricevuti
res_1	INT	reserved
res_2	INT	reserved
res_3	INT[260]	reserved

Tabella codici element:

	Elements	Code
STS	parameter status	0x01
NAME	parameter name	0x02
ATTR	parameter attribute	0x04
UNIT	parameter units	0x08
MIN	parameter minimum value	0x10
MAX	parameter maximum value	0x20
DATA	parameter data (Default)	0x40
DEF	parameter default value	0x80

Tabella codici:

© 2025 Robox SpA

	Codice
1	Executed command procedure (with autocheck if ended).
2	Stop command procedure.
3	Send the specified value to handle command procedure.

Codici tipi di dato (dataType)

Codice	Tipo di dato
0	Selezione automatica (default).
2	INTEGER8
3	INTEGER16
4	INTEGER32
5	UNSIGNED8
6	UNSIGNED16
7	UNSIGNED32
8	REAL32
9	VISIBLE_STRING
10	OCTET_STRING
17	REAL64
21	INTEGER64
27	UNSIGNED64
45	BITARR8
46	BITARR16
47	BITARR32
-2	ARRAY INTEGER8
-3	ARRAY INTEGER16
-4	ARRAY INTEGER32
-5	ARRAY UNSIGNED8
-6	ARRAY UNSIGNED16
-7	ARRAY UNSIGNED32
-8	ARRAY REAL32

© 2025 Robox SpA

Codice	Tipo di dato
-9	VISIBLE_STRING SOE

Esempi

Esempio di definizione:

```
; definizione della variabile s di tipo STRU_SOE_IDN
STRU_SOE_IDN s
```

Struttura STRU_STAT

Questa keyword permette di definire una variabile di tipo STRU_STAT (STRUCT_P 918).

Viene utilizzata dall'istruzione statistics 621().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
initdone	I32	Flag per memorizzare inizializzazione fatta
min	REAL	Valore minimo raggiunto dalla variabile sulla quale si esegue la statistica
max	REAL	Valore massimo raggiunto dalla variabile sulla quale si esegue la statistica
average	REAL	Valore medio della variabile sulla quale si esegue la statistica
ncount	REAL	Numero di campioni desiderati sul quale eseguire la media
actcount	REAL	Numero di campioni attuale (modulato su ncount)
acttot	REAL	Totalizzatore somma dei valori campione

Esempi

Esempio di dichiarazione:

```
; definizione di variabile miaStatistica di tipo STRU_STAT
STRU_STAT miaStatistica
```

Esempio di utilizzo:

```
; utilizzo dei dati contenuti nella struttura
```

© 2025 Robox SpA

```
; definizione di variabile miaStatistica di tipo STRU_STAT
STRU_STAT statT1
; definizione di variabili accessorie
REAL reso, T1, minVal, maxVal
; imposto a 1000 il numero dei cicli su cui eseguire la media
statT1.ncount = 1000
...
; nel main loop
; misuro la durata del loop
T1 = loop_time()
; memorizzo il valore medio nella variabile in "reso"
reso = statistics(T1,statT1)
; memorizzo il valore minimo nella variabile in "minVal"
minVal = statT1.min
; memorizzo il valore massimo nella variabile in "maxVal"
maxVal = statT1.max
```

Struttura STRU_SYNCHRO

Questa keyword permette di definire una variabile di tipo STRU_SYNCHRO (STRUCT_P).

Viene utilizzata dall'istruzione di movimento mv_synchro().

È composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
qMaster	REAL	Posizione della grandezza master. Deve essere aggiornata prima di ogni chiamata di funzione.
velMaster	REAL	Velocità della grandezza master. Deve essere aggiornata prima di ogni chiamata di funzione.
qSyncMaster	REAL	Quota di sincronismo dell'asse master
qSlave	REAL	Posizione della grandezza slave. Deve essere aggiornata prima di ogni chiamata di funzione.
velSlave	REAL	Velocità della grandezza slave. Deve essere aggiornata prima di ogni chiamata di funzione.
qSyncSlave	REAL	Quota di sincronismo dell'asse slave
velSyncSlave	REAL	Velocità che deve avere lo slave alla quota di sincronismo qSyncSlave

© 2025 Robox SpA

Campo	Tipo	Descrizione
velMaxSlave	REAL	È il limite superiore di velocità ammessa per il movimento dell'asse slave. Nel caso la formula di correzione richiedesse di salire al di sopra della velocità massima, il valore reso dalla funzione sarebbe pari a velMaxSlave.
accMaxSlave	REAL	È il valore di accelerazione massima accettato per l'asse slave. La quantità di errore correggibile viene automaticamente ridotta in modo da non superare il valore di accelerazione impostato.
si	REAL	Tempo di campionamento si
kappa	REAL	Permette di deformare la traiettoria dello slave: tanto più il valore è alto, tanto più il movimento slave viene anticipato, cioè si cerca di percorrere uno spazio maggiore durante la prima metà della missione. Valori ragionevoli vanno da 1.0 (nessun anticipo) fino a 3.0 (massimo anticipo).

Esempi

Esempio di dichiarazione:

```
; definizione di variabile acciuffa di tipo STRU_SYNCHRO
STRU_SYNCHRO acciuffa
```

Esempio

```
; utilizzo della struttura per una movimentazione
; definizione di variabile m1 di tipo STRU_SYNCHRO
STRU_SYNCHRO m1
; definizione di variabili accessorie
I32 esito
I32 fase
REAL rrPassoMaster
REAL rrPassoSlave
REAL rrAccSlave
; inserimento dei dati nella struttura
; prima esecuzione
if(first_time 346())
    m1.velMaxSlave = rr(1003)                ;velocità massima slave
    m1.accMaxSlave = rr(1004)                ;accelerazione massima
```

© 2025 Robox SpA

```
    m1.si = si                                  ;tempo di campionamento
    m1.kappa = rr(1006)                         ;parametro di deformazi
    fase = 0
endif
; eseguo le operazioni in base alla variabile "fase"
select (fase)
    case 0                                      ;fase iniziale (0)
        ; setto quota di aggancio master.
        ; Quando il master raggiungerà m1.qsyncMaster anche
        ; lo slave avrà raggiunto m1.qsyncSlave e in quel
        ; punto avrà la velocità m1.velsyncslave
        m1.qsyncMaster = ip(1) + rrPassoMaster
        ; setto quota di aggancio slave
        m1.qsyncSlave = ip(2) + rrPassoSlave
        fase = 1
        break
    case 1                                      ;fase 1
        ; Aggiorno parametri master
        m1.qMaster = ip(1)
        m1.velMaster = iv(1)
        ; Aggiorno parametri slave
        m1.qSlave = ip(2)
        m1.velSlave = iv(2)
        ; setto la velocità dello slave sul punto di aggancio
        m1.velsyncslave = iv(1)
        ; Muovo l'asse
        ip(2) = mv_synchro[479](esito, m1)
        ; Se ho raggiunto la sincronizzazione passo alla fase 2
        if(esito = M_REACHED)
            fase = 2
        endif
        break
    case 2                                      ;fase 2
        ; continuo alberato al master per 100 unit
        ; poi passo alla fase 3
        iv(2) = iv(1)
        _if ((ip(2) - m1.qsyncSlave) > 100)
            fase = 3
        break
    case 3                                      ;fase 3
        ; fermo l'asse slave
        iv(2) = ramp[555](iv(2) , 0 , rrAccSlave)
        ; poi passo alla fase 4
        if ((iv(2) < K_EPSILON))
            fase = 4
            time = TFB
        endif
        break
    case 4                                      ;fase 4
        ; Aspetto 1 secondo
        _if ((tfb - time) > 1)
```

© 2025 Robox SpA

```
            ; Poi torno alla fase 0
            fase = 0
        break
end_select
```

Struttura STRU_TRACK

Validità: **RC3E v33.3.2** (o successive)

Questa keyword permette di definire una variabile di tipo STRU_TRACK (STRUCT_P 918).

Viene utilizzata dalla funzione tracking 648().

E' composta dalle seguenti informazioni:

Campo	Tipo	Descrizione
despos	REAL	Quota da raggiungere desiderata. può essere anche una grandezza che evolve nel tempo (quota di un altro asse)
desspeed	REAL	Deve essere aggiornata ad ogni chiamata della funzione e rappresenta la velocità del campo "despos". In caso di asse rappresenta la IV dell'asse stesso mentre in caso di variabile può essere calcolata come la variazione del campo "despos" nel tempo: [nuovo target - despos precedente / campo "delta"]
actpos	REAL	Quota attuale della grandezza controllata dalla funzione
actspeed	REAL	Velocità attuale della grandezza controllata dalla funzione
maxspeed	REAL	Massima velocità della grandezza controllata dalla funzione
acc	REAL	Massima accelerazione della grandezza controllata dalla funzione
posmin	REAL	Paracarro inferiore slave della grandezza controllata dalla funzione
posmax	REAL	Paracarro superiore slave della grandezza controllata

© 2025 Robox SpA

Campo	Tipo	Descrizione
		dalla funzione
delta	REAL	Intervallo di campionamento (si)

Esempi

Esempio di dichiarazione:

```
; definizione di variabile trk di tipo STRU_TRACK
STRU_TRACK trk
```

Esempio di utilizzo:

```
; utilizzo della struttura per una movimentazione
; se è attivo il comando di inseguimento l'asse 2 insegue l'asse 1
; definizione di variabile m1 di tipo STRU_TRACK
STRU_TRACK m1
; definizione di variabili accessorie
I32 trakka
I32 reso
; se viene dato il comando di inseguimento
if(trakka)
    ; inserisco i dati nella struttura
    m1.delta = si                                   ;intervallo di cam
    m1.desspeed = iv(1)                             ;velocità desidera
    m1.despos = ip(1)                               ;posizione desider
    m1.actpos = ip(2)                               ;posizione corrent
    m1.actspeed = iv(2)                             ;velocità corrente
    m1.maxspeed = max_spe(2)                        ;velocità massima
    m1.acc = max_acc(2)                             ;accelerazione mas
    m1.posmin = min_str(2)                          ;paracarro inferio
    m1.posmax = max_str(2)                          ;paracarro superio
    ; chiamo la funzione di inseguimento
    reso = tracking 648 (m1)
    ;assegno all'asse fisico la velocità calcolata
    ip(2) = m1.actpos
    ; se invece non è attivo il comando di inseguimento...
else
    ; ...fermo l'asse
    iv(2) = ramp 555 (iv(2) ,0 , max_acc(2))
endif
```

Struttura STRUCT

Questa keyword permette di definire strutture di dati.

La definizione della struttura inizia con la keyword "struct" seguita dal nome che l'utente desidera assegnare alla struct.

In seguito é possibile dichiarare tipo e nome dei campi che compongono la struct.

© 2025 Robox SpA

Infine la Struct viene terminata con la keyword "end_struct".

Esempi

Di seguito una struct di nome "CUSTOM_DATE" contenente tre interi rispettivamente chiamati "day", "month", "year":

```
struct CUSTOM_DATE
    INT day
    INT month
    INT year
end_struct
```

Di seguito inizializziamo la struct precedente con la data di fondazione di ROBOX:

```
CUSTOM_DATE foundation

foundation.day = 20
foundation.month = 2 ;febbraio
foundation.year = 1975
```

Esempio di utilizzo di vettori in una struct:

```
struct MYSTRUCT
    INT intNumber[32]    ;32 interi
    FLOAT floatingPointNumber[16]   ;16 float
end_struct
...
MYSTRUCT testStruct
testStruct.intNumber[5] = 1975
testStruct.floatingPointNumber[3] = 3,1415926536
```

Esempio di utilizzo di vettori di struct:

```
struct MYSTRUCT
    INT intNumber
    FLOAT floatingPointNumber
end_struct
...
MYSTRUCT testStruct[8]   ;dichiaro un array di 8 struct
testStruct[0].intNumber = 1975
testStruct[4].floatingPointNumber = 3,1415926536
;esempio di utilizzo dei campi in un calcolo
testStruct[6].floatingPointNumber = testStruct[0].intNumber * testStru
```

NOTA: Non e' possibile utilizzare dei registri per indexare array di strutture o index array interni.

Struttura STRUCT_P

E' equivalente alla STRUCT 917 con la differenza che i dati sono allineati al byte.

NOTA: Utilizzo tipico per la definizione di strutture per lo scambio dati con dispositivi esterni.

© 2025 Robox SpA

Struttura TABSTR

Validità: **Rule**, **Task**

Questa keyword permette di definire una TABELLA di COSTANTI STRINGA.

La definizione viene chiusa dalla keyword "ENDTAB".

Le costanti stringa possono essere raggruppate in tabelle. Alla tabella deve essere associato un identificatore; la singola stringa sarà individuata dall'identificatore di tabella cui dovrà essere associato il numero indice della stringa in questione. L'indice della prima stringa è 0, quello della seconda è 1, e così via.

Campo	Tipo
"testo"	STRING
"testo"	STRING
"testo"	STRING
"testo"	STRING
"testo"	STRING
.....	STRING

Esempi

Esempio di utilizzo:

```
; definizione di una variabile STRING di nome messaggio
STRING messaggio
; definizione di una struttura TABSTR di nome comandi
tabstr comandi
    "Velocità"
    "Posizione"
    "Accelerazione"
endtab
; messaggio conterrà "la velocità è eccessiva"
messaggio = "la " # comandi[0] # " è eccessiva"
; messaggio conterrà "la accelerazione è eccessiva"
messaggio = "la " # comandi[2] # " è eccessiva"
```

Tipo BOOL

Questa keyword permette di definire una variabile booleana.

Sintassi	**BOOL**
Valori rappresentabili	1 (TRUE), 0 (FALSE)
Annotazioni	

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
;---------------------------------------
; definizione variabili locali task 1
;---------------------------------------
BOOL pippo, pluto
pippo = FALSE
pluto = 1
```

Tipo CAST

Con il termine "forzamento" (CAST) si intende la possibilità di convertire una grandezza appartenente a un certo tipo in un'altra del tipo specificato. La grandezza in questione può essere indifferentemente una costante, una variabile oppure il risultato di una espressione matematica.

Campo	Tipo	Descrizione
Forzamento a I8 925	I8	espressione/variabile/costante
Forzamento a I16 923	I16	espressione/variabile/costante
Forzamento a I32 924	I32	espressione/variabile/costante
Forzamento a U8 932	U8	espressione/variabile/costante
Forzamento a U16 930	U16	espressione/variabile/costante
Forzamento a U32 931	U32	espressione/variabile/costante
Forzamento a REAL 928	REAL	espressione/variabile/costante
Forzamento a FLOAT 922	FLOAT	espressione/variabile/costante
Forzamento a INT 924	Equivale a I32	
Forzamento a UNSIGNED 931	Equivale a U32	
Annotazioni:		È opportuno utilizzare un forzamento tutte le volte che si desidera cambiare la normale sequenza delle conversioni implicitamente effettuate all'interno di una espressione matematica.

Esempi

© 2025 Robox SpA

Esempio di utilizzo:

```
; Il contenuto del registro reale 4 viene convertito in intero
; quindi gli viene sommato 2 e assegnato a r(5).
r(5) = (I32) (rr(4)) + 2

; La costante intera 2 viene convertita (automaticamente) in tipo REAL
; e viene quindi sommata al valore contenuto nel registro reale 4.
; Il risultato viene quindi convertito in tipo I32 e assegnato a r(5).
; Si noti che in questo esempio il forzamento (I32) è superfluo
; in quanto non altera la sequenza di operazioni che si sarebbe verifi
; Al contrario, nell'esempio precedente il forzamento(I32)
; produce una sequenza di eventi diversa da quella naturale.
r(5) = (I32) (rr(4) + 2)

; Il contenuto del registro intero r(2) viene convertito in reale
; e viene quindi moltiplicato per la costante reale 65536.0
; Il contenuto del registro intero r(1) viene convertito in U32
; e convertito successivamente in reale (conversione implicita)
; e sommato quindi al risultato della precedente moltiplicazione.
; Il risultato finale viene assegnato al registro reale 5.
rr(5) = (U32)(r(1)) + (REAL)(r(2)) * 65536.0
```

Tipo CHAR

È equivalente a I8 925.

Validità: **Rule**, **Task**

Questa keyword permette di definire una variabile contenente un carattere, con riferimento alla codifica ASCII.

Sintassi	**CHAR**
Minimo numero rappresentabile	-128
Massimo numero rappresentabile	127
Dimensione	1 Byte
Annotazioni	Se precede, tra parentesi, il nome variabile effettua un operazione di forzamento "CAST 920"

Tipo COUNTER

Questa keyword permette di definire una variabile di tipo COUNTER (contatore).

Questa variabile viene utilizzata dalle istruzioni di gestione counter ct_cu 699() , ct_cd 697() , ct_cur 700() , ct_cdr 698().

Esempi

© 2025 Robox SpA

Esempio di dichiarazione:

```
; definizione delle variabili counter1 e counter2 di tipo COUNTER
COUNTER counter1, counter2
```

Tipo FLOAT

Questa keyword permette di definire il tipo di variabile come floating point (formato 4 byte - IEEE Std 754 e 854).

Sintassi	**FLOAT**
Numero minimo rappresentabile	8.43e-37
Numero massimo rappresentabile	3.37e+38
Dimensione	4 Byte
Annotazioni	Se precede, tra parentesi, il nome variabile effettua un operazione di forzamento "CAST 920"

Esempi

Esempio di utilizzo:

```
;----------------------------------------
; definizione variabili locali task 1
;----------------------------------------
FLOAT pippo, pluto
FLOAT appoggio
; esempio di forzamento tipo "CAST"

FLOAT f1, f2, f3, f4
...
r(100) = 1000000000                         ; 1e9
r(102) = 5
r(103) = 128
r(104) = 5
...
; la moltiplicazione avviene tra interi.... quindi con i dati sopra sc
; 5 * 1e9 = 1000000000 -> 0x12A05F200, che viene troncato in 0x2A05F20
f1 = r(100) * r(101)                        ; f1 = 7.050327e+8
; la moltiplicazione avviene tra interi ma il cast (FLOAT) forza la co
f2 = (FLOAT)r(100) * r(101)                 ; f2 = 5e+9
if (R(103))
    ; la divisione avviene tra interi
    f3 = R(102) / r(103)                    ; f3 = 25
    ; i dati sono convertiti in FLOAT prima di effettuare la divisione
    f4 = (FLOAT)r(102) / r(103)             ; f3 = 25.6
else
    f3 = 0
```

© 2025 Robox SpA

```
    f4 = 0
endif
```

Tipo I16

Validità: **Rule**, **Task**

Questa keyword permette di definire una variabile numerica intera a 16 bit con segno in complemento a 2 (C2).

Sintassi	**I16**
Numero minimo rappresentabile	-32768
Numero massimo rappresentabile	32767
Dimensione	2 Byte
Annotazioni	Se precede, tra parentesi, il nome variabile effettua un operazione di forzamento "CAST 920"

Esempi

Alcuni numeri nei due sistemi di rappresentazione:

Rappresentazione esadecimale	Rappresentazione decimale
0XA	10
0AH	10
10H	16
0FFH	255
0x200	512
0xFE00	-512
7FFFH	32767
8000H	-32768
0FFFFH	-1

Esempio di utilizzo:

```
;---------------------------------------
; definizione variabili locali task 1
;---------------------------------------
I16 pippo, pluto
I16 appoggio
; esempio di forzamento tipo "CAST"
; appo è una variabile di tipo intero con segno a 32 bit
```

© 2025 Robox SpA

```
; peppo è una variabile di tipo intero con segno a 16 bit
I32 appo
I16 peppo
; il valore di appo viene convertito da 32 a 16 bit
; quindi assegnato a peppo
peppo = (I16) (appo)
```

Tipo I32

Validità: **Rule**, **Task**

Queste keyword permettono di definire una variabile numerica intera a 32 bit in complemento a 2 con segno.

Nota: Le due keyword sono equivalenti. L'utilizzo di I32 oppure di INT e' lasciato al gusto del programmatore.

Sintassi	**I32** **INT**
Numero minimo rappresentabile	-2147483648
Numero massimo rappresentabile	2147483647
Dimensione	4 Byte
Annotazioni	Se una delle due keyword precede, tra parentesi, il nome di una variabile, viene effettuata un'operazione di forzamento "CAST 920"

Esempi

Alcuni numeri nei due sistemi di rappresentazione:

Rappresentazione esadecimale	Rappresentazione decimale
0XA	10
0AH	10
10H	16
0FFH	255
0FFFFH	65535
7FFFFFFFH	2147483647
80000000H	-2147483648
0FFFFFFFFH	-1

Esempio di utilizzo:

```
;----------------------------------------
```

© 2025 Robox SpA

```
; definition of local variables of task 1
;---------------------------------------
I32 pippo, pluto
I32 appoggio
INT integerVar_1
; esempio di forzamento tipo "CAST"
; date due variabile reali (il cui contenuo in valore assoluto e' infe
; si puo' procedere come segue :
REAL first, second, sumResult, integerSumResult
...
first  = 1.23445
second = 3.98765
...
sumResult        = first + second             ; sumResult = 5.22215
integerSumResult = (I32)first + (I32)second   ; integerSumResult = 4
```

Tipo I8

Validità: **Rule**, **Task**

Questa keyword permette di definire una variabile numerica intera a 8 bit in complemento a 2 con segno.

Sintassi	**I8**
Minimo numero rappresentabile	-128
Massimo numero rappresentabile	127
Dimensione	1 Byte
Annotazioni	Se precede, tra parentesi, il nome variabile effettua un operazione di forzamento "CAST 920"

Esempi

Alcuni numeri nei due sistemi di rappresentazione:

Rappresentazione esadecimale	Rappresentazione decimale
0XA	10
0AH	10
10H	16
7FH	127
80H	-128
0FFH	-1

Esempio di utilizzo:

© 2025 Robox SpA

```
;----------------------------------------
; definizione variabili locali task 1
;----------------------------------------
I8 pippo, pluto
I8 appoggio
; esempio di forzamento tipo "CAST"
; appo è una variabile di tipo intero con segno a 16 bit
; peppo è una variabile di tipo intero con segno a 8 bit
I16 appo
I8 peppo
; il valore di appo viene convertito da 16 a 8 bit
; quindi assegnato a peppo
peppo = (I8)(appo)
```

Tipo POWER_SET

Questa keyword permette di definire il tipo di variabile come POWER_SET.

Attenzione: Quando viene creato un Power Set tramite ambiente RDE, viene automaticamente definita (all'interno del file cfgdefs.i3) una variabile di tipo POWER_SET con il nome del Power Set. Di norma, quindi, non c'e' bisogno di definire la variabile da task R3.

Sintassi	**POWER_SET**

E' composta dalle seguenti informazioni:

Campo	Tipo	R/W	Descrizione
power_enable	BOOL	R/W	Abilitazione del power set
channel_enable	U32	R/W	Abilitazione dei drive del power set
channel_enable_feedback	U32	R	Maschera dei segnali di feedback dei singoli drive
channel_reset_fault	U32	R/W	Reset di un allarme di uno o più drive
drive_reset	U32	R/W	Reset di uno o più drive
status	U32	R	Informazioni di stato del power set (vedi ps_status 543)
channel_enabled	U32	R	Informazioni sui drive abilitati del power set
channel_fault	U32	R	Informazioni sui drive in fault del power set
eba	BOOL	R	Consenso all'energizzazione del

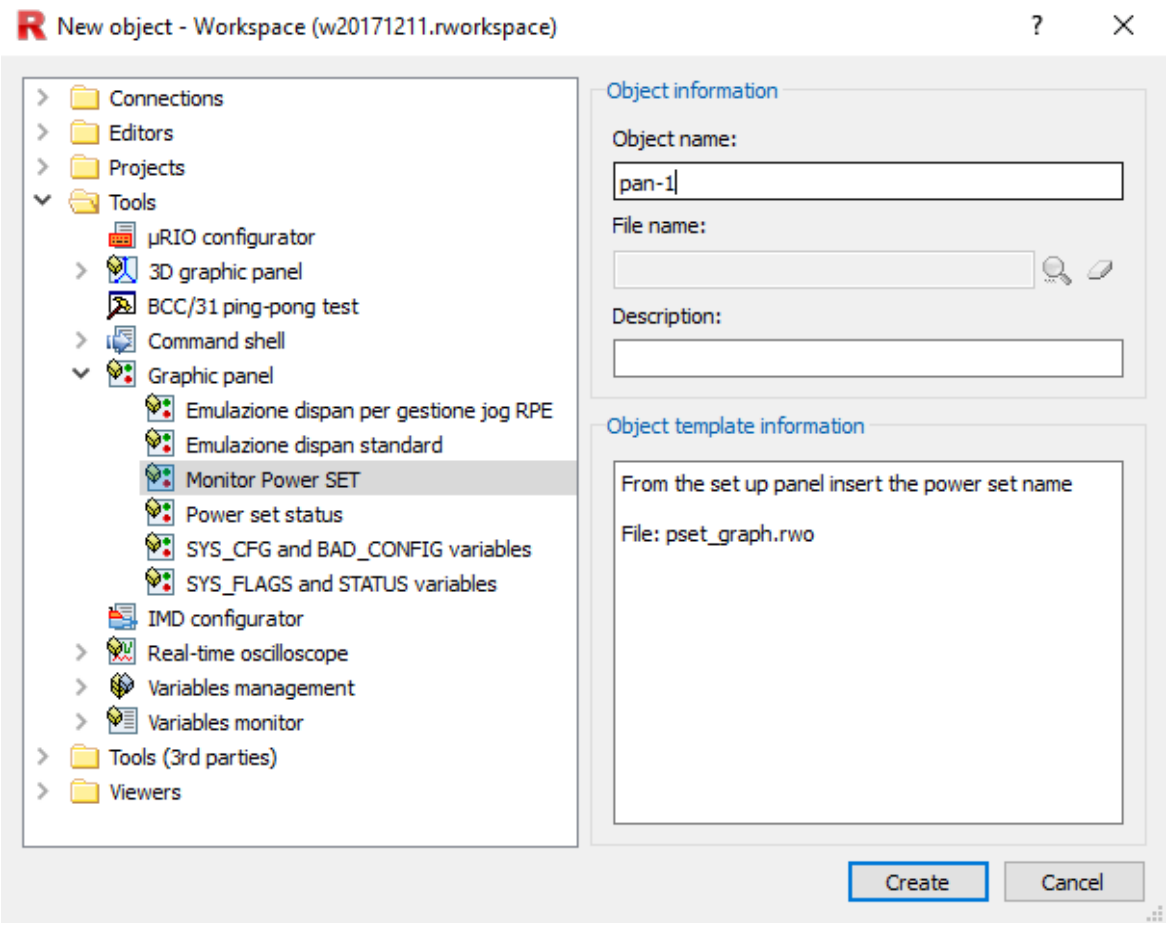
© 2025 Robox SpA

Campo	Tipo	R/W	Descrizione
			power set da parte degli allarmi
prq	BOOL	R	Presenza del segnale di richiesta potenza
gpfb	BOOL	R	Presenza della serie dei segnali di feedback
gpfm	U32	R	Maschera dello stato dei singoli segnali di feedback definiti da configuratore
gpfc	U32	R	Maschera dello stato dei singoli segnali di feedback degli assi gestiti CAN402
pam	U32	R	Physical Axis Mask
lam	U32	R	Logic Axis Mask

E' possibile accedere alle informazioni del power set sia per mezzo delle variabili che per mezzo di apposite funzioni.

NOTA: In RDE e' presente un pannello grafico predefinito nel quale vengono visualizzate le informazioni presenti nella struttura POWER_SET. Per legarlo al power set desiderato, bisogna inserire il nome del power set come valore di PSNAME, che si trova sotto impostazioni -> parametri

© 2025 Robox SpA



Esempi

Esempio di utilizzo:

```
; definizione (presente in cfgdefs.i3) di una variabile di tipo POWER_
POWER_SET isola1
; accesso alle informazioni del power set tramite variabile
; la variabile "stato_pwr" conterrà le informazioni di stato del power
U32 stato_pwr
stato_pwr = isola1.status
```

Tipo REAL

Validità: **Rule**, **Task**

Questa keyword permette di definire il tipo di variabile come REAL, vale a dire floating point 8 Byte (double).

Sintassi	**REAL**
Minimo numero rappresentabile	2.225074e-308
Massimo numero rappresentabile	1.797693e+308

© 2025 Robox SpA

Sintassi	**REAL**
Dimensione	8 Byte
Annotazioni	Se precede tra parentesi il nome variabile effettua un operazione di forzamento "CAST 920" È possibile inizializzare il valore della variabile usando la seguente sintassi: REAL nomeVariabile = { 0.0 }

Esempi

Esempio di utilizzo:

```
;----------------------------------------
; definizione variabili locali task 1
;----------------------------------------
REAL pippo, pluto
REAL appoggio
; definizioni di array di due elementi con relativa inizializzazione
REAL accostoX[2] = {  0.0,  0.0 }
REAL accostoY[2] = {  0.0, 50.0 }
REAL accostoZ[2] = { 50.0, 50.0 }
; esempio di forzamento tipo "CAST"
; appo è una variabile di tipo intero con segno a 32 bit
; peppo è una variabile di tipo REAL
I32 924 appo
REAL peppo
; il valore di appo viene convertito da 32 bit a REAL
; quindi assegnato a peppo
peppo = (REAL) (appo)
```

Tipo STRING

Validità: **Rule**, **Task**

Questa keyword permette di definire il tipo di variabile come STRING, vale a dire stringhe di caratteri.

Sintassi	**STRING** (len=128)
len	Lunghezza stringa (opzionale, default 128).
Annotazioni	Il terminatore è 0 ed occupa la posizione successiva all'ultimo carattere.

La lunghezza delle stringhe è un parametro opzionale, se omessa il valore di default è 128 caratteri (127+1).

Il terminatore è gestito da RTE.

Esempi

© 2025 Robox SpA

Esempio di definizione:

```
; con la seguente dichiarazione si associa:
; - alla stringa pippo una lunghezza massima di 32 caratteri (31 utili
; - alla stringa zippo una lunghezza massima di default di 128 caratte
; - alla stringa string1 una lunghezza massima di 2000 caratteri (1999
; Se la lunghezza è specificata, il numero di caratteri della stessa
; non dovrà superare detto valore, ma potrà essere minore o uguale.
STRING pippo(32), zippo, string1(2000)
```

Tipo TIMER

Questa keyword permette di definire una variabile di tipo TIMER.

Questa variabile viene utilizzata dalle funzioni di gestione Timer tm_on 782() , tm_of 780() , tm_int 779() , tm_sqw 784().

Esempi

Esempio di dichiarazione:

```
; definizione delle variabili timer1 e timer2 di tipo TIMER
TIMER timer1, timer2
```

Tipo U16

Questa keyword permette di definire una variabile numerica intera a 16 bit senza segno.

Sintassi	**U16**
Minimo numero rappresentabile	0
Massimo numero rappresentabile	65535
Dimensione	2 Byte
Annotazioni	Se precede, tra parentesi, il nome variabile effettua un operazione di forzamento "CAST 920"

Esempi

Alcuni numeri nei due sistemi di rappresentazione:

Rappresentazione esadecimale	Rappresentazione decimale
0XA	10
0AH	10
10H	16
7FH	127

© 2025 Robox SpA

Rappresentazione esadecimale	Rappresentazione decimale
80H	128
0FFH	255
7FFFH	32767
8000H	32768
0FFFFH	65535

Esempio di utilizzo:

```
;----------------------------------------
; definizione variabili locali task 1
;----------------------------------------
U16 pippo, pluto
U16 appoggio
; esempio di forzamento tipo "CAST"
; appo è una variabile di tipo intero senza segno a 32 bit
; peppo è una variabile di tipo intero senza segno a 16 bit
U32 appo
U16 peppo
; il valore di appo viene convertito da 32 a 16 bit
; quindi assegnato a peppo
peppo = (U16)(appo)
```

Tipo U32

Questa keyword permette di definire una variabile numerica intera a 32 bit senza segno.

Sintassi	**U32**
Minimo numero rappresentabile	0
Massimo numero rappresentabile	4294967295
Dimensione	4 Byte
Annotazioni	Se precede, tra parentesi, il nome variabile effettua un operazione di forzamento "CAST 920"

Esempi

Alcuni numeri nei due sistemi di rappresentazione:

Rappresentazione esadecimale	Rappresentazione decimale
0XA	10
0AH	10

© 2025 Robox SpA

Rappresentazione esadecimale	Rappresentazione decimale
10H	16
7FH	127
80H	128
0FFH	255
7FFFH	32767
8000H	32768
0FFFFH	65535
7FFFFFFFH	2147483647
80000000H	2147483648
0FFFFFFFFH	4294967295

Esempio di utilizzo:

```
;----------------------------------------
; definizione variabili locali task 1
;----------------------------------------
U32 pippo, pluto
U32 appoggio
; esempio di forzamento tipo "CAST"
; date due variabile reali (il cui contenuo in valore assoluto e' infe
; si puo' procedere come segue :
REAL first, second, sumResult, integerSumResult
...
first  = 1.23445
second = 3.98765
...
sumResult        = first + second             ; sumResult = 5.22215
integerSumResult = (U32)first + (U32)second   ; integerSumResult = 4
```

Tipo U8

Questa keyword permette di definire una variabile numerica intera a 8 bit senza segno.

Sintassi	**U8**
Minimo numero rappresentabile	0
Massimo numero rappresentabile	255
Dimensione	1 Byte
Annotazioni	Se precede, tra parentesi, il nome variabile effettua un operazione di forzamento "CAST"

© 2025 Robox SpA

Esempi

Alcuni numeri nei due sistemi di rappresentazione:

Rappresentazione esadecimale	Rappresentazione decimale
0XA	10
0AH	10
10H	16
7FH	127
80H	128
0FFH	255

Esempio di utilizzo:

```
;----------------------------------------
; definizione variabili locali task 1
;----------------------------------------
U8 pippo, pluto
U8 appoggio
; esempio di forzamento tipo "CAST"
; appo è una variabile di tipo intero senza segno a 16 bit
; peppo è una variabile di tipo intero senza segno a 8 bit
U16 appo
U8 peppo
; il valore di appo viene convertito da 16 a 8 bit
; quindi assegnato a peppo
peppo = (U8)(appo)
```

Varie

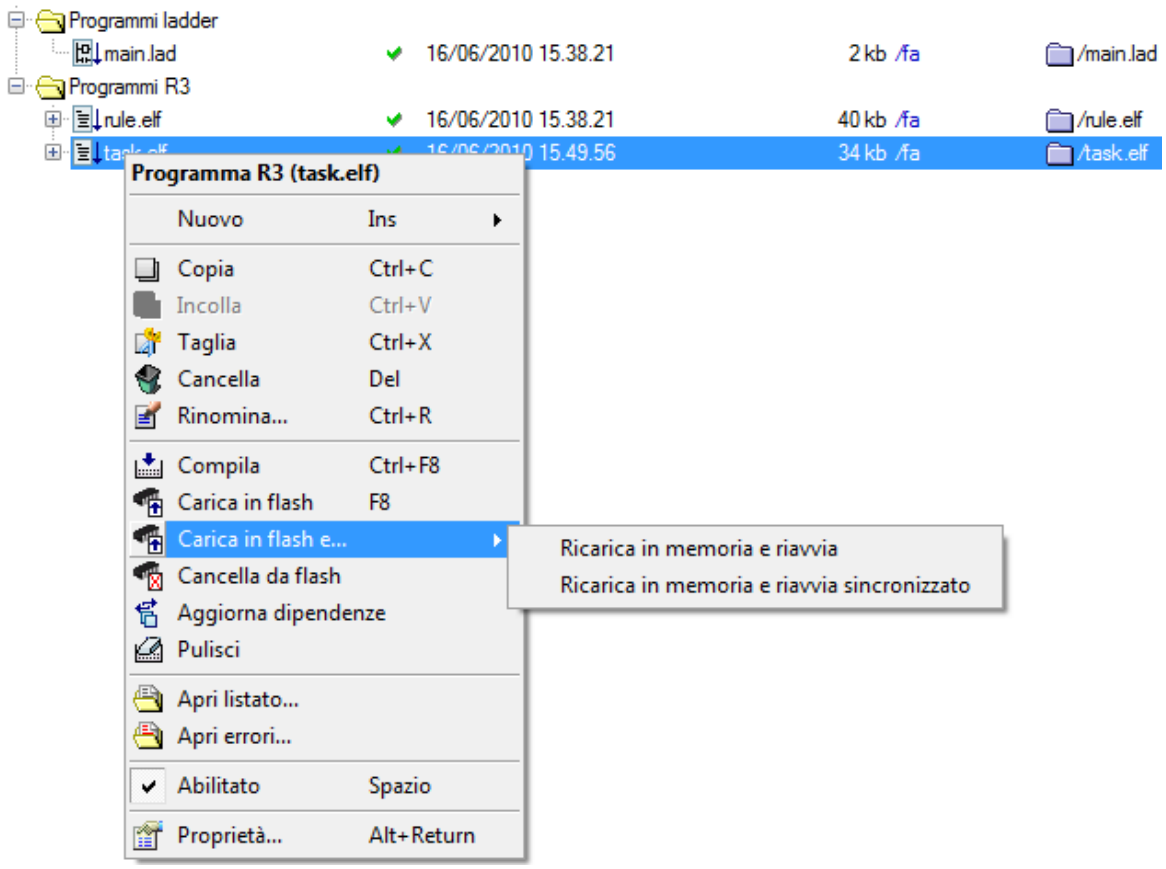
Modifiche LIVE

Da RTE v33.18, riferito a programmi scritti in R3, è possibile "iniettare" dei programmi modificati mentre il controllo è in modo esecuzione (led arancione mode lampeggiante) senza dover eseguire un boot di sistema.

Operatività

Dopo aver eseguito la modifica al programma ed averlo compilato con zero errori, dalla finestra di progetto è possibile effettuare l'operazione in due modi diversi a scelta del programmatore:

© 2025 Robox SpA



Il programmatore dovrà quindi strutturare il suo programma (task/rule) in modo tale da evitare che vengano rieseguite funzioni di inizializzazione che tipicamente vengono messe prima del loop principale. A tal fine è possibile utilizzare le variabili predefinite RESTARTED e SYNC_RESTART

I registri volatili R, RR ed SR che vengono azzerati al boot non vengono azzerati in caso di modifica "LIVE".

La keyword RESTART_POINT 746 permette di selezionare il punto in cui il programma si deve fermare, prima di far partire (dall'inizio) quello caricato (questo solo se è stato selezionato il riavvio sincronizzato)

La keyword RESTART_POINT 746 può essere utilizzata più volte nello stesso programma.

Ricarica in memoria e riavvia

- Il programma (TASK) viene fermato all' istruzione in cui si trova e viene sostituito con quello nuovo
- La variabile predefinita RESTARTED segnala la presenza di questa situazione
- Se la modifica viene fatta alle RULE il sistema deve essere in missing power (GLOBAL_POWER = false)

Ricarica in memoria e riavvia sincronizzato

- Il programma (TASK) viene fermato alla prima istruzione RESTART_POINT 746 e viene sostituito con quello nuovo
- La variabile predefinita SYNC_RESTART segnala la presenza di questa situazione
- La keyword RESTART_POINT può essere utilizzata più volte nello stesso programma

© 2025 Robox SpA

- Se la modifica viene fatta alle RULE il sistema deve essere in missing power (GLOBAL_POWER = false)

Esempi

Esempio di utilizzo:

```
; boot backing already executed
lit rInitDone r(1000)
lit rOperativeMode r(1001)

; if first boot init is started so I can handle the LIVE modifications
_if (not rInitDone)
    call InitTask1
rInitDone = true

; Main loop
__main_loop__
    ; stop point in the running program
    restart_point

    ; management of operating mode changes
    call HndModeChange
    out (150) = inp(1)

    ; machine state manager
    select (rOperativeMode)
        case M_MIS_POWER          ; Missing power handling
           call HndMisPower
           break
        ...
    end_select
    ...
end_main_loop
```

© 2025 Robox SpA

Indice

- - -

- 804

- # -

808

- $ -

$check_array 798
$close_loop 798
$define 800
$else 801
$endif 801
$error 798
$expand 798
$ifdef 801
$ifndef 801
$include 798, 802
$rule 798
$rule_inp 798
$rule_periodic 798
$rule_time 798
$step 798
$task 798
$undef 798, 802
$use_rpe 798
$warning 798
$write_on_extern 798

- * -

* 804

- / -

/ 804

- _ -

__main_loop__ 677
_else 678
_if 678

- + -

+ 804

- < -

< 811
<= 811
<> 811

- = -

= 808, 811

- > -

> 811
>= 811

- A -

abs 156
abs_mod 157
acos 157
ah_cmd 158
ah_get_d 159
ah_get_s 160
ah_info 162
ah_log 163
ah_log_us 164
al2am 166
alarm_reset 169
alarm_reset_par 170
alarm_reset_par_us 172
alarm_set 175
alarm_set_us 176
and 809
as_get_d 178
asin 178
Asse lineare: ciclo di zero a micro 69
Asse lineare: ciclo di zero a micro + index 74
Asse lineare: ciclo di zero a paracarro 79
Asse lineare: ciclo di zero a paracarro + index 82
Asse Rotativo Monodirezionale: ciclo0 a micro + index con ricerca preliminare della variazione del micro 88
Asse rotativo monodirezionale: ciclo0 a micro + index SENZA ricerca preliminare della variazione del micro 94
Asse rotativo monodirezionale: ciclo0 solo a micro con ricerca preliminare della variazione del micro 99
Asse rotativo monodirezionale: ciclo0 solo a micro senza ricerca preliminare della variazione del micro 104
atan 179
atan2 180
aux 679
axes 680
axes_m 680
axio_iol_r 181

© 2025 Robox SpA

axio_iol_w 182
axio_pm_r 183
axio_robj 186
axio_wobj 199

- B -

b<n> 213
b0 213, 681
b1 215, 681
b2 215, 681
b3 216, 682
b4 216, 682
b5 217, 682
b6 218, 683
b7 218, 683
bcc_get_double 219
bcc_get_float 220
bcc_get_i16 220
bcc_get_i32 221
bcc_get_i8 221
bcc_get_string 222
bcc_get_tcp_clients_info 222
bcc_get_u16 223
bcc_get_u32 224
bcc_get_u8 224
bcc_h_setup 225
BCC_MSGUSER 820
bcc_receive 226
bcc_reply_ack 227
bcc_reply_msg 228
bcc_reply_nack 229
bcc_send 229
bcc_set_double 683
bcc_set_float 684
bcc_set_i16 684
bcc_set_i32 685
bcc_set_i8 685
bcc_set_string 686
bcc_set_u16 686
bcc_set_u32 687
bcc_set_u8 688
bit_mask_add 688, 798
bit_mask_remove 690
bit_mask_set 692, 798
BOOL 919
break 694

- C -

call 694
case 696
CAST 920
CHAR 921
check_mv_follow2 230
check_mva_follow2 233
check_mva_follow2_m 236
co_asw_sdo 239
co_controlword_om 241
co_controlword_state 243
co_get_ws_diagno 245
co_par_download 248
co_par_upload 251
co_rerr 255
co_rnmt 256
co_robj 258
co_rpdo 265
co_send_sdo 266
co_statusword 270
co_sync 272
co_wnmt 272
co_wobj 273
co_wpdo 278
Codici di errore funzione 59
Codici di errore Modbus 68
Codici reso funzioni MV_ e MVA_ 63
coe_robj 279
coe_wobj 284
command 288
cond_val 289
continue 697
cos 290
Costanti 17
COUNTER 921
crc16_be 290
crc16_le 292
ct_cd 697
ct_cdr 698
ct_cu 699
ct_cur 700
ct_cv 294, 701
ct_load 701
ct_pv 294, 701
ct_qd 295

© 2025 Robox SpA

ct_qu 295
ct_reset 702

- D -

dec 295
default 702
delta_enc16 296
delta_enc32 297
diff 299
do 703
dw<n> 300
dw0 300, 703
dw1 301, 704
dwbe2le 302
dwbe2lef 302
dwbe2mbe 303
dwbe2mle 303
dwell 704
dwle2be 304
dwle2bef 304
dwle2mbe 305
dwle2mbef 305
dwle2mle 306
dwle2mlef 306
dwmbe2be 307
dwmbe2le 307
dwmbe2lef 308
dwmle2be 308

- E -

ecat_dev_on_off 309
ecat_get_state 312
ecat_probe 313
ecat_set_state 314
else 704
end_aux 705
end_do_while 706
end_for 706
end_fun 706
end_main_loop 707
end_motion 707
end_ref 708
end_rule 708
end_select 709
end_while 710
endif 710
excl_lam 316, 711
excl_pam 317, 712

- F -

f_avail 318
f_close 318
f_copy 319
f_del 320
f_dir 321
f_format 323
f_gets 324
f_mkdir 326
f_open 327
f_puts 329
f_read 330
f_read_table 331
f_read_table_s 333
f_rename 335
f_seek 336
f_write 337
fall 338
fall_filt 339
fb_ctrlw 270
fb_stsw 270
fill_mem 712
filter_lp 341
filter_lp_st 342
filter_notch 343
filter_notch_st 345
first_time 346
FLOAT 922
for 713
force 347
fract 350
function 715
Funzioni/istruzioni matematiche 109
Funzioni/istruzioni safety 111
Funzioni/istruzioni supporto rule/task 112

- G -

Generalità grammatica 790
Generalità linguaggio R3 791
Gestione allarmi 114
Gestione Axioline 116
Gestione canali seriali 116
Gestione CANopen/EtherCAT 118
Gestione CNET 120
Gestione comunicazione BCC 121

© 2025 Robox SpA

Gestione controllo di flusso 123
Gestione conversione endianess 125
Gestione counter 130
Gestione dati 131
Gestione flash 136
Gestione forzatura I/O 138
Gestione modbus/TCP client 138
Gestione MOT 142
Gestione movimento 142
Gestione power set 145
Gestione stringhe 146
Gestione timer 148
Gestione UDP/IP 148
Gestione variabili globali 149
Gestioni miscellanee 152
get_float 350
get_hw_info 351
get_i16 353
get_i32 354
get_i64 719
get_i8 355
get_real 355
get_u16 356
get_u32 357
get_u64 720
get_u8 358
get_virt_ow 359
goto 721
group 721
gv_create 359
gv_datatype 362
gv_datatype_byname 362
gv_destroy 363
gv_exist 364
gv_flags 364
gv_flags_byname 366
gv_link 368
gv_lock 369
gv_name 370
gv_read 370
gv_read_partial 371
gv_size 372
gv_size_byname 372
gv_unlink 373
gv_unlock 373
gv_update_file 374
gv_write 374
gv_write_partial 375

- I -

I16 923
I32 924
I8 925
if 723
in_range 376
inc 377
index_of 377
Indice funzioni/istruzioni 20
Indice generale 17
inf_report 724
int_after 725
int_inp 726
int_timer 728
ipaddr_to_str 378
is_alarm_par_present 379
is_alarm_par_present_us 381
is_alarm_present 383
is_forced 384

- K -

K_1_PI 17
K_2PI 17
K_DUEPI 17
K_E 17
K_EPSILON 17
K_MAXFLOAT 17
K_MAXREAL 17
K_MINFLOAT 17
K_MINREAL 17
K_PI 17
K_PI_2 17
K_PIMEZZI 17
kin_conv 386

- L -

lit 729
log 388
log10 388
loop_time 389

- M -

M_ACCEL 63
M_ACCEL_LIMITED 63
M_ADJUSTING_DOWN 63

© 2025 Robox SpA

M_ADJUSTING_UP 63
M_AT_SPEED 63
M_DECEL 63
M_FOLLOWING 63
M_HOLDED 63
M_HOMING 63
M_ILLEGAL_ARGS 63
M_IN_ALARM 63
M_INITIALIZING 63
M_LAST_CYCLE 63
M_LOW_ACCELERATION 63
M_OUT_OF_LIMITS 63
M_OVERSHOOTING 63
M_POSITIONING 63
M_REACHED 63
M_SLAVE_EARLY_STOPPED 63
M_SLAVE_IS_EARLY 63
M_SLAVE_TOO_LATE 63
M_SPEED_LIMITED 63
M_SYNCHRONIZED 63
M_WAITING 63
M_WAITING_NEXT_CYCLE 63
master_filter 389
maximum 390
mbc_connect 391
mbc_init 391
mbc_mask_write_reg 394
mbc_mask_write_reg_s 396
mbc_read_coils 398
mbc_read_coils_s 399
mbc_read_discr_inp 400
mbc_read_discr_inp_s 402
mbc_read_hold_reg 403
mbc_read_hold_reg_s 405
mbc_read_inp_reg 406
mbc_read_inp_reg_s 408
mbc_shutdown 410
mbc_write_coils 410
mbc_write_coils_s 412
mbc_write_multiple_coils 413
mbc_write_multiple_coils_s 414
mbc_write_multiple_regs 416
mbc_write_multiple_regs_s 418
mbc_write_reg 419
mbc_write_reg_s 421
mbcu_mask_write_reg 422
mbcu_read_coils 424
mbcu_read_discr_inp 426
mbcu_read_hold_reg 427
mbcu_read_inp_reg 429
mbcu_write_coils 431
mbcu_write_multiple_coils 433
mbcu_write_multiple_regs 434
mbcu_write_reg 436
Metacomandi 798
minimum 438
mod 439
Modifiche LIVE 933
mot_append 439
mot_clear 443
mot_create 443
mot_delete 447
mot_get 447
mot_remove 449
motion 730
move_byte 731
move_mem 732
mt_ats 449
mt_ats2 450
mt_dis 733
mt_en 734
mt_rr_flag 734
mt_wait 735
mv_cam 451
mv_crimper 453
mv_follow 454
mv_follow2 458
mv_mot_exec 470
mv_phase_adj 472
mv_phase_adj2 474
mv_ramp 477
mv_reach_target 477
mv_sine_cam 478
mv_synchro 479
mv_table 481
mv_to 482, 484
mv_to (inline) 484
mv_to (struct) 482
mv_to_cj 486
mv_to_cjv 487
mv_to_cjv_info 489
mv_to_vel 489

© 2025 Robox SpA

mv_tracking 491
mva_follow2 458, 467
mva_follow2_m 458, 469
mva_jog 491
mva_jog2 493
mva_open_loop 496
mva_to_n 496
mva_to_n_cj 500
mva_to_n_v 504
mva_zc 511

- N -

n 734
normalize 513
not 809

- O -

onoff_filt 514
Operatori 803
Operatori a bit (binari) 806
Operatori aritmetici 804
Operatori di assegnazione 808
Operatori di concatenamento stringhe 808
Operatori logici 809
Operatori relazionali numerici 811
or 809
order 735
out_m 737

- P -

par_even 516
par_odd 517
planar_comp_cp 518
planar_comp_ip 519
pos_capture_cmd 520
pos_capture_sts 524
pow 525
POWER_SET 926
ps_channel_enable 526
ps_channel_enabled 529
ps_channel_fault 531
ps_channel_reset_fault 532
ps_drive_reset 535
ps_exclude_lax 538
ps_exclude_pax 539
ps_power_enable 540
ps_reinclude_lax 541
ps_reinclude_pax 542
ps_status 543

- Q -

qwbe2le 738
qwbe2lef 547
qwbedwx2le 739
qwbedwx2lef 547
qwle2be 739
qwle2bedwx 740
qwle2bedwxf 740
qwle2bef 741
qwle2mbe 741
qwle2mbedwx 742
qwle2mbedwxf 742
qwle2mbef 743
qwmbe2le 743
qwmbe2lef 548
qwmbedwx2le 744
qwmbedwx2lef 548

- R -

r_and 806
r_not 806
r_or 806
r_rol 549
r_ror 550
r_shl 551
r_shr 552
r_xor 806
r2m16 553
r2m32 554
r2m8 554
ramp 555
rand32 556
random 557
random_value 557
range 557
range_adj 558
rbit 559
REAL 928
ref 744
ref_prologue 745
release 560
restart_point 746
return 746
rise 562

© 2025 Robox SpA

rise_filt 563
round 565
rtc_date 565
rtc_time 566
rule 748
rule_epilogue 756
rule_freq 756
rule_init 757
rule_periodic 758
rule_prologue 759
rule_start_norc 760

- S -

safe_appl_ack 567
safe_appl_diag 568
safe_diag 569
safe_diag_reset 572
safe_enab_out 573
safe_oper_ack 573
safe_prj_info 574
safe_sts 575
sb<n> 577
sb0 577, 760
sb1 578, 761
sb2 578, 761
sb3 579, 761
sb4 580, 762
sb5 580, 762
sb6 581, 762
sb7 582, 763
sbit 582
sdw<n> 583
sdw0 583, 763
sdw1 584, 763
select 764
ser_485_tx_sts 585
ser_bcc_h_init 587
ser_bin_in 588
ser_bin_out 588
ser_bin_ready 589
ser_buff_free 590
ser_buff_queue 590
ser_con 765
ser_flush 590
ser_in 591
ser_mod 592
ser_out 593
ser_r_sts 593
ser_ready 594
ser_w_sts 766
set_float 767
set_i16 768
set_i32 769
set_i64 770
set_i8 771
set_kinematics 595
set_real 772
set_u16 772
set_u32 773
set_u64 774
set_u8 775
set_version 776
set_virt_iw 595
shift_left 596
shift_right 597
shl 599
shr 599
sign 600
similar 601
sin 602
sizeof 602
soe_ridn 603
soe_widn 615
sqrt 620
statistics 621
store_nvr_nopf 622
store_nvrr_nopf 623
str_asc 624
str_chr 625
str_compact 625
str_equal 626
str_find 627
str_format 627
str_i 629
str_insert 630
str_len 631
str_mid 631
str_r 632
str_replace 633
str_to_ipaddr 634
str_val 635
STRING 929

© 2025 Robox SpA

STRU_AAX 822
STRU_AH_D 823
STRU_AH_D_EXT 824
STRU_AH_INFO 825
STRU_AS_D 826
STRU_AS_D_EXT 827
STRU_AXIO_PDI 828
STRU_BCC_TCP_CLIENT 830
STRU_CAM 831
STRU_CO_OBJ 832
STRU_COE_OBJ 834
STRU_COERRMSG 836
STRU_CRIMPER 839
STRU_ECAT_PROBE 841
STRU_FILEHND 842
STRU_FILEINFO 843
STRU_FOLLOW 844
STRU_FOLLOW2 846
STRU_FOLLOW2_INFO 854
STRU_FOLLOW2_S 855
STRU_FOLLOW2_SM 862
STRU_GROR 869
STRU_HW_INFO 870
STRU_INT 870
STRU_LARGE_INT 872
STRU_LARGE_UINT 873
STRU_MASTERF 874
STRU_MATCAM 875
STRU_MODBUS 876
STRU_MVGENITP 877
STRU_MVITP 879
STRU_MVITPCJ 880
STRU_MVTO 881
STRU_MVTOCJ 884
STRU_MVTOCJV 886
STRU_MVTOCJV_INFO 888
STRU_MVTOCJVE 890
STRU_MVZC 892
STRU_PHADJ 898
STRU_PHADJ2 901
STRU_PM_PD 904
STRU_REACHTARGET 906
STRU_SAFETY_PRJINFO 908
STRU_SOE_IDN 909
STRU_STAT 912
STRU_SYNCHRO 913
STRU_TRACK 916
STRUCT 917
STRUCT_P 918
sum16_mem 638
sum32_mem 638
sum8_mem 637
sumr_mem 638
sw<n> 639
sw0 639, 777
sw1 639, 777
sw2 640, 778
sw3 641, 778

- T -

TABSTR 919
tan 641
terminator 778
time_micro 642
TIMER 930
Tipi di ciclo di zero 69
tm_et 642, 779
tm_int 779
tm_of 780
tm_on 782
tm_pt 643, 784
tm_pt2 644, 784
tm_q 645
tm_reset 784
tm_sqw 784
tm_tp 786
tobcd 645
tobin 646
todeg 646
tolower 647
torad 647
toupper 648
tracking 648
trunc 650

- U -

U16 930
U32 931
U8 932
udp_close 651
udp_open_client 651
udp_open_server 654
udp_recv 657

© 2025 Robox SpA

udp_recv_from 658
udp_recv_notify 660
udp_send 662
udp_send_notify 665
udp_send_to 667

- V -

virt_io 669

- W -

w<n> 671
w0 671, 787
w1 672, 787
w2 672, 788
w3 673, 788
wbe2le 674
wbit 674
while 789
wle2be 675
write_dis 676

- X -

xor 809

© 2025 Robox SpA