Logic Lab

User's Manual

Manual Code: MU00001E00 V_2.5

Index

FIRMWARE VERSIONS: 12.1 / 22.1

Application level available resources

- 65536 word of Program Memory FLASH
- 4Kword of Data Memory RAM
- 30Kword Data Memory available on permanent memory (EEPROM)
- FAST task executed togheter motor control routines evevry PWM period
- CYCLIC task with Process Image, executed cyclically with a selectable period (usually some ms). This task can be interrupted by Fast and motor control routines
- SLOW task for background
- Direct serial access with Modbus rtu protocol (functions “Preset Multiple Registers” and “Read Holding Registers” with 4KWord from address 0x2000)
- Direct CANOpen access with up to 100 configurable Dictionary objects and possibility to configure 4SDO, 4 TPDO ,4 RPDO, NMT state
- Secondary CAN line for Drive to Drive communication or I/O Expansion
- 100 Extra Parameters (E00-E99)
- 64 Internal values (d64-d127) that can be seen into display and OPD Explorer
- Using Standard I/O Management (“Standard_IO()”):
  - 32 Logical Input Functions (I00-I31) that can be configured on 8 physical inputs.
  - 32 Logical Output Functions (O32-O63) that can be configured on 4 physical outputs
  - 32 Internal value for monitor and analog outputs (O68-O99) that can be configured on 2 physical outputs

1. DATA BLOCKS

The purpose of the data blocks is to allow firmware declared variables to be accessed from the PLC code. These variables could be read only or read/write.

1.1 GROUP = "I/O VARIABLES"

Type	Index	Name	Elements		R/W	Description
Input	IX0.0	sysLogicalInput	ARRAY[8]	BOOL	R	8 physical logic inputs state
Input	IX1.0	sysTabDigInput	ARRAY[32]	BOOL	R	32 logical input functions state (from connector, serial line and fieldbus)
Input	IW2.0	sysHwDigInputMask	1	INT	R	3 power inputs state into less significant bit: 0 =ALL_ENC; 1=MAXV; 2=RETEOFF
Input	ID3.0	sysAnalogInput	ARRAY[3]	REAL	R	3 analog inputs value [-1÷1]
Output	QX0.0	sysLogicalOutput	ARRAY[4]	BOOL	RW	4 physical logical outputs state
Output	QX1.0	sysTabDigOutput	ARRAY[64]	BOOL	R	64 logical output function state
Output	QD2.0	sysAnalogOutput	ARRAY[2]	REAL	RW	2 analog outputs value [-1÷1]

1.2 GROUP = "COUNTERS"

Type	Index	Name	Elements		R/W	Description
Input	IW6.0	sysPLDCounter	1	INT	R	12 bit frequency input counter
Input	IW7.0	sysPLDDeltaCount	1	INT	R	Frequency input delta pulses measured in PWM period

1.3 GROUP = "DRIVE STATUS"

Type	Index	Name	Elements		R/W	Description
Memory	MW10.0	sysActiveAlarms	1	UINT	R	Alarm word
Memory	MW11.0	sysStatusWord	ARRAY[16]	BOOL	R	Status array 0=Run state -- 2=Brake on -- 3=Main supply-off 4=Initial Reset period on -- 5=Active Alarms -- 8=Drive ok – 10=Power Soft Start On -- 12=Active Autotuning
Memory	MW12.0	sysAlarmsCode	ARRAY[16]	INT	R	Alarms subcode array
Memory	MW13.0	sysC41StatusWord	1	UINT	R	Status test C41 Bit0=UVW_test -- Bit1=Poles_test -- Bit2=test active Bit3=end test -- Bit5=Fasing on
Memory	MW14.0	sysC42StatusWord	1	UINT	R	Status test C42 Bit2=Rs_test – Bit3=Ls_test – Bit4=Id_test Bit5=Tr test -- Bit6=test active – Bit7=end test
Memory	MW15.0	sysKeysStatus	1	UINT	R	Remote Keyboard buttons status
Memory	MW16.0	sysLedStatus	1	UINT	RW	Remote Keyboard led status

1.4 GROUP = “MOTOR SENSOR”

Type	Index	Name	Elements		R/W	Description
Memory	MD30.0	sysActualSpeed	1	REAL	R	Actual motor speed referred to MOT_SPD_MAX (P65)
Memory	MD32.0	sysMechPosition	1	DINT	R	Absolute multi-turns mechanical motor position [65536=1turn]
Memory	MD33.0	sysSensorRead	1	DINT	R	Motor position read by sensor [2^{32} = 1turn]
Memory	MX34.0	sysAbsSensor	1	BOOL	R	Flag to confirm that data read from sensor are absolute on the turn
Memory	MX35.0	sysAbsSensorMultiturn	1	BOOL	R	Flag to confirm that data read from sensor are absolute multi-turns
Memory	MD36.0	sysSensorReadIncr	1	DINT	R	Motor incremental position read by sensor [2^{32} = 1turn]
Memory	MD37.0	sysSensorZeroTop	1	DINT	R	Zero top position read by sensor [2^{32} = 1turn]
Memory	MD38.0	sysSensorFracBit	1	DINT	R/W	Number of fractional bit of sysMechPositionShifted
Memory	MD38.1	sysMechPositionShifted	1	DINT	R	Actual encoder position ([32-sysSensorFracBit.sysSensorFracBit] bits)

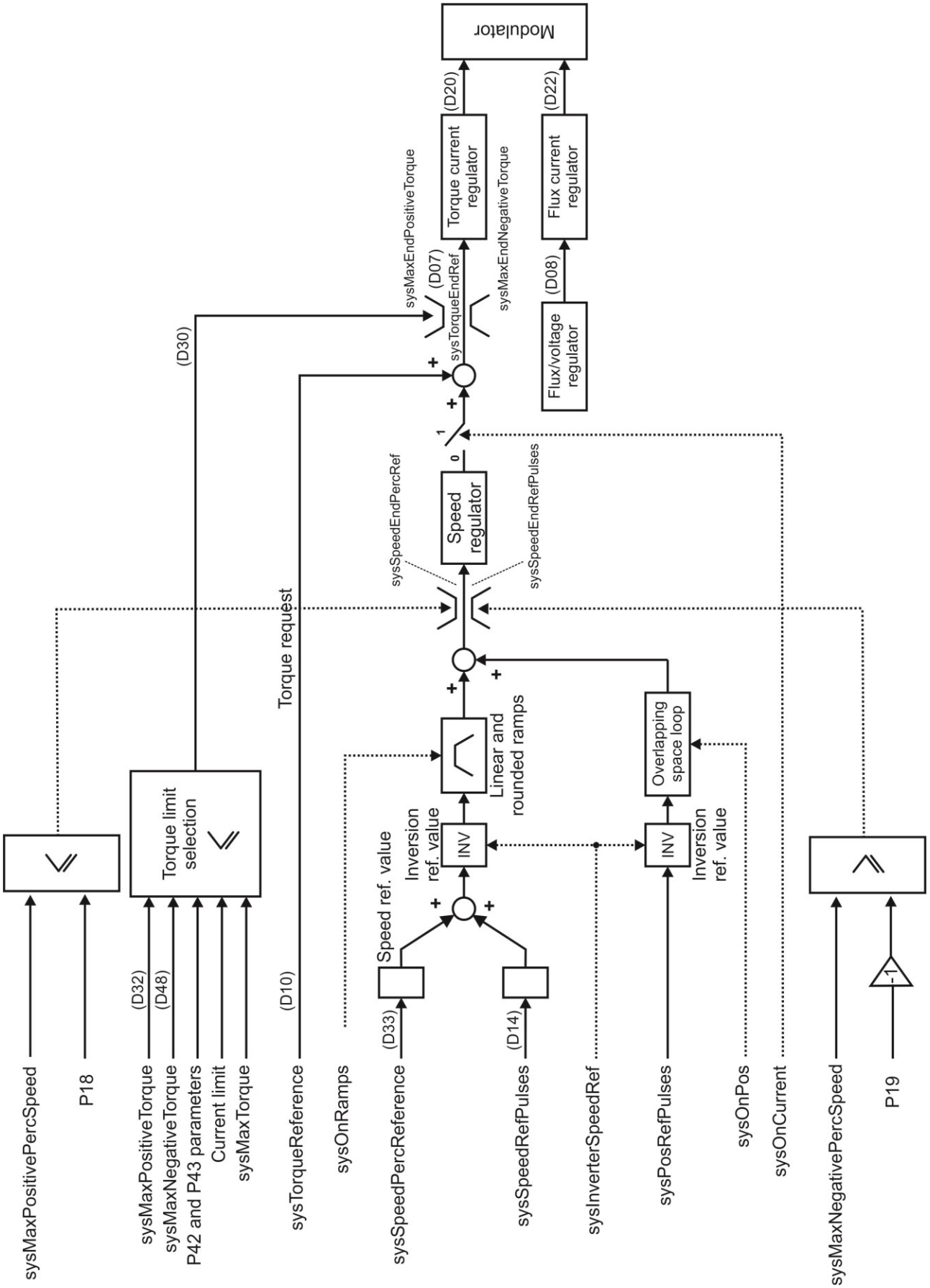
1.5 GROUP = “SECOND SENSOR”

Type	Index	Name	Elements		R/W	Description
Memory	MW40.0	sysSecondSensorCode	1	INT	R	Second sensor electronic card code
Memory	MX40.1	sysSecondSensorHwNotOk	1	BOOL	R	Second sensor electronic card and firmware selected consistency
Memory	MX40.2	sysSecondSensorNotConnected	1	BOOL	R	Second sensor presence
Memory	MX40.3	sysSecondSensorAbs	1	BOOL	R	Flag to confirm that data read from second sensor are absolute on the turn
Memory	MX40.4	sysSecondSensorAbsMultiturn	1	BOOL	R	Flag to confirm that data read from second sensor are absolute multi-turns
Memory	MD41.0	sysSecondSensorRead	1	DINT	R	Position read by second sensor [2^{32} = 1turn]
Memory	MD41.1	sysSecondSensorMechPosition	1	DINT	R	Absolute multi-turns mechanical second sensor position [65536=1turn]
Memory	MD41.2	sysSecondSensorActualSpeed	1	REAL	R	Actual second sensor speed referred to MOT_SPD_MAX (P65)
Memory	MD42.0	sysSecondSensorSine	1	REAL	R	Second sensor sine
Memory	MD43.0	sysSecondSensorCosine	1	REAL	R	Second sensor cosine
Memory	MD44.0	sysSecondSensorFracBit	1	DINT	R/W	Number of fractional bit of sysSecondSensorPositionShifted
Memory	MD44.1	sysSecondSensorPositionShifted	1	DINT	R	Actual second encoder position ([32-sysSecondSensorFracBit.sysSecondSensorFracBit] bits

1.6 GROUP = “CONTROL VARIABLES”

Type	Index	Name	Elements		R/W	Description
Memory	MD50.0	sysSpeedRefPulses	1	DINT	R/W	Speed reference in pulses per PWM period [2^{16}=1 mechanical turn if sysHighResPulsesRef=FALSE, 2^{32} =1 mechanical turn if sysHighResPulsesRef=TRUE]
Memory	MD50.1	sysPosRefPulses	1	DINT	R/W	Incremental position reference in pulses per PWM period [2^{16}=1 mechanical turn if sysHighResPulsesRef=FALSE, 2^{32} =1 mechanical turn if sysHighResPulsesRef=TRUE]
Memory	MD50.2	sysSpeedPercReference	1	REAL	R/W	Speed reference referred to MOT_SPD_MAX (P65)
Memory	MD50.3	sysMaxPositiveTorque	1	REAL	R/W	Maximum positive torque referred to MOT_T_NOM
Memory	MD50.4	sysMaxNegativeTorque	1	REAL	R/W	Maximum negative torque referred to MOT_T_NOM
Memory	MD50.5	sysMaxTorque	1	REAL	R/W	Symmetrical Torque limit referred to MOT_T_NOM
Memory	MD50.6	sysTorqueReference	1	REAL	R/W	Torque reference referred to MOT_T_NOM
Memory	MD50.7	sysSpeedRegKpCoeff	1	REAL	R/W	Speed regulator Proportional gain multipl factor
Memory	MD50.8	sysSpeedRegTaCoeff	1	REAL	R/W	Speed regulator lead time constant multiplication factor
Memory	MD50.9	sysMaxPositivePercSpeed	1	REAL	R/W	Maximum positive speed referred to MOT_SPD_MAX (P65)
Memory	MD50.10	sysMaxNegativePercSpeed	1	REAL	R/W	Maximum positive speed referred to MOT_SPD_MAX (P65)
Memory	MW51.0	sysRunCommand	1	BOOL	R/W	Run command
Memory	MW51.1	sysOnCurrent	1	BOOL	R/W	Enable only torque control
Memory	MW51.2	sysOnPos	1	BOOL	R/W	Enable overlapped incremental position loop
Memory	MW51.3	sysInvertSpeedRef	1	BOOL	R/W	Invert speed reference value
Memory	MW51.4	sysOnRamps	1	BOOL	R/W	Enable linear ramps on speed reference
Memory	MW51.5	sysStopIntegral	1	BOOL	R/W	Freeze speed regulator integral memory
Memory	MW51.6	sysResetAll	1	BOOL	R/W	Alarms reset
Memory	MW51.7	sysExtEnable	1	BOOL	R/W	External enable from the field
Memory	MW51.8	sysSelectBank	1	BOOL	R/W	Enable second parameter bank
Memory	MW51.9	sysMotorThermalSwitch	1	BOOL	R/W	Motor termo-witch state
Memory	MW51.10	sysdoubleSpeedRef	1	BOOL	R/W	Enable double speed frequency reference, in pulses per PWM period and time decoded
Memory	MW51.11	sysHighResPulsesRef	1	BOOL	R/W	Enable High Resolution (32bit) speed and position pulses reference
Memory	MD52.0	sysSpeedTheta	1	REAL	R	Conversion factor from percent of MOT_SPD_MAX to pulses per PWM period
Memory	MD53.0	sysSpeedEndRefPulses	1	DINT	R	Final speed reference in pulses per PWM period for integral part of speed regulator.

Memory	MD53.1	sysSpeedEndPercReference	1	REAL	R	Final speed reference referred to MOT_SPD_MAX for proportional part of speed regulator.
Memory	MD53.2	sysMaxEndPositiveTorque	1	REAL	R	Actual Maximum positive torque referred to MOT_T_NOM
Memory	MD53.3	sysMaxEndNegativeTorque	1	REAL	R	Actual Maximum negative torque referred to MOT_T_NOM
Memory	MD53.4	sysTorqueEndReference	1	REAL	R	Actual torque reference referred to MOT_T_NOM
Memory	MD53.5	sysMaxEndPositivePercSpeed	1	REAL	R	Final maximum positive speed referred to MOT_SPD_MAX (P65)
Memory	MD53.6	sysMaxEndNegativePercSpeed	1	REAL	R	Final maximum negative speed referred to MOT_SPD_MAX (P65)


sysMaxPositivePercSpeed
P18
sysMaxPositiveTorque
sysMaxNegativeTorque
P42 and P43 parameters
Current limit
sysMaxTorque
Torque limit selection
(D32)
(D48)
(D30)
(D10)
sysTorqueReference
sysOnRamps
sysSpeedPercReference
(D33)
Speed ref. value
Inversion ref. value
INV
Linear and rounded ramps
sysSpeedRefPulses
(D14)
sysInverterSpeedRef
sysPosRefPulses
Overlapping space loop
Inversion ref. value
INV
sysOnPos
sysOnCurrent
sysMaxNegativePercSpeed
P19
-1
sysSpeedEndPercRef
sysSpeedEndRefPulses
Speed regulator
1
0
sysMaxEndPositiveTorque
(D07)
sysTorqueEndRef
sysMaxEndNegativeTorque
Torque current regulator
(D20)
Modulator
Flux/voltage regulator
(D08)
Flux current regulator
(D22)

1.7 GROUP = “DRIVE PARAMETERS”

Type	Index	Name	Elements		R/W	Description
Memory	MW100.0	sysParameters	ARRAY[200]	INT	R	Parameters table
Memory	MW101.0	sysConnections	ARRAY[100]	INT	R	Connections table
Memory	MW102.0	sysInt	ARRAY[128]	INT	R	Internal value table
Memory	MW103.0	sysOSC	ARRAY[100]	INT	R	Monitor value table
Memory	MW104.0	sysExtraParameters	ARRAY[100]	INT	R	Application parameters table
Memory	MW105.0	sysExtraInt	ARRAY[70]	INT	R	OpdExplorer internal value table
Memory	MW106.0	sysCommands	ARRAY[12]	INT	R/W	Drive utility commands

- **MW102.0 sysInt** , refreshed in the background core task

Name	Description	UM	Scale
FW_REV	D00 - Software version		256
ACTV_POW	D01 - Active power delivered	kW	16
PRC_TOT_APP_SPD_REF	D02 - Speed reference value before ramp	% MOT_SPD_MAX	163.84
PRC_END_SPD_REF	D03 - Speed reference value after ramp	% MOT_SPD_MAX	163.84
PRC_MOT_SPD	D04 - Speed reading	% MOT_SPD_MAX	163.84
PRC_T_REF	D05 - Torque request	% MOT_T_NOM	40.96
PRC_IQ_REF	D07 - Request torque current Iq rif	% DRV_I_NOM	40.96
PRC_ID_REF	D08 - Request magnetizing current Id rif	% DRV_I_NOM	40.96
V_REF	D09 - Voltage reference value at max. rev.	% MOT_E_NOM	40.96
PRC_APP_T_REF	D10 - Torque reference value (application generated)	% MOT_T_NOM	40.96
MOT_I	D11 - Current module	A rms	16
REF_FRQ_IN	D12 - Frequency in input	KHz	16
EL_FRQ	D13 - Rotor flux frequency	Hz	16
PRC_APP_FRQ_SPD_REF	D14 - Frequency speed reference value (application generated)	% MOT_SPD_MAX	163.84
PRC_IQ	D15 - Current torque component	% DRV_I_NOM	40.96
PRC_ID	D16 - Current magnetizing component	% DRV_I_NOM	40.96
MOT_V	D17 - Stator voltage reference value module	V rms	16
PRC_MOT_V	D18 - Stator voltage reference value module	% MOT_E_NOM	40.96
MOD_INDEX	D19 - Modulation index		40.96
PRC_VQ_REF	D20 - Vq rif	% DRV_E_NOM	40.96
MOT_SPD	D21 - Motor rotation speed	rpm	1
PRC_VD_REF	D22 - Vd rif	% DRV_E_NOM	40.96
MOT_POS	D23 - Actual position	±16384	1
DC_BUS	D24 - Bus voltage	V	16
DRV_TEMP	D25 - Radiator temperature reading	°C	16

Name	Description	UM	Scale
MOT_TEMP	D26 - Motor temperature	°C	16
PRC_DRV_I_THERM	D28 - Motor thermal current	% soglia All	40.96
PRC_DRV_I_MAX	D29 - Current limit	% DRV_I_NOM	40.96
PRC_DRV_T_MAX	D30 - Maximum torque	% MOT_T_NOM	40.96
PRC_DRV_I_T_MAX	D31 - Maximum torque by current limit	% MOT_T_NOM	40.96
PRC_APP_T_MAX	D32 - Maximum torque limit by application	% MOT_T_NOM	40.96
PRC_APP_SPD_REF	D33 - Speed reference (application generated)	% MOT_SPD_MAX	163.84
PRC_MOT_T	D35 - Actual torque produced	% MOT_T_NOM	40.96
MOT_TURN_POS	D36 - Absolute mechanical position (on current revolution)	±16384	1
MOT_N_TURN	D37 - Number of revolutions		1
OFFSET_SINCOS_ENC	D38 - Compensation Sin/Cos analog/digital term	pulses	1
SENSOR_FRQ_IN	D39 - Input frequency	kHz	16
REG_CARD_TEMP	D40 - Regulation card temperature	°C	16
MOT_PRB_RES	D41 - Thermal probe resistance	Ohm	1
AI1	D42 - Analog Input AI1	%	163.84
AI2	D43 - Analog Input AI2	%	163.84
AI3	D44 - Analog Input AI3	%	163.84
SPD_ISR	D45 - Speed routine duration	us	64
I_ISR	D46 - Current routine duration	us	64
I_LOOP_BAND	D47 - Current loop bandwidth	Hz	1
PRC_APP_T_MIN	D48 - Minimum torque limit by application	% MOT_T_NOM	40.96
WORK_HOURS	D49 - Work Hours	hours	1
ENC_HALL_SECTOR	D50 - Encoder and Hall sens sector read		1
SENS2_SPD	D51 - Second sensor rotation speed	rpm	1
SENS2_TURN_POS	D52 - Second sensor Absolute mechanical position (on current revolution)	16384	1
SENS2_N_TURN	D53 - Second sensor Number of revolutions	16384	1
SENS2_FRQ_IN	D54 - Second sensor Frequency input	KHz	16
SENS1_ZERO_TOP	D55 - Sensor1 Zero Top	pulses	1
SENS2_ZERO_TOP	D56 - Sensor2 Zero Top	pulses	1
SYNC_DELAY	D57 - Delay from SYNC reception to Speed routine execution	us	1
PWM_SYNC_OFFSET	D58 - PWM offset for SYNC delay control	pulses	1
SERIAL_NUMBER	D59 - Drive Serial Number		1
FLD_CARD	D60 - Fieldbus Card		1
APPL_REV	D61 - Application Revision		40.96
HW_SENSOR2	D62 - Sensor2 presence		1
HW_SENSOR1	D63 - Sensor1 presence		1

- **MW103.0 sysOsc** (word), refreshed in the fast core task

Name	Description	UM	Scale
ACT_POS	O00 Actual mechanical position read by sensor	100%=180	327.67
ELECTRIC_POS	O01 Actual electrical position read by sensor	100%=180	327.67
PRC_TOT_APP_SPD_REF	O02 Reference speed value before ramps	% MOT_SPD_MAX	163.84
PRC_END_SPD_REF	O03 Reference speed value after ramps	% MOT_SPD_MAX	163.84
PRC_MOT_SPD	O04 Filtered Rotation speed	% MOT_SPD_MAX	163.84
PRC_T_REF	O05 Torque request	% MOT_T_NOM	40.96
PRC_IQ_REF	O07 Torque current request	% DRV_I_NOM	40.96
PRC_ID_REF	O08 Flux corrent request	% DRV_I_NOM	40.96
PRC_V_REF	O09 Request voltage at maximum rev.	% MOT_E_NOM	40.96
ALARMS	O10 Internal value: alarms		1
PRC_MOT_I	O11 Current module	% DRV_I_NOM	40.96
ZERO_TOP	O12 Zero top	pulses	1
PRC_IU	O13 U phase current reading	% DRV_I_MAX	40.96
INPUTS	O14 Physical inputs		1
PRC_IQ	O15 Torque component of current reading	% DRV_I_NOM	40.96
PRC_ID	O16 Magnetizing component of current reading	% DRV_I_NOM	40.96
TU	O17 U phase voltage duty-cycle		327.67
PRC_MOT_V	O18 Stator voltage reference value module	% MOT_E_NOM	40.96
MOD_INDEX	O19 Modulation index		40.96
PRC_VQ_REF	O20 Request Q axis voltage (Vq_rif)	% DRV_E_NOM	40.96
PRC_POWER	O21 Delivered power	%MOT_POW_NOM	40.96
PRC_VD_REF	O22 Request D axis voltage (Vd_rif)	% DRV_E_NOM	40.96
PRC_T_OUT	O23 Torque produced	% MOT_T_NOM	40.96
PRC_DC_BUS	O24 Bus voltage	%900V	40.96
PRC_DRV_TEMP	O25 Radiator temperature reading	%37.6°	40.96
PRC_MOT_TEMP	O26 Motor temperature reading	% 80°	40.96
PRC_DRV_I_THERM	O28 Motor thermal current	% soglia All	40.96
PRC_DRV_I_MAX	O29 Current limit	% DRV_I_NOM	40.96
PRC_DRV_T_MAX	O30 CW maximum torque	% MOT_T_NOM	40.96
PRC_DRV_T_MIN	O31 CCW maximum torque	% MOT_T_NOM	40.96
OUTPUTS	O32 Physical outputs		1
PRC_IV	O34 V phase current reading	% DRV_I_MAX	40.96
PRC_IW	O35 W phase current reading	% DRV_I_MAX	40.96
ALFA_FI	O36 Actual electrical position (alfa_fi) [		327.67
AI1	O37 Analog input A.I.1	100%=16383	163.84

AI2	O38 Analog input A.I.2	100%=16383	163.84
AI3	O39 Analog input A.I.3	100%=16383	163.84
SYS_SPD_PERC_REF	O41 Application speed reference value (sysSpeedPercReference)	% MOT_SPD_MAX	163.84
SYS_T_PERC_REF	O42 Application torque reference value (sysTorqueReference)	% MOT_T_NOM	40.96
SYS_T_MAX	O43 Application torque limit reference value (sysMaxTorque)	% MOT_T_NOM	40.96
SYS_SPD_REF_PULS	O44 Frequency speed reference value from application (sysSpeedRefPulses)	Pulses per TPWM	1
SYS_POS_REF_PULS	O45 Overlapped space loop reference value from application (sysPosRefPulses)	Pulses per TPWM	1
RES_AMPLITUDE	O46 Amplitude to the square of sine and cosine feedback signals	1=100%	1
RES_SIN	O47 Sen_theta		1
RES_COS	O48 Cos_theta		1
PRC_MOT_SPD	O49 Rotation speed not filtered	% MOT_SPD_MAX	163.84
PULSES_RD	O50 Delta pulses read in PWM period in frequency input	Pulses per PWM	1
MEM_POS_LSW	O51 Overlapped space loop memory lsw	electrical pulses	1
MEM_POS_MSW	O52 Overlapped space loop memory msw	electrical pulses	1
INCR_SIN	O53 Incremental SIN theta Sin/Cos Encoder		1
INCR_COS	O54 Incremental COS theta Sin/Cos Encoder		1
INIT_RESET	O55 Initial reset ended		1
PTM_TH_PRB	O56 PTM motor thermal probe		40.96
PTR_TH_PRB	O57 PTR radiator thermal probe		40.96
SENS_PULSES_RD	O58 Pulses read by sensor		1
PRC_SENS2_SPD	O59 SENS2 Rotation speed not filtered	% MOT_SPD_MAX	163.84
ACT_SENS2_POS	O60 SENS2 Actual position		327.67
SENS2_SIN	O61 SENS2 Sin_theta		1
SENS2_COS	O62 SENS2 Cos_theta		1
SYNC_DELAY	O63 Delay on SYNC reception		1
SYS_T_MIN	O64 Application minimum torque limit reference value	% MOT_T_NOM	40.96
BRAKE_EN	O65 Energy dissipated on breaking resistance	Joule	1

- **MW105.0 sysExtraInt** , refreshed in the background core task

Name	Description	UM	Scale
BRAKE_R_AD_ENERGY	Dexp00 - Adiabatic Energy dissipated on brake resistance	Joule	1
BRAKE_R_POWER	Dexp01 - Average Power dissipated on brake resistance	Watt	1
AI16	Dexp04 - 16 bit Analog input (optional)	%	163.84
ANYBUS_EN	Dexp05 - Anybus module enabled		1
ANYBUS_STATE	Dexp06 - Anybus module state		1
SW_RESET_CNT	Dexp07 - Software reset occours		1
BO_CAN_MOD	Dexp08 - Bus-off status. If 1 the CAN module is in bus-off status		1
REC_CAN_MOD	Dexp09 - CAN Receive Error Counter		1
TEC_CAN_MOD	Dexp10 - CAN Transmit Error Counter		1
STATE_SM	Dexp11 - Actual states of the State Machine		1
I_LOOP_BAND	Dexp23 - Current loop bandwidth	Hz	1
TEST_CONN_FEEDBACK	Dexp24 - Connection tests feedback		1
SYNC_DELAY	Dexp25 - Delay from SYNC reception to Speed routine execution	us	1
PWM_SYNC_OFFSET	Dexp26 - PWM offset for SYNC delay control	pulses	1
MOT_T_NOM	Dexp30 - Nominal motor torque	Nm	1
MOT_P_NOM	Dexp31 - Nominal motor power	KW	10
PWM_COUNTER	Dexp33 - ISR counter		1
SPD_ISR	Dexp34 - Speed routine duration	us	64
I_ISR	Dexp35 - Current routine duration	us	64
APP_ISR	Dexp36 - Application fast task duration	us	64
APP_AVBLE_ISR	Dexp37 - Application fast task available time	us	64
DRV_F_PWM_MAX	Dexp38 - Max PWM frequency available	Hz	1
MOTOR_SENSOR_RES	Dexp39 – Motor sensor resolution	bit	1
SENS2_RES	Dexp40 – Second sensor resolution	bit	1
SPD_LOOP_BW_MAX	Dexp41 – Max speed loop bandwidth	Hz	10
MOT_POS	Dexp42-D43 Actual motor position	pulses	1
SENS2_POS	Dexp44-D45 Second sensor position	pulses	1
TEST_CONN_RES_RATIO	Dexp48 - Connection test Motor and Sensor pole ratio		100
ANYBUS_EXCP_CODE	Dexp50 - Anybus Exception Code		1
APP_CYCLIC_ISR	Dexp54 - Application cyclic task duration	us	64
FACTORY_FW_REV	Dexp55 - Factory Firmware revision		256
MOT_F_MAX	Dexp57 - Motor max frequency	Hz	10
ISR_PWM	Dexp61 - Control Routines Frequency	Hz	1
IGBT_PWM	Dexp62 - IGBT Frequency	Hz	1
DC_BUS_RIPPLE	Dexp63 - DC Bus Ripple at 100Hz	Volt rms	16

- **MW106.0 sysCommands**

Name	Description	UM	Scale
EN_SENSOR2_TUNE	U00 - Enable second sensor auto-tune		1
EN_TEST_SPD	U01 – Enable speed test		1
SPD_REG_SETTING	U02 – Speed regulator autosetting		1
MAPPING_CONFIG	U03 – Select the mapping configuration (Canopen & EtherCAT)		1
EN_SENSOR_TUNE	U04 – Enable motor sensor auto-tune		1
EN_START_UP_APPL	U05 – Enable Quick Start Application		1
START_UP_SPD_SEL	U06 – Quick Start Application Speed Reference Selection		1
START_UP_RUN_SEL	U07 – Quick Start Application Run Command Selection		1
START_UP_EN_REF	U08 - Quick Start Application Enable Reference		1
START_UP_EN_LIN_RAMP	U09 - Quick Start Application Linear Ramps Enables		1
EN_I_VECTOR	U10 – Enable Current Vector for Power Part Test		1
I_VECTOR_FREQ	U11 – Current Vector Frequency for Power Part Test		1

1.8 GROUP = "SYSTEM VARIABLES"

Type	Index	Name	Elements		R/W	Description
Memory	MD1000.0	sysHiResTimer	1	UDINT	R	Free-running Timer with resolution of 6.6ns
Memory	MD1001.0	sysTimer	1	UDINT	R	Free-running Timer with resolution of 1ms
Memory	MD1001.1	sysFastPlcMaxDuration	1	UDINT	R	Maximum PLC fast routine execution time measured in unit of 6.6ns
Memory	MD1001.2	sysFastPlcDuration	1	UDINT	R	Actual PLC fast routine execution time in unit of 6.6ns
Memory	MD1001.3	sysSlowPlcMaxDuration	1	UDINT	R	Maximum PLC slow routine execution time measured in unit of 6.6ns
Memory	MD1001.4	sysSlowPlcDuration	1	UDINT	R	Actual PLC slow routine execution time in unit of 6.6ns
Memory	MD1001.5	sysCyclicPlcMaxDuration	1	UDINT	R	Maximum PLC cyclic routine execution time measured in unit of 6.6ns
Memory	MD1001.6	sysCyclicPlcDuration	1	UDINT	R	Actual PLC cyclic routine execution time in unit of 6.6ns
Memory	MD1002.0	sysFastPlcTimeLimit	1	UDINT	R	Maximum PLC fast routine execution time admitted in unit of 6.6ns
Memory	MD1003.0	sysTimerCapture	1	INT	R	Free-running Timer capture with resolution of 6.6ns
Memory	MD1004.0	sysCyclicPeriod	1	REAL	R/W	Desidered Cyclic Task Period [ms]
Memory	MD1005.0	sysTrueCyclicPeriod	1	REAL	R	True Cyclic Task Period [ms]
Memory	MW1006.0	sysUserDbSync	48	UINT	R/W	Application Process Image words

1.9 GROUP = “CAN OPEN VARIABLES”

Type	Index	Name	Elements		R/W	Description
Memory	MD1020.0	sysTimerSyncCont	1	UDINT	R	SYNC free-running timer with resolution equals to Baud rate period (1µs at 1Mbit/s)
Memory	MD1021.0	sysTimerSyncRec	1	UDINT	R	SYNC free-running timer freezed at SYNC reception
Memory	MD1022.0	sysFlagSyncRec	1	BOOL	R	SYNC signal received
Memory	MW1023.0	sysPdoCustom	1	INT	R/W	Enable custom PDO sending on 4 less significant bit
Memory	MW1024.0	sysFieldbusProblem	1	INT	R/W	Set to 1 from internal CAN routine when a Life time error occours
Memory	MD1025.0	sysSyncPeriod	1	DINT	R	SYNC period set via CAN with object Idx=1006 Sub=0
Memory	MW1026.0	sysNmtState	1	INT	R	NMT state

1.10 GROUP = “MODBUS VARIABLES”

Type	Index	Name	Elements		R/W	Description
Memory	MW1050.0	sysModbusIndex	1	INT	R/W	Register Address managed via Modbus
Memory	MW1050.1	sysModbusValue	1	INT	R/W	Register Value managed via Modbus
Memory	MX1050.2	sysModbusdirection	1	INT	R/W	Modbus message direction Read = FALSE; Write = TRUE
Memory	MD1051.0	sysModbusInputs	1	UDINT	R	32 logical input functions state from Modbus

1.11 GROUP = “FIELDBUS VARIABLES”

Type	Index	Name	Elements		R/W	Description
Memory	MD1070.0	sysFieldbusTorqueRef	1	REAL	R	Fieldbus Torque reference referred to MOT_T_NOM
Memory	MD1070.1	sysFieldbusMaxTorque	1	REAL	R	Symmetrical Fieldbus Torque limit referred to MOT_T_NOM
Memory	MD1070.2	sysFieldbusSpeedPercRef	1	REAL	R	Fieldbus Speed reference referred to MOT_SPD_MAX (P65)
Memory	MD1070.3	sysFieldbusSpeedRefPulses	1	DINT	R	Fieldbus Speed reference in pulses per PWM period [65536=1 mechanical turn]
Memory	MD1070.4	sysFieldbusMaxPositiveTorque	1	REAL	R	Positive Fieldbus Torque limit referred to MOT_T_NOM
Memory	MD1070.5	sysFieldbusMaxNegativeTorque	1	REAL	R	Negative Fieldbus Torque limit referred to MOT_T_NOM
Memory	MD1071.0	systabProcessData	ARRAY [40]	UINT	RW	Mapping data for Profibus and Anybus
Memory	MD1072.0	sysFieldbusInputs	1	UDINT	R	32 logical input functions state from fieldbus

1.12 GROUP = “SPI VARIABLES”

Type	Index	Name	Elements		R/W	Description
Memory	MW1080.0	sysEnSpiMaster	1	BOOL	R/W	Configures the drive in SPI Master mode Disable = FALSE; Enable = TRUE
Memory	MW1081.0	sysEnSpiSlave	1	BOOL	R/W	Configures the drive in SPI Slave mode Disable = FALSE; Enable = TRUE
Memory	MX1082.0	sysSpiNWord	1	INT	R/W	Transmitted and received words Range from 0 to 4 words (a word = 16 bit). 0 disable the SPI module
Memory	MX1083.0	sysSpiBaud	1	INT	R/W	SPI Baud rate equals to: 37.5MHz / (sysSpiBaud + 1) Default value =12 → 2.88MHz

2. EMBEDDED BLOCKS

The embedded blocks are firmware functions written in C language that can be used inside PLC applications.

2.1 GROUP = “DRIVE PARAMETERS”

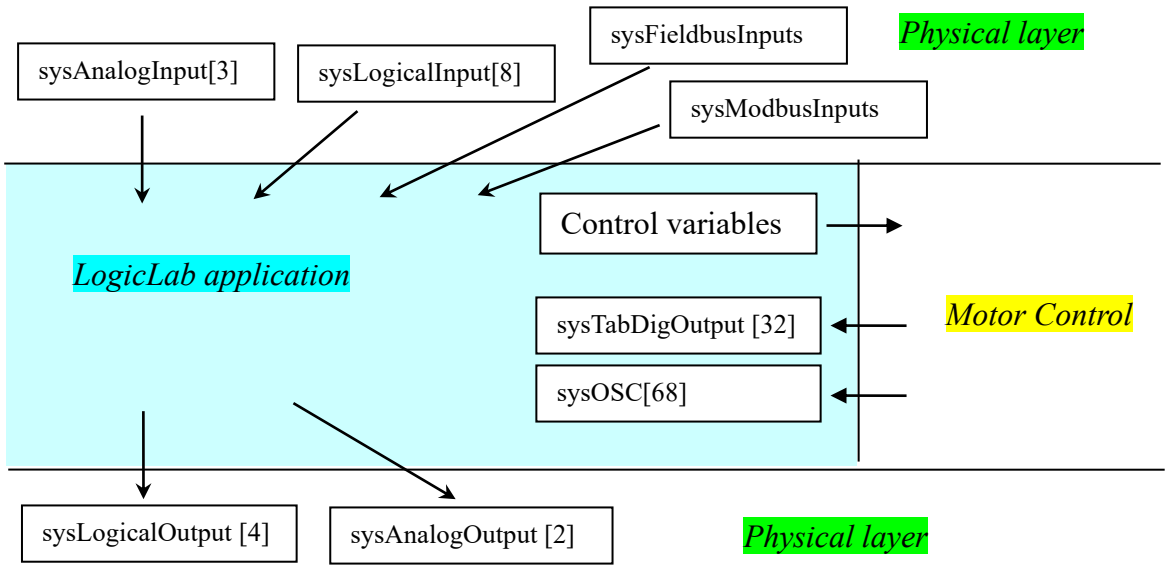
Name	Input Variables			Output		Description
	Name	Descr.	Type	Descr.	Type	
WritePar(index.data)	index	Parameter Table index [0÷199]	INT	Write result TRUE=ok FALSE=no	BOOL	This block can be used to write a standard parameter [0÷199] with key and drive status control.
	data	value	INT			
WriteExtraPar (index,data)	index	Extra Parameter Table index [0÷99]	INT	Write result TRUE=ok FALSE=no	BOOL	This block can be used to write an extra parameter [0÷99] with key and drive status control.
	data	value	INT			
WriteExtraParDINT (index,data)	index	Extra Parameter Table index [0÷98]	DINT	Write result TRUE=ok FALSE=no	BOOL	This block can be used to write a 32 bit extra parameter [0÷98] with key and drive status control.
	data	valore	DINT			
WriteCon(index,data)	index	Connection Table index [0÷99]	INT	Write result TRUE=ok FALSE=no	BOOL	This block can be used to write a standard conection [0÷199] with key and drive status control.
	data	valore	INT			
WriteInt (index,data)	index	Application internal value index [64÷127]	INT	Write result TRUE=ok FALSE=no	BOOL	This block is usually used into SLOW task to refresh an application internal value [64÷127].
	data	value	INT			
WriteOutDig (index,data)	index	Logic Output index [0÷3]	INT	Write result TRUE=ok FALSE=no	BOOL	This block is used to change immediately the state of one digital output
	data	Value to set	BOOL			

2.2 I/O MANAGEMENT

There are 2 way to manage digital and analog reference from application poit of view:

1. Read and write directly physical data. This approach is very easy and give to application maximum freedom degree.
2. Call Embedded block “Standard I/O” from slow or fast PLC task. In this way the application exchange data with physical layer using some data block.

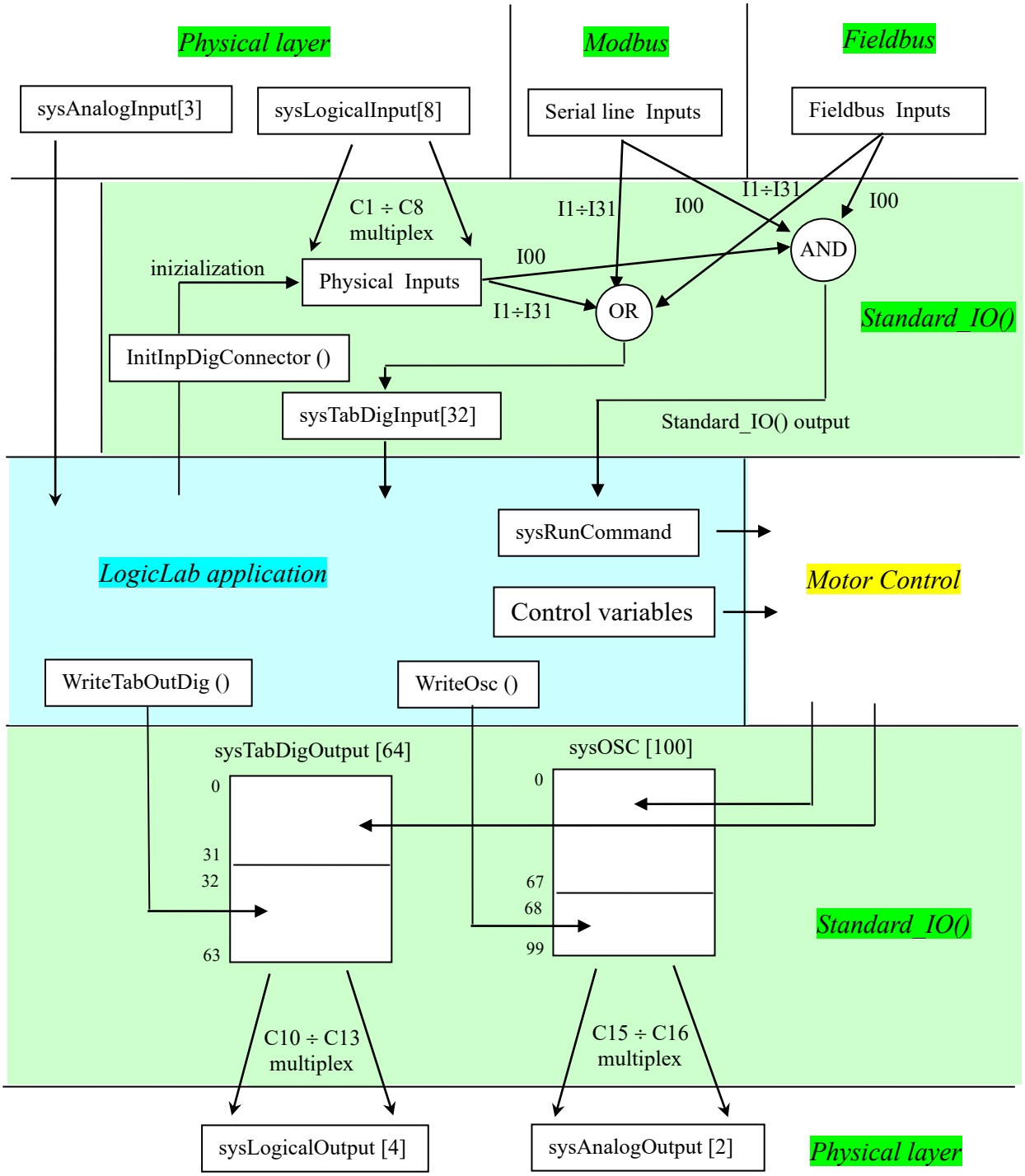
2.2.1 DIRECT CONTROL



LogicLab application can read directly analog, digital, fieldbus amd modbus inputs and can write digital and analog output. In this case the application has the complete control of physical I/O, but this meaning that is disabled multiplexing of digital inputs, digital outputs and analog outputs.

About digital outputs, the Motor Control core changes the first 32 digital output function of data block “sysTabDigOutput” and application can use this information to change the physical logic outputs with data block “sysLogicalOutput”.
About analogl outputs, the Motor Control core changes the first 68 monitor value of data block “sysOSC” and application can use this information to change the physical analog outputs with data block “sysAnalogOutput”.

2.2.2 GROUP = “STANDARD I/O”



Name	Description					
Standard_IO ()	This embedded block manages in parallel way physical inputs (multiplexed with connection C1÷C8), serial line inputs and fieldbus inputs and produce the output function logics into data block “sysTabDigInput”. Standard_IO deals of logical output multiplexing (with connection C10÷C13) starting from data block “sysTabDigOutput” and analog output multiplexing (with connection C15÷C16) starting from data block “sysOSC”. Standard_IO boolean output is the logic AND of run command logic function I00 from connector, serial line and fieldbus. Usually this output is give to motor run command in this way: sysRunCommand := Standard_IO();					
InitInpDigConnector (data)	data	Logical input function index [0÷31]	INT	Inizialization result TRUE=ok FALSE=no	BOOL	With this function it’s possible to set a logical input function. (“sysTabDigInput”), usually in BOOT task. This value is kept if this logical function isn’t configured in any physical input. It’s mandatory to use it in conjuction with Standard_IO function

Name	Input Variables			Output		Description
	Name	Descr.	Type	Descr.	Type	
WriteOsc (index,data)	index	Indice grandezza monitor extra [68÷99]	INT	Write result TRUE=ok FALSE=no	BOOL	This block is used to refresh an application monitor value with index [68÷99]. Usually this function is called in FAST task. It’s mandatory to use it in conjuction with Standard_IO function
	data	value	INT			
WriteTabOutDig (index,data)	index	Output logic function index [32÷63]	INT	Write result TRUE=ok FALSE=no	BOOL	This block is used to change the state of one application digital output function (index [32÷63]. If this logical output function is configured on one physical output, this will be changed after an half PWM period. It’s mandatory to use this block in conjuction with Standard_IO function
	data	value	INT			

2.3 GROUP = “PERMANENT MEMORY”

Name	Input Variables			Output		Description
	Name	Descr.	Type	Descr.	Type	
ReadApplData (address, n_word, index)	address	Target address	DINT	Read result 0 = ok 1 = EEPROM alarm 2 = index too big 3 = n_word too big 4 = n_word <0	INT	With this function it’s possible to read up to 30Kapplication word stored in permanent memory and copy them at desidered address. This function has to be call from Slow task.
	n_word	Word to read	INT			
	index	EEPROM memory index [0÷29999]	INT			

WriteApplData (address, n_word, index)	address	Starting address	DINT	Write result 0 = ok 1 = EEPROM alarm 2 = index too big 3 = n_word too big 4 = n_word <0 5 = too much repetitive write	INT	With this function it's possible to write up to 30Kapplication word in permanent memory from desidered address. This function has to be call from Slow task. **NB**: every 7 word takes about 5-10ms to be written, so slow task will be stopped for this time
	n_word	Word to write	INT			
	index	EEPROM memory index [0÷29999]	INT			

2.4 GROUP = "GENERIC FUNCTIONS"

Name	Input Variables			Output		Description
	Name	Descr.	Type	Descr.	Type	
Setbit (data, index, status)	data	word	INT	Word with bit modified	INT	Allows to set one bit of input word
	index	bit index	INT			
	status	bit value	BOOL			
Getbit (data, index)	data	word	INT	Bit status	BOOL	Allows to test one bit status of input word
	index	bit index	INT			
WriteApplAll (code)	code	Application alarm code A04 [0÷3]	INT	TRUE	BOOL	Activation of Application alarm with desidere code [0÷3]
WriteApplRev(num)	num	Application revision number	REAL	TRUE	BOOL	Application revision number showed in d61 with one decimal number.
SetRamps(speed)	speed	Value forced into memory ramps referred to MOT_SPD_MAX	REAL	TRUE	BOOL	With this block it's possible to set memory ramps and S-ramps with desidered value, percent of MOT_SPD_MAX
ChangeNmax(newmax)	newmax	New MOT_SPD_MAX	INT	Write result TRUE=ok FALSE=no	BOOL	With this block it's possible to change on-line motor maximum speed (P65) with memory ramps ricalculation

2.5 GROUP = "CANOPEN FUNCTIONS"

Name	Input			Output		Description
	Name	Descr.	Type	Descr.	Type	
new_object (address, type, size, index)	address	New object variable address	DINT	1 = ok 2 = generic error 16 = space memory complete	INT	This function creates a new object dictionary entry of the Manufacturer Specific Profile Area or Standardised Device Profile Area (see CiA DS301 CANOpen). **It must be used into BOOT or SLOW tasks.**
	type	Object features	UINT			
	size	Variable length, in word (16 bit). ONLY DOMAIN OBJECT	UINT			
	index	Object Index	UINT			

new_object function realizes new entries in the Dictionary Object Can Open. The objects can have the following general structure (for more information, see CiA Draft Standard 301): VAR, ARRAY and DOMAIN; with the following data types: INTEGER8, INTEGER16, INTEGER 32, UNSIGNED8, UNSIGNED 16, UNSIGNED32. It is possible to define the object attributes like the access type (READ, WRITE).
new_object can create up to 100 Dictionary Objects; the indexes are included in the range from 2000 h to A000 h. Pay attention that the indexes from 2000 h to 2024 h are used for old Open Drive Dictionary (for more information, see CANOpen OPD Documentation). It is possible to rewrite these objects in the BOOT task.

new_object have four input parameters:

- New object variable address. The address must be a 32 bit integer value (Use the function ADR(Variable Name) to pass the variable address);
- Object features. It defines the General Structure, the Data Type and the Access Type.
- Variable length. It defines the dimension of the DOMAIN structure.
- Object index. New object can have indexes from 2000 h to A000 h to create objects in Manufacturer Specific Profile Area and Standardised Device Profile Area.

Use the following tables to define new object features.

Data Types (bit)	Data Type Code	General Structure	Structure Code	Data Length
INTEGER8 (8)	8	VAR	0	N.U. (*)
		ARRAY(***)	1 - 255	N.U. (*)
UNSIGNED8 (8)	8	VAR	0	N.U. (*)
		ARRAY	1 - 255	N.U. (*)
INTEGER8 (8)	9	VAR	0	N.U. (*)
		ARRAY DS301(***)	1 - 255	N.U. (*)
UNSIGNED8 (8)	9	VAR	0	N.U. (*)
		ARRAY DS301	1 - 255	N.U. (*)
INTEGER16 (16)	16	VAR	0	N.U. (*)
		ARRAY	1 - 255	N.U. (*)
UNSIGNED16 (16)	16	VAR	0	N.U. (*)
		ARRAY	1 - 255	N.U. (*)
INTEGER16 (16)	17	VAR	0	N.U. (*)
		ARRAY DS301	1 - 255	N.U. (*)
UNSIGNED16 (16)	17	VAR	0	N.U. (*)
		ARRAY DS301	1 - 255	N.U. (*)
INTEGER32 (32)	32	VAR	0	N.U. (*)
		ARRAY	1 - 255	N.U. (*)
UNSIGNED32 (32)	32	VAR	0	N.U. (*)
		ARRAY	1 - 255	N.U. (*)
INTEGER32 (32)	33	VAR	0	N.U. (*)
		ARRAY DS301	1 - 255	N.U. (*)
UNSIGNED32 (32)	33	VAR	0	N.U. (*)
		ARRAY DS301	1 - 255	N.U. (*)
DOMAIN (**)	63	DOMAIN	0	1 - 2047 word

(*) Not Used
(**) DOMAIN Length is defined in size parameter
(***) ARRAY: A multiple data filed object where each data field is a simple variable of the SAME basic data type e.g. array of UNSIGNED16 etc. Sub-index 0 is the first elements of the ARRAY data.
ARRAY DS301: A multiple data filed object where each data field is a simple variable of the SAME basic data type e.g. array of UNSIGNED16 etc. Sub-index 0 is of UNSIGNED8 and therefore not part of the ARRAY data.

	Readable	Writeable	Readable and writeable
Access Type Code	1	2	3

Object Features parameter is shared in three fields. In these fields are inserted the codes to define: General Structure, Data Type and Access Type.

Variable	Object Description															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Fields	Data Length Field								Data Type Field						Access Type Field	

new_object have a output parameter, it indicates the object building. In the following table are descripted the return values.

Return Value	New Object Status	Description
1	CREATED	Object created
2	NOT CREATED	Object creation failed. Check the Object Features parameter. Example: new object can not have Access Type Field equal to zero, the Data Type Filed is wrong or not exist, etc.
16	NOT CREATED	Object creation failed. Space memory complete.
32	NOT CREATED	Error index. The index is too small.
48	NOT CREATED	Error index. The index is too big.

New object example

New object description:

- Readable and writeable;
- ARRAY;
- INTEGER16,
- 10 array elements;
- Index 2020h;
- Variable name Tab_dati_applicazione;

We obtain the codes for **new_object** function from object description:

Readable and writeable => Access Type Field =3
ARRAY => Data Type Field =16
Array containing 10 elements => Data Length Field=10;

Variabile	Object Description															
Fields	Data Length Field								Data Type Field						Access Type Field	
Codes	10								16						3	
N^ Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1	1
Value	**0**				**A**				**4**				**3**			

Object features = 0A43h;
Variable length=0;
Object index=2020h;

new_object function will be defined in the following way:

Rest := **new_object**(ADR(Tab_dati_applicazione),16#0A43,0,16#2020);

Name	Input			Output		Description
	Name	Descr.	Type	Descr.	Type	
Nmt_config(state_config)	state_config	NMT command specifier	UINT	1 = ok 0 = generic error	UINT	This function defines the NMT Services. In other words, it changes the state of the NMT slave (OPD) . **It must be used into BOOT or SLOW tasks.**
Sdo_commun_config (sdo_number, cob_id_client, cob_id_server)	sdo_number	SDO index (values from 0 to 3)	UINT	1 = ok For other values, see Abort SDO Transfer Protocol Table (CiA Draft Standard 301)	DINT	This function defines the Server SDO Parameter. **It must be used into BOOT or SLOW tasks.**
	cob_id_client	COB-ID client	DINT			
	cob_id_server	COB-ID server	DINT			
Rec_pdo_commun_conf (pdo_number, cob_id_pdo, transmission_type)	pdo_number	Receive PDO Communication Index (values from 0 to 3)	UINT	1 = ok For other values, see Abort SDO Transfer Protocol Table (CiA Draft Standard 301)	DINT	This function defines the Receive PDO Communication Parameter. **It must be used into BOOT or SLOW tasks.**
	cob_id_pdo	COB-ID PDO	DINT			
	transmission_type	Transmission type	UINT			
Rec_pdo_mapping_conf (pdo_number, index_map, sub_index_map)	pdo_number	Receive PDO Mapping Index (values from 0 to 3)	UINT	1 = ok For other values, see Abort SDO Transfer Protocol Table (CiA Draft Standard 301)	DINT	This function defines the Receive PDO Mapping Parameter. **It must be used into BOOT or SLOW tasks.**
	index_map	Object mapping indexes (address of a array containing 8 elements)	DINT			
	sub_index_map	Object mapping Sub-indexes (address of a array containing 8 elements)	DINT			
Tra_pdo_commun_conf (pdo_number, cob_id_pdo, transmission_type,	pdo_number	Transmit PDO Communication Index (values from 0 to 3)	UINT	1 = ok For other values, see Abort SDO Transfer Protocol	DINT	This function defines the Transmit PDO Communication

inhibit_time, event_time)	cob_id_pdo	COB-ID PDO	DINT	Table (CiA Draft Standard 301)		Parameter. **It must be used into BOOT or SLOW tasks.**
	Transmission_type	Transmission type	UINT			
	Inhibit_time	Inhibit time	UINT			
	Event_timer	Event timer	UINT			
Tra_pdo_mapping_conf (pdo_number, index_map, sub_index_map)	pdo_number	Transmit PDO Mapping Index (values from 0 to 3)	UINT	1 = ok For other values, see Abort SDO Transfer Protocol Table (CiA Draft Standard 301)	DINT	This function defines the Transmit PDO Mapping Parameter. **It must be used into BOOT or SLOW tasks.**
	Index_map	Object mapping indexes (address of a array containing 8 elements)	DINT			
	Sub_nde x_map	Object mapping Sub-indexes (address of a array containing 8 elements)	DINT			

The functions in the table are used for Server SDO Parameter configuration, Receive PDO Communication Parameter configuration, Receive PDO Mapping Parameter configuration, Transmit PDO Communication Parameter configuration and Transmit PDO Mapping Parameter configuration from LogicLab. Moreover it is possible to manage NMT protocol to change the state of the NMT slave (OPD).

State of the NMT Slave Configuration Example

In order to change the state of the NMT Slave (OPD EXP) is introduced the **NMT_config** function.
The state is changed by the NMT command specifier of the NMT protocol (for more information, see CiA Draft Standard 301 protocol).

NMT Slave Configuration:
NMT Slave (OPD EXP) in the OPERATIONAL State.

NMT Command Specifiers:

CS = 1	=>	START, **NMT_config** function sets the state of the OPD EXP to OPERATINAL
CS = 2	=>	STOP, **NMT_config** function sets the state of the OPD EXP to STOPPED
CS = 128	=>	enter PRE-OPERATIONAL, **NMT_config** function sets the state of the OPD EXP to PRE-OPERATIONAL
CS = 129	=>	RESET APPLICATION, **NMT_config** function sets the state of the OPD EXP from any state to the “reset application” sub-state
CS = 130	=>	RESET COMMUNICATION, **NMT_config** function sets the state of the OPD EXP from any state to the “reset communication” sub-state

NMT _config function will be defined in the following way:

Rest := **NMT _config(**1**)**

Server SDO Parameter Configuration Example

In order to describe the SDOs used on the OPD EXP is introduced the **Sdo_commun_config** function.
Server SDO Parameter Configuration:
2nd Server SDO parameter (1201 h object index)
COB-ID Client->Server (rx) = 610 h
COB-ID Server->Client (tx) = 590 h

Sdo_commun_config function will be defined in the following way:

Rest := **Sdo_commun_config(**1, 16#0610, 16#0590**)**

Transmit PDO Communication Parameter Configuration Example

In order to set the TPDOs communication parameter used on the OPD EXP is introduced the **Tra_pdo_commun_conf** function.
Transmit PDO Communication Parameter Configuration:
3rd Transmit PDO Communication Parameter (1802 h object index)
TPDO COB-ID = 210 h
Transmission type = PDO is transmitted on an event time (code=FE h)
Inhibit time = no inhibit time
Event timer = 100 ms

Tra_pdo_commun_conf function will be defined in the following way:

Rest := **Tra_pdo_commun_conf (**2, 16#0210, 16#00FE, 0, 16#0064**)**

Transmit PDO Mapping Parameter Configuration Example

In order to set the TPDOs mapping parameter used on the OPD EXP is introduced the **Tra_pdo_mapping_conf** function.
Transmit PDO Mapping Parameter Configuration:
4rd Transmit PDO Mapping Parameter (1A03 h object index)
Mapped objects of the Manufacturer Specific Profile Area (see lower table)
=> Speed reference object (index=201C h)
=> Speed reference object (index=201D h)
=> 4th element of the application area array (index=201E h, sub-index=4)
=> Logical output (index=201F h)
=> OPD status (index=2020 h)

Index (hex)	Structure	Type	Name	Description	Access
201C	VAR	INTEGER16	Speed_ref	Speed reference	rw
201D	VAR	INTEGER16	Torque_ref	Torque refernce	rw
201E	ARRAY	INTEGER16	Application_Tab[100]	Application Area	rw
201F	VAR	INTEGER8	Logic_Out	Logical output	rw
2020	VAR	UNSIGNED8	OPD_Status	OPD status	r

In LogicLab, we create two array containing 8 elements (PAY ATTENTION, the array must be declared as global variables):
DINT index[8];
DINT sub_index[8];

The arrays must be defined in this way:

Nr	index
0	201C
1	201D
2	201E
3	201F
4	2020
5	0
6	0
7	0

Nr	sub_index
0	0
1	0
2	4
3	0
4	0
5	0
6	0
7	0

Tra_pdo_ mapping _conf function will be defined in the following way:

Rest := **Tra_pdo_ mapping _conf (**3, ADR(index), ADR(sub_index)**)**

Name	Input			Output		Description
	Name	Desc	Type	Desc	Type	
ErrorFieldEmcy(data_error, n_byte)	data_error	Error field definition	DINT	1 = ok 0 = generic error	UINT	This function configures the last three byte of the Manufacturer Specific Error Field. **It must be used into BOOT or SLOW tasks.**
	n_byte	Error field byte used	UINT			

ErrorFieldEmcy function configures the last three byte of the Manufacturer Specific Error Field.

Manufacturer Specific Error Field Configuration Example

Manufacturer Specific Error Field Configuration: set 2 bytes with the code_alarm value.

ErrorFieldEmcy function will be defined in the following way:

Rest := **ErrorFieldEmcy(**code_alarm, 2**)**

2.6 GROUP = “FREQUENCY INPUT”

Name	Input			Output		Description
	Name	Desc	Type	Desc	Type	
Time_Pulses (type)	type	If type is TRUE, pulses counted are always reported to the 4 edges counted with an encoder input (C9=1) So pulses from frequency/sign (C9=2) are multiplied by 2 and frequency/sign 1 edge counted (C9=3) are multiplied by 4	BOOL	Average pulses	REAL	This function computes the average pulses received in the last PWM period.

Capture_Digit_In1 (addr_time_base_capt, addr_elapsed_time, edge) – input 6 **Capture_Digit_In2** (addr_time_base_capt, addr_elapsed_time, edge) – input 7 **Capture_Digit_In3** (addr_time_base_capt, addr_elapsed_time, edge) – input 8	addr_time_base_capt	Capture timer pointer "sysTimerCapture"	DINT	See Table 1	UINT	This function analysis the edge of a digital signal with frequency smaller than 2 * FPWM. **It must be used into FAST task**
	Addr_elapsed_time	Elapsed time pointer (unit=6.6ns)	DINT			
	edge	Active edge	INT			

Capture_Digit_Inx function captures the edge of a digital signal with frequency smaller than two times PWM frequency. This function can capture up to 4 events in the PWM period. The pulses with time duration lower than 26 µs are neglected.
The function is formed from two pointers and a active edge parameter:

- Capture timer pointer. This pointer returns the event timer (in CPU clock cycles).
- Elapsed time pointer. This pointer returns the elapsed time between two happened events (in CPU clock cycles).
- Active edge. This parameter sets the active edge for the signal analysis (Active Rising Edge or Active Falling Edge).

Moreover the function has an output parameter for events description (see Table 1).
If the event is a pulse, the "elapsed time" is the time duration, whereas if the event is an edge, the "elapsed time" is the elapsed time between PWM period start and event time (see Figure 1).

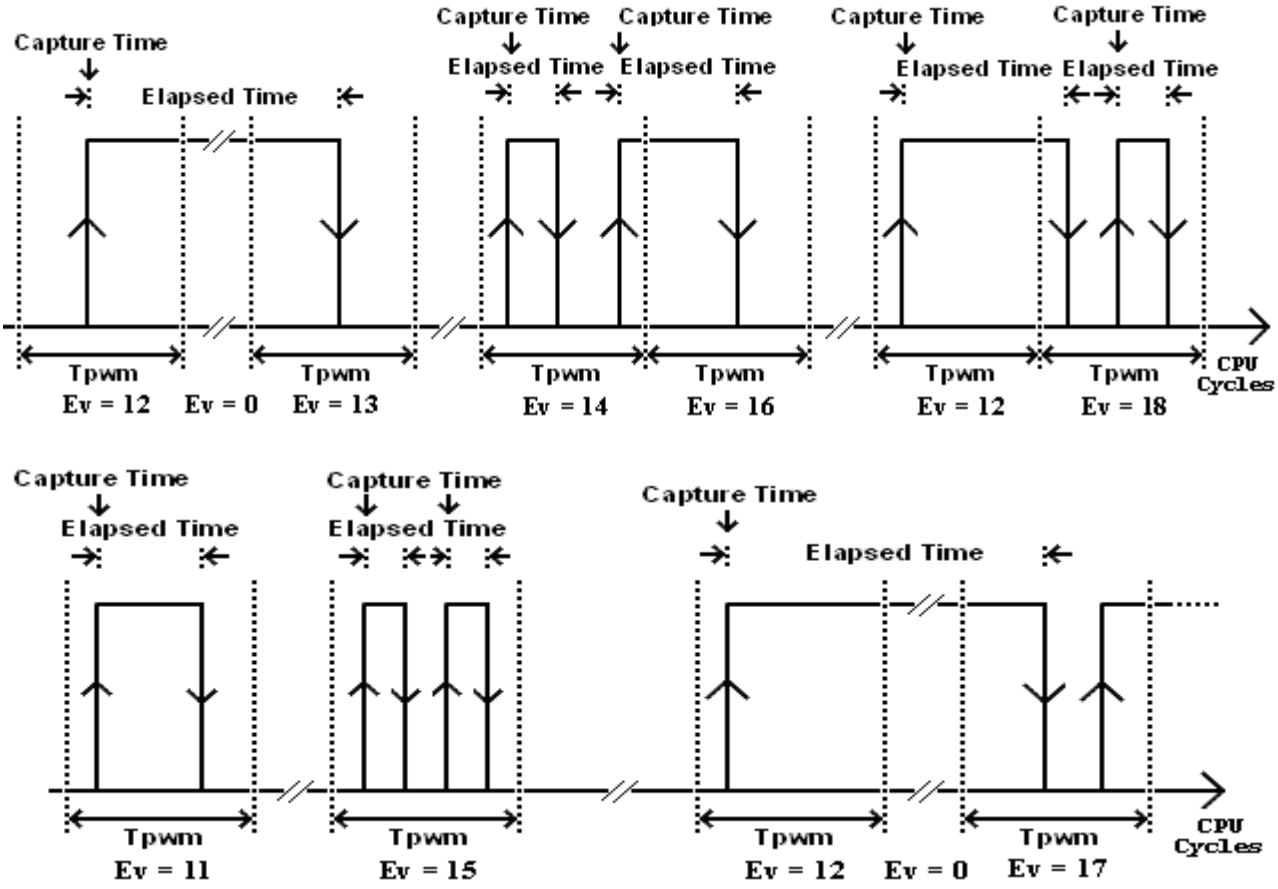
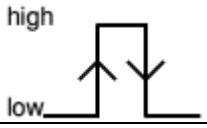
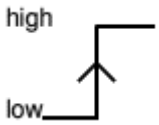
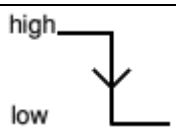
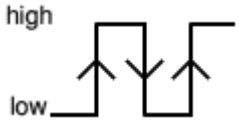
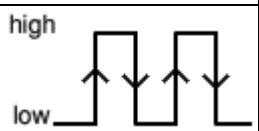
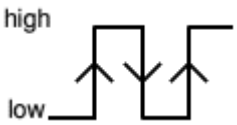
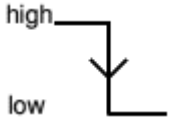
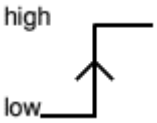
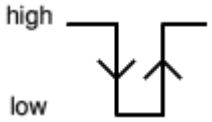
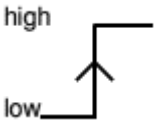
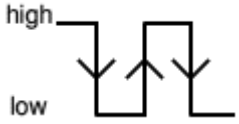
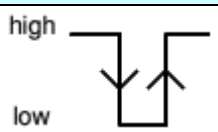
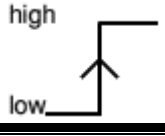


Figure 1. Capture_Digit_In function working

Return Value	Event	Event happened in the previous PWM period	Event happened in the current PWM period	Description
0	-	-	-	No event
Rising Edge Active				
11	Positive pulse	-		Positive pulse in the current PWM period
12	Rising edge	-		Rising edge in the current PWM period.
13	Falling edge	-		Falling edge in the current PWM period.
14	Positive pulses + Rising edge	-		Positive pulse and a rising edge in the current PWM period.
15	Two positive pulses	-		Two positive pulses in the current PWM period.
16	Falling edge			Falling edge in the current PWM period, whereas in the previous PWM the function have found a 14 event type.
17	Negative pulse			Negative pulse in the current PWM period, whereas in the previous PWM the function have found a rising edge.
18	Falling edge + Positive pulse			Falling edge and a positive pulse in the current PWM period, whereas in the previous PWM the function have found a rising edge
Falling Edge Active				
21	Negative pulse	-		Negative pulse in the current PWM period
22	Falling edge	-		Falling edge in the current PWM period.
23	Rising edge	-		Rising edge in the current PWM period.

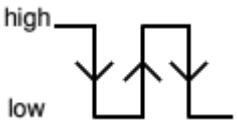
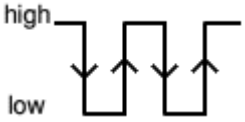
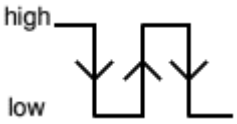
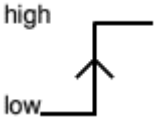
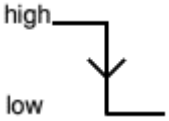
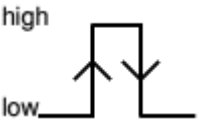
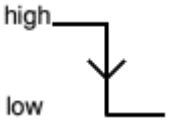
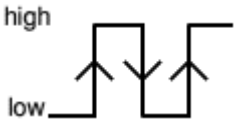
24	Negative pulse + Falling edge	-		Negative pulse and fall edge in the current PWM period.
25	Two negative pulses	-		Two negative pulses in the current PWM period.
26	Rising edge			Rising edge in the current PWM period, whereas in the previous PWM the function have found a 24 event type.
27	Positive pulse			Positive pulse in the current PWM period, whereas in the previous PWM the function have found a falling edge.
28	Rising edge + negative pulse			Rising edge and negative pulse in the current PWM period, whereas in the previous PWM the function have found a falling edge

Table 1. Capture_Digit_Inx function return value

The active edge is the first event took in PWM period. It is the rising edge if active edge parameter is equal to '0', whereas is falling edge if active edge parameter is equal to '1'.

Capture_Digit_Inx Example

return_cap=**Capture_Digit_In1**(ADR(event_time),ADR(elapsed_time), edge)

Name	Input			Output		Description
	Name	Desc	Type	Desc	Type	
Capture_Edge_In1 (addr_time_base_capt, addr_elapsed_time, edge, width) – input 6 **Capture_Edge_In2** (addr_time_base_capt, addr_elapsed_time, edge, width) – input 7 **Capture_Edge_In3** (addr_time_base_capt, addr_elapsed_time, edge width) – input 8	addr_time_base_capt	Capture timer pointer "sysTimerCapture"	DINT	See Table 2	UINT	This function captures the digital signal edge. **It must be used into FAST task**
	Addr_elapsed_time	Elapsed time pointer (unit=6.6ns)	DINT			
	edge	Active edge	INT			
	width	Minimum pulse time (in us)				

Capture_Edge_Inx function captures the dominant edge of a digital signal. This function returns the event time and the elapsed time between event time and PWM period end (see Figure 2). The pulses with time duration lower than "Minimum pulse time" will be neglected. If the elapsed time between event time and PWM period end is lower to "Minimum pulse time", the function waits the next PWM period before to define the event as impulse noise.
The function is formed from two pointers, an active edge parameter and a minimum pulse time:

- Capture timer pointer. This pointer returns the event timer (in CPU clock cycles).
- Elapsed time pointer. This pointer returns the elapsed time between two happened events (in CPU clock cycles).
- Active edge. This parameter sets the active edge for the signal analysis (Rising Edge Active or Falling Edge Active).
- Minimum pulse time. It defines the minimum time duration for the impulse (in µs). The permitted maximum value is 200 µs.

Moreover the function has an out parameter for events description (see Table 2).

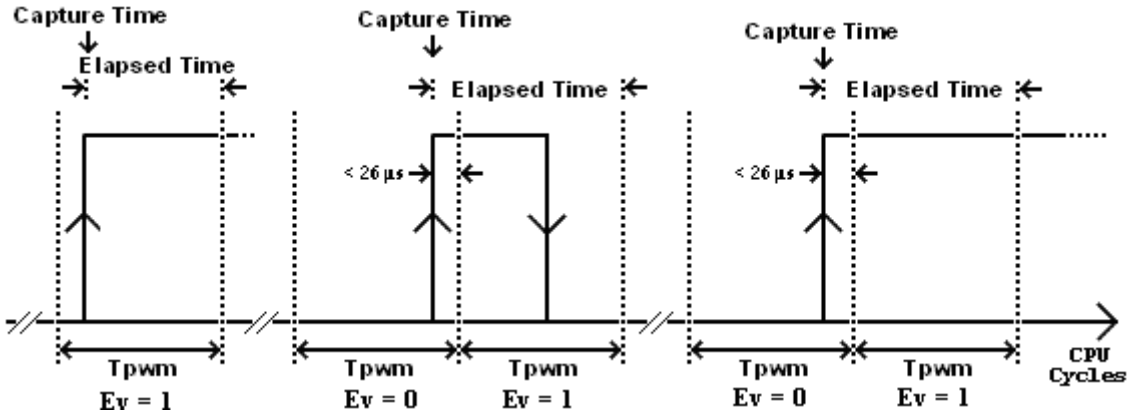
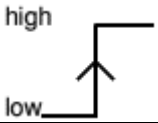
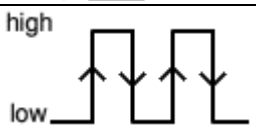
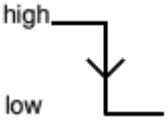
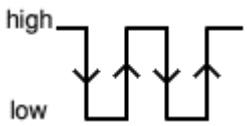


Figure 2. Capture_Edge_Inx function working. In this figure “Minimum pulse time” is equal to 26 µs

Return Value	Event	Event happened in the current PWM period	Description
0	-	-	No event
Rising Edge Active			
1	Rising edge		Rising edge in the current PWM period.
2	Two positive pulses		Two positive pulses in the current PWM period.
Falling Edge Active			
10	Falling edge		Falling edge in the current PWM period.
20	Two negative pulses		Two negative pulses in the current PWM period.

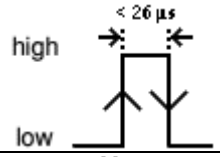
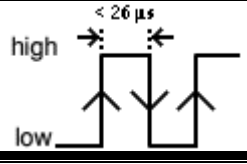
Noises found			
100	Noise		Pulses with time duration lower than “Minimum pulse time”.
100	Noise		Pulses with time duration lower than “Minimum pulse time” (in 14 or 24 event type, see Table 1).

Table 2. Capture_Edge_Inx function return value

Capture_Edge_Inx Example

return_cap=**Capture_Edge_In1**(ADR(event_time),ADR(elapsed_time), edge, min_pulse_time)

In event_time is inserted the timer value when the event is happened, in CPU cycles, for example

event_time =5612144 (CPU cycles)

In milliseconds:

event_time_ms = event_time / DSP_Frequency =
= 5612144 / 150*10^6 = 37.4142 ms

Pay attention which the CPU timer is a free-running 32 bit register, therefore the maximum value is 2^32-1= 4294967295. In elapsed_time is inserted the elapsed time between event time and PWM period. elapsed_time maximum value is always lower than 400 µs.

2.7 GROUP = “SPI FUNCTIONS”

Name	Input			Output		Description
	Name	Desc	Type	Desc	Type	
SendDataSPI (addr_data_TX)	addr_data_TX	Transmitted Data	DINT	TRUE=data transmitted FALSE=data not transmitted. SPI module disabled	BOOL	In Slave mode data are ready to be sent, in Master mode this function sends the SPI data. **It must be used into FAST task**
ReceiveDataSPI (addr_data_TX)	addr_data_RX	Received Data	DINT	0 = data not received 1 = data received 2 = error. Wrong CRC has been detected. 3 = error. The SPI Module has received too many data.	UINT	This function receives the SPI data. **It must be used into FAST task**

2.8 GROUP= “MATHEMATICS”

Name	Input			Output		Description
	Name	Desc	Type	Desc	Type	
Scale64Bit (in,mul,div,add_RES,add_CRY)	in	input	DINT	TRUE=scaling executed without overflow in the output FALSE=output overflow	BOOL	This function execute the input scaling with 64bits calculation RES=in x mul/div RES is the output in 32bits and CRY is the carry in 32 bits.
	mul	Multiplication factor	DINT			
	div	Divisor factor	DINT			
	add_RES	Address Result	DINT			
	add_CRY	Address Carry	DINT			

2.9 GROUP= “SECONDARY CAN BUS”

Name	Input			Output		Description
	Name	Descr.	Type	Descr.	Type	
CAN2_Init (Baud, RxCobId1, RxCobId2)	Baud	CAN bus baudrate code	UINT	TRUE=CAN bus configured FALSE=error configuring the bus	BOOL	This function initializes the secondary CAN bus interface at specified bitrate with 2 receiving mailboxes for specified COB-IDs. **It must be used into BOOT or SLOW tasks.**
	RxCobId1	COB-ID of 1st receive mailbox	UINT			
	RxCobId2	COB-ID of 2nd receive mailbox	UINT			

CAN2_Init function initializes the secondary CAN bus for communication with other drives or devices (like I/O expansion).
The Baud parameter has the same range and coding of connection C48 (CANOPEN_BAUD_SET):

0	1 Mbit/s
1	reserved (1 Mbit/s)
2	500 kbit/s
3	250 kbit/s
4	125 kbit/s
5	100 kbit/s
6	50 kbit/s
7	20 kbit/s

RxCobId1 and RxCobId2 parameters are for COB-ID of the 2 receiving mailbox available: only CAN standard mode (11bit IDs) is supported (not the extended 29bit IDs) so allowed range is 000h-7FFh. The IDs can be the same.

CAN2_Init can be re-issued multiple times during run-time if the receiving mailbox IDs had to be changed. During reconfiguration all pending messages will be dropped.

Name	Input			Output		Description
	Name	Descr.	Type	Descr.	Type	
CAN2_Tx1 (CobId, n_byte, addr_data_TX)	CobId	COB-ID of message to send	UINT	TRUE=previous msg sent and current msg queued into tx mailbox FALSE=previous mgs still unsent	BOOL	This function check if the 1st transmit mailbox is empty and, if so, queue in the new data for transmission.
	n_byte	n. of byte of data to send	INT			
	addr_data_TX	pointer to the buffer containing data to send	DINT			
CAN2_Tx2 (CobId, n_byte, addr_data_TX)	CobId	COB-ID of message to send	UINT	TRUE=previous msg sent and current msg queued into tx mailbox FALSE=previous mgs still unsent	BOOL	This function check if the 2nd transmit mailbox is empty and, if so, queue in the new data for transmission.
	n_byte	n. of byte of data to send	INT			
	addr_data_TX	pointer to the buffer containing data to send	DINT			

CAN2_Tx1 and **CAN2_Tx2** functions are used to send messages into the CAN bus, using 1st and 2nd transmit mailboxes.
These functions don't wait for trasmission completion (as the transmission time is affected by the CAN bitrate, the message data size and the COB-ID priority respect to the packet traffic on the bus) but just queue the new message into the trasmission mailbox (if empty).
Before queueing the new message, the transmission mailbox is checked: if a previous message is still present unsent the function exits returning FALSE.

If n_byte parameter is negative, the function only check for transmission mailbox emptyness without queueing new message: this can be used for polling-check the mailbox.
If n_byte is zero, a message with no data will be sent (so, just the COB-ID).
The max number of byte to send is 8.
The **addr_data_TX parameter must point to a valid 8 elements buffer** (even if n_byte is less than 8).
Only CAN standard mode (11bit IDs) is supported (not the extended 29bit IDs) so allowed range is 000h-7FFh.
All the parameters can be changed at every call.

Name	Input			Output		Description
	Name	Descr.	Type	Descr.	Type	
CAN2_Rx1 (addr_n_byte, addr_data_RX)	addr_n_byte	pointer to n. of byte of data received	DINT	TRUE=a message was received and data buffer filled FALSE=no received message	BOOL	This function check if a received message is present in to the 1st receive mailbox and, if so, returns data message and size.
	addr_data_RX	pointer to the buffer to contain the received data	DINT			
CAN2_Rx2 (addr_n_byte, addr_data_RX)	addr_n_byte	pointer to n. of byte of data received	DINT	TRUE=a message was received and data buffer filled FALSE=no received message	BOOL	This function check if a received message is present in to the 2nd receive mailbox and, if so, returns data message and size.
	addr_data_RX	pointer to the buffer to contain the received data	DINT			

CAN2_Rx1 and **CAN2_Rx2** functions are used to receive messages from the CAN bus, using 1st and 2nd receive mailboxes.
These functions don't wait for a message but check if a new message is present into the receive mailbox: if there is no message the function exits returning FALSE.
CAN messages are accepted only if the COB-ID equals to those specified with **CAN2_Init** function.
The **addr_data_RX parameter must point to a valid 8 elements buffer**.
The function removes the received message from the mailbox so a second call will return FALSE (if meanwhile a new message was not received).

Via dell'Oreficeria, 41
36100 Vicenza - Italy
Tel +39 0444 343555
Fax +39 0444 343509
www.bdfdigital.com