Utilities TDE MACNO

User's Manual

A.F.E. Logic Lab

Cod. MU00201E00 V_1.1

Index

1. Application level available resources

- 100Kword of Program Memory FLASH
- 4Kword of Data Memory RAM
- 30Kword Data Memory available on permanent memory (EEPROM)
- FAST routine execution time = PWM Period – 50us (150us with 5KHz)
- 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.
- 100 Extra Parameters (P200-P299)
- 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

2. 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.

2.1 Group = “I/O VARIABLES”

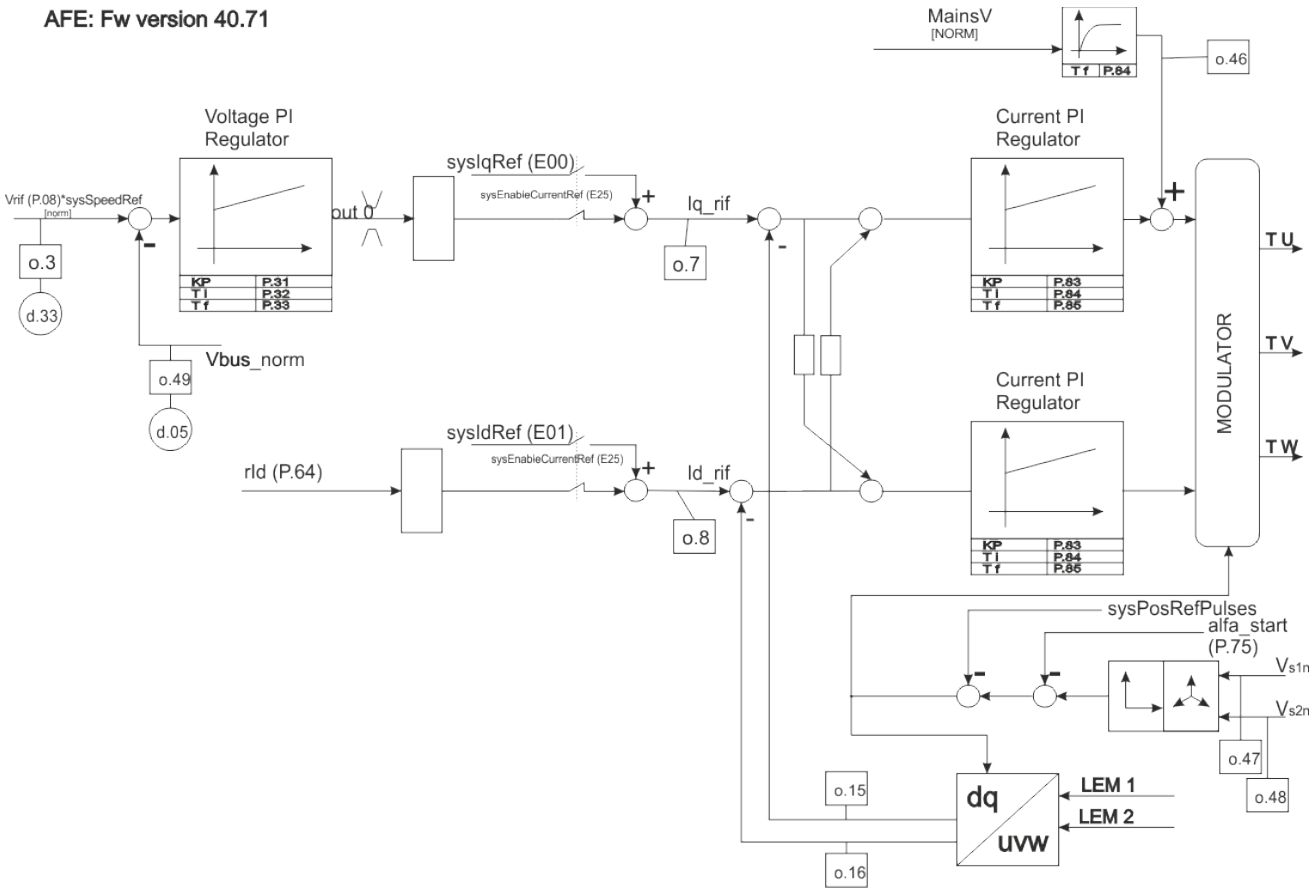
Type	Index	Name	Elements		R/W	Description
Input	IX0.0	sysLogicalInput	ARRAY[8]	BOOL	R	8 physical logic inputs state
Input	IX1.0	sysTabDigInput	ARRAY[32]	BOOL	R	32 logical input functions state (from connector, serial line and fieldbus)
Input	IW2.0	sysHwDigInputMask	1	INT	R	3 power inputs state into less 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]

2.2 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

2.3 Group = “CONTROL VARIABLES”

Type	Index	Name	Elements		R/W	Description
Memory	MD50.1	sysPosRefPulses	1	DINT	R/W	Grid phase shift
Memory	MD50.2	sysSpeedPercReference	1	REAL	R/W	Voltage reference in % of P08 (DC Bus voltage reference)
Memory	MD50.3	sysMaxPositiveTorque	1	REAL	R/W	Maximum positive torque referred to CONV_I_NOM
Memory	MD50.4	sysMaxNegativeTorque	1	REAL	R/W	Maximum negative torque referred to CONV_I_NOM
Memory	MD50.5	sysMaxTorque	1	REAL	R/W	Symmetrical Torque limit referred to CONV_I_NOM
Memory	MD50.7	sysIdRef	1	REAL	R/W	Percentual (*reactive current*) Id reference 0..2
Memory	MD50.8	sysIqRef	1	REAL	R/W	Percentual (*active current*) Iq reference 0..2
Memory	MX51.0	sysRunCommand	1	BOOL	R/W	Run command
Memory	MX51.6	sysResetAll	1	BOOL	R/W	Alarms reset
Memory	MX51.7	sysExtEnable	1	BOOL	R/W	External enable from the field
Memory	MX51.9	sysMotorThermalSwitch	1	BOOL	R/W	Reactance termo-witch state
Memory	MX51.11	sysEnableCurrentRef	1	BOOL	R/W	Enable current reference



2.4 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	sysIntOpdExplorer	ARRAY[50]	INT	R	OpdExplorer internal value table

- **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
TERNA	D03 – Senso ciclico terna di rete		1
GRID_F	D04 – Measured grid frequency	Hz	16
V_BUS_NORM	D05 – V bus Norm	% VBUS_NOM	163.84
GRID_SEL	D06 – Grid Type		1
PRC_IQ_REF	D07 - Request of active Iq rif	% I_CONV_NOM	40.96
PRC_ID_REF	D08 - Request of reactive Id rif	% I_CONV_NOM	40.96

REACT_I	D11 – Current module	A rms	16
PRC_IQ	D15 – Active current Iq	% I_CONV_NOM	40.96
PRC_ID	D16 – Reactive current Id	% I_CONV_NOM	40.96
PRC_CONV_V	D18 – Reference voltage module	% V_GRID_NOM	40.96
MOD_INDEX	D19 - Modulation index		40.96
PRC_VQ_REF	D20 - Vq rif	% V_GRID_NOM	40.96
GRID_V	D21 – Grid AC Voltage	V rms	1
PRC_VD_REF	D22 - Vd rif	% V_GRID_NOM	40.96
DC_BUS	D24 - Bus voltage	V	16
CONV_TEMP	D25 - Radiator temperature reading	°C	16
RACT_TEMP	D26 – Reactor temperature	°C	16
PRC_REACT_I_THERM	D28 – Reactor thermal current	% soglia All	40.96
PRC_CONV_I_MAX	D29 – Current limit	% I_CONV_NOM	40.96
VBUS_REF_NORM	D33 – DC Voltage Reference (Norm)	DC_BUS_NOM	163.84
REG_CARD_TEMP	D40 - Regulation card temperature	°C	16
MOT_PRB_RES	D41 - Thermal probe resistance	KOhm	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
CPLD_FW_REV	D47 – CPLD software version		1
PRC_APP_T_MIN	D48 - Minimum torque limit by application	% MOT_T_NOM	40.96
WORK_HOURS	D49 - Work Hours	hours	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
PWM_SYNC_DELAY	D81 – PWM SYNC DELAY	us	10

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

Description	UM	Scale
O00 Angle read		327.68
O01 Delta m		327.68
O03 V Bus Ref Norm	% VBUS_NOM	163.84
O05 out0	% I_CONV_NOM	40.96
O06 Internal value status: (Monitor Only)		1

O07 Iq rif		40.96
O08 Id rif	% I_CONV_NOM	40.96
O10Internal value alarmss: (Monitor Only)		1
O11 Current module	% I_CONV_NOM	40.96
O13 U phase current reading	% DRV_I_MAX	1
O14 Internal value alarmss: (Monitor Only)		40.96
O15 Iq component of current reading	% I_CONV_NOM	40.96
O16 Id component of current reading	% I_CONV_NOM	40.96
O17 U phase voltage duty-cycle		40.96
O18 Module of the reference voltage	% V_GRID_NOM	40.96
O19 Modulation index		40.96
O20 Request Q axis voltage (Vq_rif)	% V_GRID_NOM	40.96
O22 Request D axis voltage (Vd_rif)	% V_GRID_NOM	40.96
O23 F_fi		40.96
O24 Bus voltage	%900V	40.96
O25 Radiator temperature reading	%37.6°	40.96
O26 Reactance temperature reading	% 80°	40.96
O28 Reactance thermal current	% soglia All	40.96
O29 Current limit	% I MAX CONV	40.96
O32 Internal value: outputs (MONITOR only)		40.96
O33 Internal value: inputs_hw (MONITOR only)		1
O34 V phase current reading	% I MAX CONV	40.96
O35 W phase current reading	% I MAX CONV	40.96
O36 alfa_fi	100%=180°	40.96
O37 Analog input A.I.1	100%=16383	163.84
O38 Analog input A.I.2	100%=16383	163.84
O39 Analog input A.I.3	100%=16383	163.84
O46 Grid voltage module filtered	% V_GRID_NOM	163.84
O47 U phase voltage reading Vu1	100%=16383	163.84
O48 V phase voltage reading Vu2	100%=16383	163.84
O49 V Bus Norm	100%=Vgrid*1,41	163.84

2.5 Group = “FIELDBUS VARIABLES”

Type	Index	Name	Elements		R/W	Description
Memory	MD1070.2	sysFieldbusSpeedPercRef	1	REAL	R	Fieldbus Speed reference referred to MOT_SPD_MAX (P65)
Memory	MD1072.0	sysFieldbusInputs	1	UDINT	R	32 logical input functions state from fieldbus

2.6 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

2.7 Gruppo = “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

2.8 Gruppo = “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 disables the SPI module

3. Embedded blocks

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

3.1 Group = “DRIVE PARAMETERS”

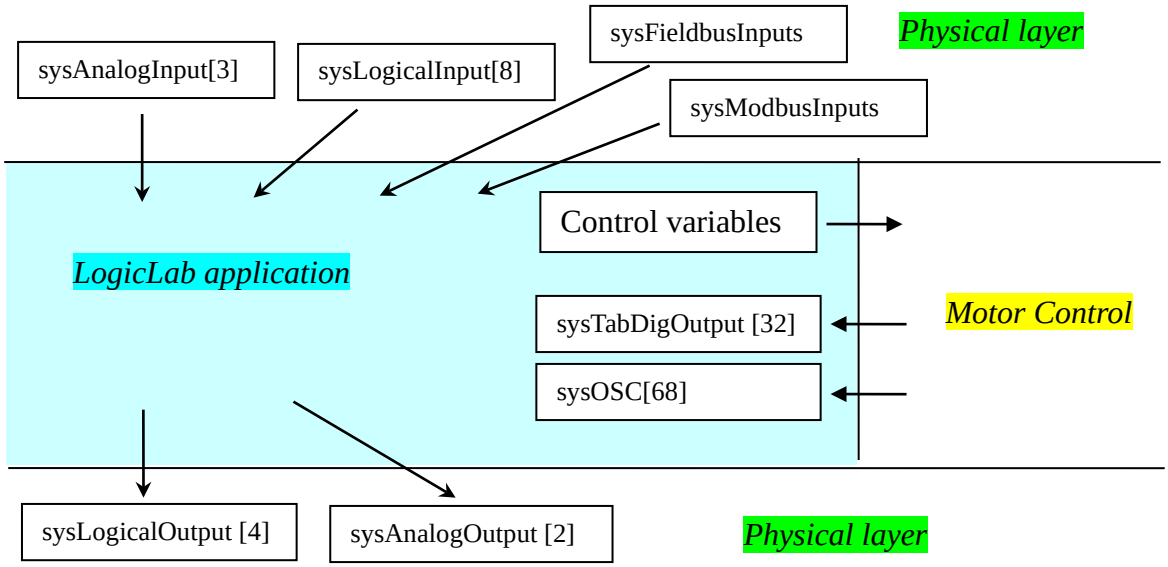
Name	Input Variables			Output		Description
	Name	Descr.	Type	Descr.	Type	
WritePar(index.data)	index	Parameter Table index [0÷199]	INT	Write result TRUE=ok FALSE=no	BOOL	This block can be used to write a standard parameter [0÷199] with key and drive status control.
	data	value	INT			
WriteExtraPar (index,data)	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			

3.2 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.

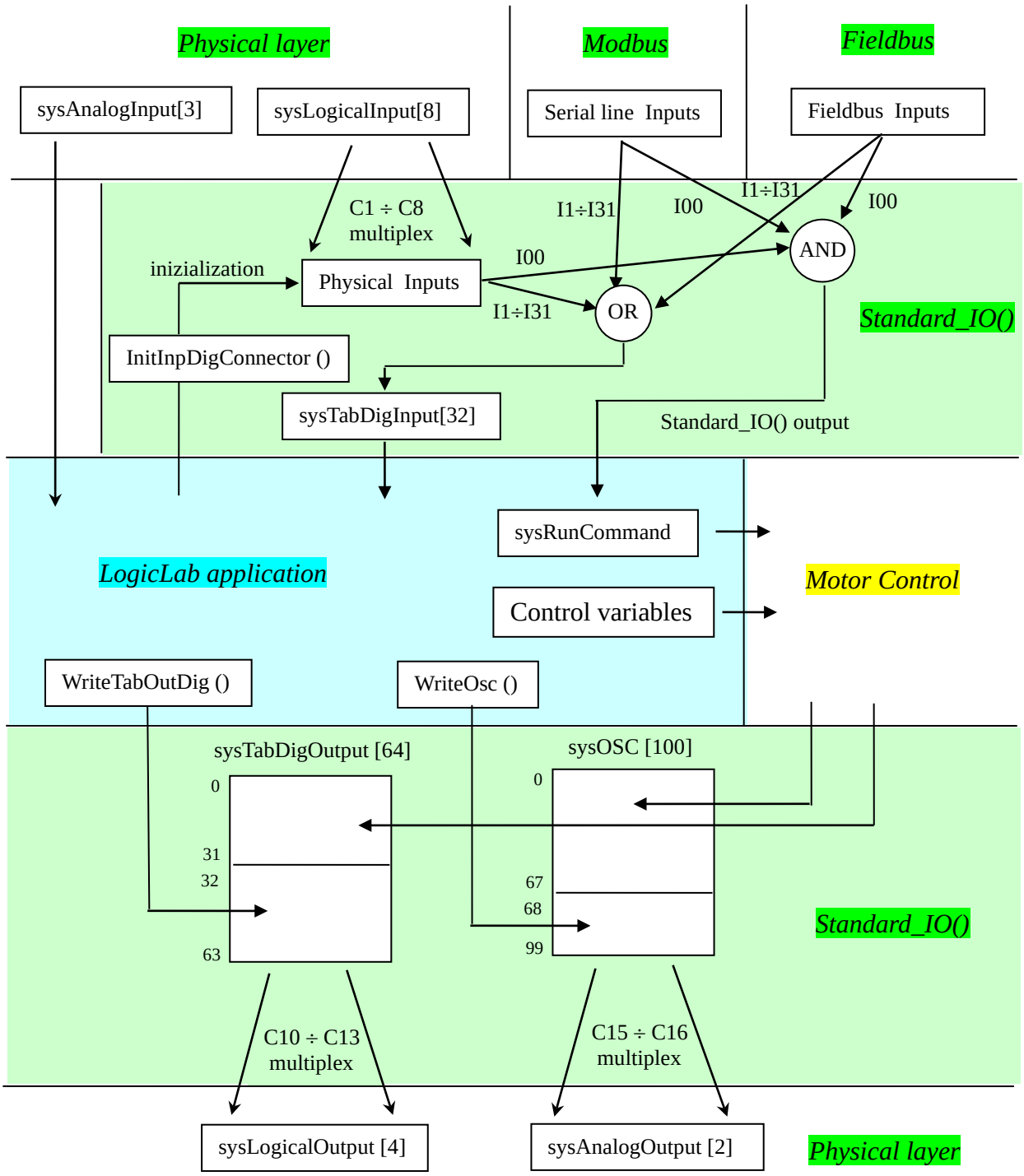
3.2.1 Direct control



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”.

3.2.2 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			

3.3 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			

3.4 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

3.5 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, inhibit_time, event_time)	pdo_number	Transmit 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 Transmit PDO Communication Parameter. **It must be used into BOOT or SLOW tasks.**
	cob_id_pdo	COB-ID PDO	DINT			
	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_ndex_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

CS = 130	=>	EXP from any state to the “reset application” sub-state 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**)**

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