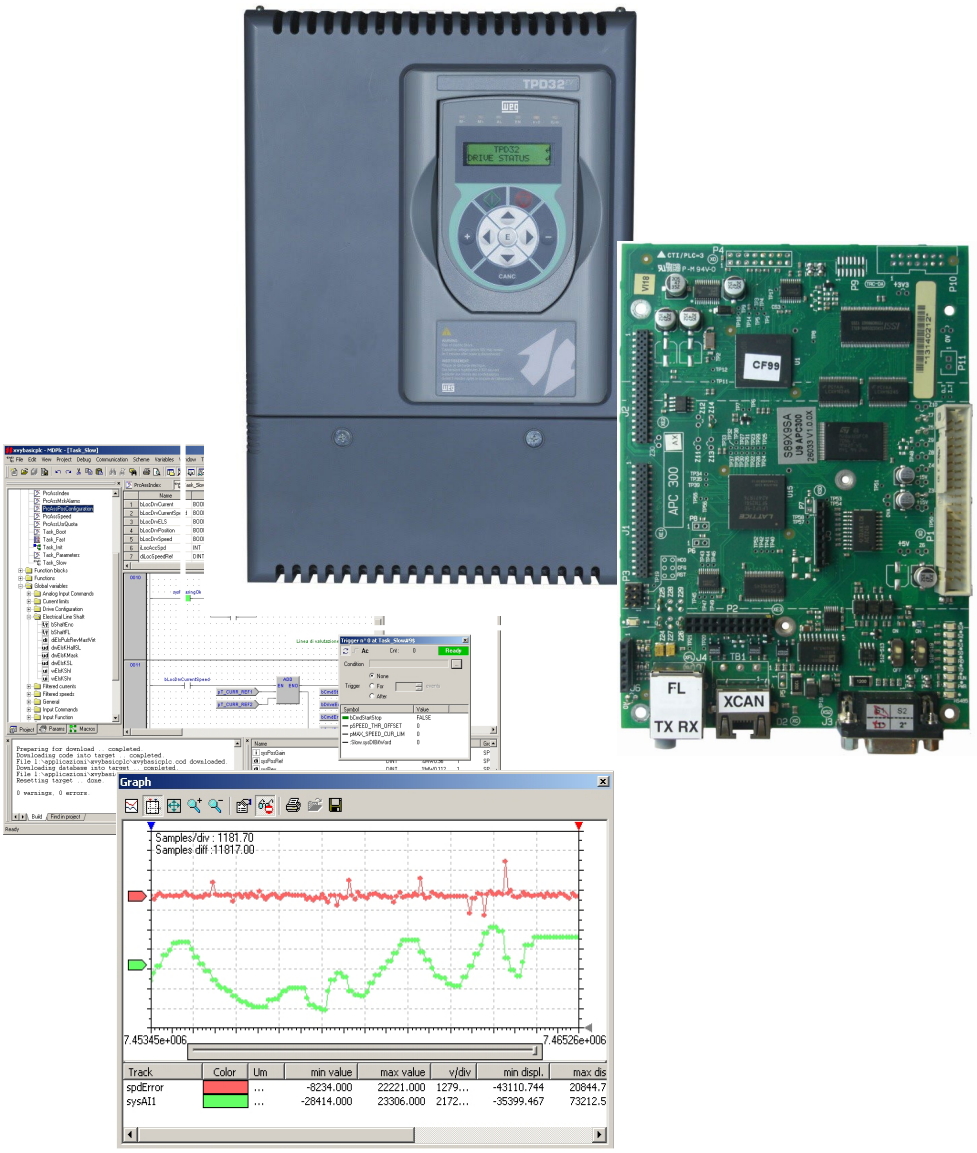
How to write the applications with the MDPlc on APC300

APC300

Language: English



How to write the applications with MDPLC on APC300 Rev. 0.1 - 31 January. 2014

This manual applies to the hardware and software configurations of the following drives:
APC300_V2_X_0_20130326

Thank you for choosing this WEG product.
We will be glad to receive any information which could help us improve this manual. The e-mail address is the following: techdoc@weg.com.
Before using the product, carefully read the safety instruction section on TPD32-EV converter manual.
Keep the manual in a safe place and available to engineering and installation personnel during the product functioning period.
WEG spa has the right to modify products, data and dimensions without notice.
The data can be used only for the product description and cannot be understood as legally stated properties.

Table of Contents

1. DEFINITION OF AN APPLICATION

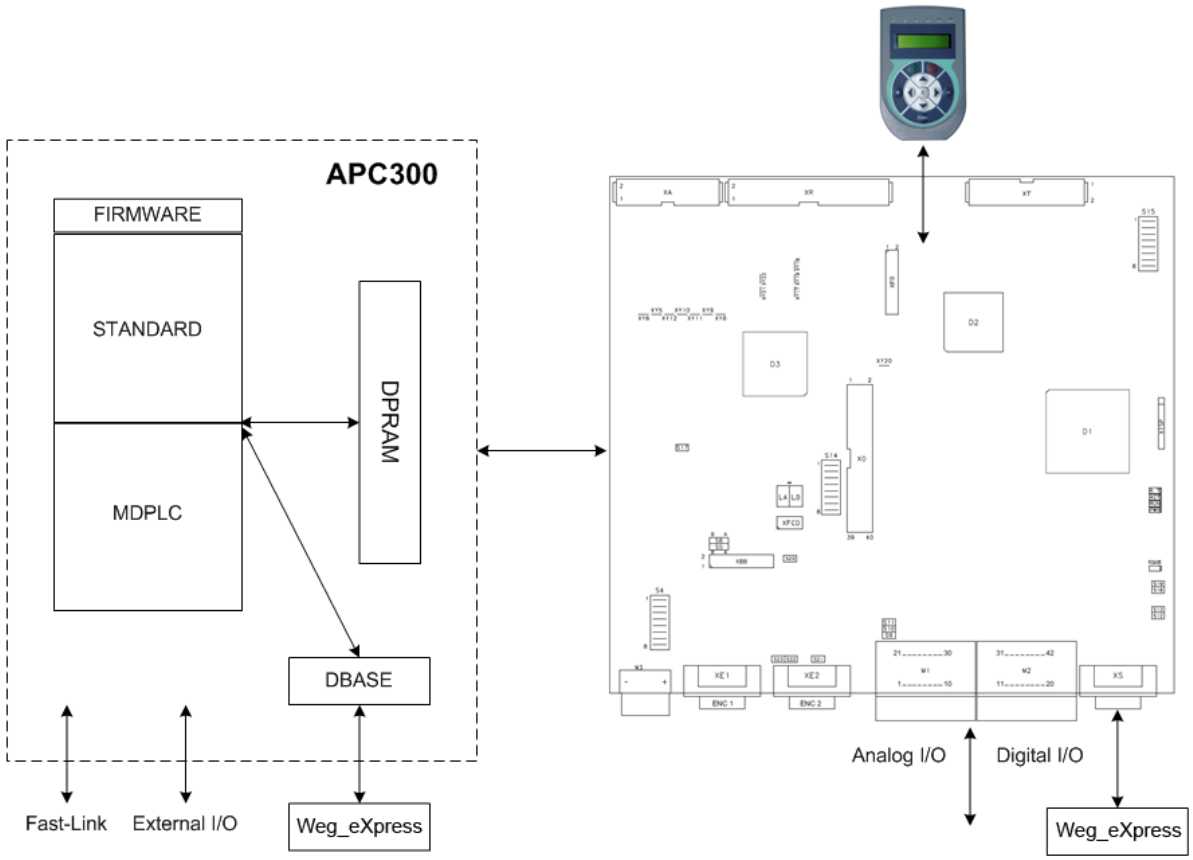
Using the MDPIc developing environment the user can write and download an application on the APC300 board.

The application is defined by:

- **Tasks** where the user can write code using the 5 available IEC-61131 languages
- **Parameters and variables defined by the user** which can be organized in menus to be accessed by the tasks

Together with the parameters defined by the user, the drive has:

- Pre-existent **system parameters and variables** organized in hierarchical menus, which allow the drive basic configuration.
- Pre-existent **system internal variables** which allow to control and manage the drive different functions such as the external or drive inputs, the external or drive outputs, the references, the DPRAM communications with TPD32-EV, etc.



On TPD32-EV there is Drive (Converter) Firmware that is responsible of Motor control, communication with Keypad, Fieldbus, I/O Digital/Analog management.
The System Firmware that performs a big number of functions for the control motor, is always present, is not possible remove it.
The APC300 option installs in the drive with which it exchange information via DUAL PORT RAM.
With the DPRAM, you can read and write the drive parameters (including parameters connected to analog and digital I/Os and to fieldbus cards) in “Fast”, “Slow” or “Manual” mode.
With the DPRAM, you can access the converter I/Os on the TPD32-EV control board.
From the drive you can also access APC300 card parameters by means of the keypad (Option2 menu), on the Option2 menu you can read and write all of the APC300 card parameters (system parameters and MDPLC application parameters).
Connection and parameterization via WEG -eXpress is possible only from the serial port of the APC300 card. The WEG - eXpress TPD32-EV control serial connection allows parameterization of drive parameters but not of APC300 card parameters.
The APC300 option also contains an interface to external I/Os via XCan - CanOpen (for example, Gilogik modules) and fast communication (Fast Link) via fiber optics.

2. TASKS

In the task the user writes source code in order to perform the required functions. Inside the task it is possible to enter all user parameters and all system internal variables. In some Task is possible read/write parameters, variables (system and application). In some Task is possible read/write the fields (min, max, eu, scale) of parameters and variables (system and application).

After starting the drive or after a reset the **Boot** task is performed, where it is possible to perform the first initialization operations.

Afterwards all the other tasks are automatically launched.

The tasks available on the APC300 drive are:

Name	Type and execution time	Description
Boot	Asynchronous. It is performed only once immediately after the start up before any other task	It is used for initialization operations, to set the string message for keypad, to rename system parameters and variables, etc.
Fast	Synchronous at 1 msec. It start when system drive initialization is done.	It is the fastest task. It is used for real time checks and for the control logic
Slow	Synchronous at 8 msec. It start when system drive initialization is done.	It is used for real time checks and for the control logic
Background	Asynchronous. It is performed only when System cpu doesn't have other jobs to complete.	It is used for slow operations such as show display message, etc.
Parameters	Asynchronous. It is performed every time the user changes a system parameter or application parameter defined by user. It is not performed in case of parameter change from task Parameters or read/write event. It is not performed at the start up during parameter initialization. It is not performed in case of read of system parameter or application parameter defined by user.	It is useful to perform specific operations after the modification of a system parameter or application parameter defined by user.

3. PARAMETERS AND VARIABLES DEFINED BY THE USER

The user defined parameters and variables (these are also defined as **read only parameters)** are specific for the application. They all have an index (IPA). Parameters inside range 11000 – 11998. Variables inside range 12000 - 12998. The use of even index is recommended for a direct addressing through Modbus communication protocol.

They can be organized in menu. In the system menu is available 1 point to hook the application menu. In the application menu is possible to define 1 sub level of menu. Parameters or variables and sub menu cannot be to the same level, that is to a level of menu there are parameters, variables or sub menu. Parameters inserted in the same menu are visualized in increasing order of Ipa.

For parameters and variables defined by the user is possible select if they have to be displayed in Easy mode or only in Expert mode.

The APC300 firmware supplies 1 data block for the parameters declaration.

The APC300 firmware supplies 1 data block for the variables declaration.

The management, the assignment of the parameter indexes and the specific menu are automatically managed by MDPlc.

Parameters of different type are inserted in the same data block.

Variables of different type are inserted in the same data block.

The parameter, contrary to the variable, can be modified and can be saved permanently in the drive flash.

If you change a parameter from keypad or APC300 PC configuration tool and make save parameter command, than the parameter value is saved permanently. If you change a parameter writing internal value and make save parameter command, than the parameter value is not saved permanently. In this case in order to save permanently the parameter value is required a read parameter read before save parameter command.

Inside the tasks it is possible to enter the user parameters and variables placing the prefix **p** before the parameter name or the prefix **v** before the variable name. In this way pINPUT_PAR allows to enter the INPUT_PAR user parameter while vOUTPUT_VAR allows to enter the OUTPUT_VAR user variable.

It is possible to associate at reading and writing parameters the execution of a routine. Inside associated routine using some built-in functions, functions block is possible read or write other parameters or variables. Inside associated routine using some built-in functions, functions block is possible read or write fields (min, max, eu, scale) of other parameters. For external parameter type Enum is not possible set scale. For external parameter type Enum with fields min or max is possible change the bound of Enum list.

Here are the properties of the parameters:

Data Block	Parameter type	Parameters number	Parameter index (IPA)	It can be modified and saved in FLASH
1	Any	320	11000-11998	YES

Here are the properties of the variables (read only parameters):

Data Block	Variable type	Variable number	Variable index (IPA)	It can be modified and saved in FLASH
2	Any	128	12000-12998	NO

The definition of a parameter needs the configuration of the followings fields:

Field	Description
IPA	Index of parameter. Automatically managed by MDPlc but user can change value.
Addr	Address inside data block. Automatically managed by MDPlc.
Menu	Name of Menu where to insert the parameter
Name	Name of parameter. Inside Tasks is possible use parameter using this name with prefix 'p'
Attribute	Parameter attribute - Write only when drive is disabled - Not initialized at start – up - Expert mode
Type Par	External parameter type Type Par: Type external Int: DB_T_INT16 Long: DB_T_INT32 Word: DB_T_UNT16 DWord: DB_T_UNT32 Float: DB_T_FLOAT Bool: DB_T_BIT Enums_user_def: DB_T_ENUM
Type Targ	Internal parameter type Type Targ: Type internal Int: DB_T_INT16_INT Long: DB_T_INT32_INT Word: DB_T_UINT16_INT DWord: DB_T_UNIT32_INT Float: DB_T_FLOAT_INT Bool: DB_T_BIT_INT
Def value	Default parameter value.

<table>
<tr><td></td><td>The changes to def value field are also brought in the actual value field.</td></tr>
<tr><td>Min</td><td>Minimum parameter value.
For parameters Type Par = Enums_user_def the minimum value is the value assigned to the first selection index.</td></tr>
<tr><td>Max</td><td>Maximum parameter value.
For parameters Type Par = Enums_user_def the maximum value is the value assigned to the last selection index.</td></tr>
<tr><td>Scale</td><td>Conversion factor. Insert a float number.
If this field is empty, then any conversion is performed.
Write: Internal value = External value
Read: External value = Internal value
If this field is not empty, then type DIRP conversion is applied .
Write: Internal value = External value * Scale
Read: External value = Internal value / Scale</td></tr>
<tr><td>Unit</td><td>Description of measure unit. (Max length 5 character is recommended)
User can insert his unity or select one unit from dialog window.</td></tr>
<tr><td>Short Description</td><td>Short Description of parameter. This message is used for visualizing the parameter in the menu (drive and PC configuration tool).
If this field is empty the drive automatically produces an alternative message containing IPA information. (English language is recommended)</td></tr>
<tr><td>Description</td><td>Description of parameter. This message is used for showing further information on PC configuration tool.</td></tr>
<tr><td>Note</td><td>Note of parameter.</td></tr>
<tr><td>Format</td><td>Display format. Automatically managed by MDPlc in accord to following table:
<table>
<tr><td>Type Par</td><td>Format</td></tr>
<tr><td>Int</td><td>Integer signed : %d</td></tr>
<tr><td>Long</td><td>Integer signed : %d</td></tr>
<tr><td>Word</td><td>Integer unsigned : %u</td></tr>
<tr><td>DWord</td><td>Integer unsigned : %u</td></tr>
<tr><td>Float</td><td>Float : %n.4f</td></tr>
<tr><td>Bool</td><td>Bool: Off-On</td></tr>
<tr><td>Enums_user_def</td><td>Enum: User selection</td></tr>
</table>
User can change automatically set format to one of the following managed by drive:
<table>
<tr><td>Format</td><td>Description</td></tr>
<tr><td>%X</td><td>Hexadecimal. Managed only for Type Par Word and DWord</td></tr>
<tr><td>%f</td><td>Float</td></tr>
<tr><td>%i.df</td><td>Float Managed only for Type Par Float
‘i’ is the number of integer but is ignored.
‘d’ is the number of decimal range 0..4 different values are converts in value 4.</td></tr>
</table>
User can change automatically set format to one of the format represented with standard printf of language C. These format are not managed by drive but only by Pc configuration tool.</td></tr>
</table>

The definition of a variable needs the configuration of the followings fields:

Field	**Description**
IPA	Index of parameter. Automatically managed by MDPlc but user can change value.
Addr	Address inside data block. Automatically managed by MDPlc.
Menu	Name of Menu where to insert the parameter
Name	Name of parameter. Inside Tasks is possible use parameter using this name with prefix ‘v’
Attribute	Parameter attribute - Expert mode
Type Par	External parameter type <table><tr><td>Type Par</td><td>Type external</td></tr><tr><td>Int</td><td>DB_T_INT16</td></tr></table>

<table>
<tr><td></td><td>
<table>
<tr><td>Long</td><td>DB_T_INT32</td></tr>
<tr><td>Word</td><td>DB_T_UNT16</td></tr>
<tr><td>DWord</td><td>DB_T_UNT32</td></tr>
<tr><td>Float</td><td>DB_T_FLOAT</td></tr>
<tr><td>Bool</td><td>DB_T_BIT</td></tr>
<tr><td>Enums_user_def</td><td>DB_T_ENUM</td></tr>
</table>
</td></tr>
<tr><td>Type Targ</td><td>Internal parameter type
<table>
<tr><td>Type Targ</td><td>Type internal</td></tr>
<tr><td>Int</td><td>DB_T_INT16_INT</td></tr>
<tr><td>Long</td><td>DB_T_INT32_INT</td></tr>
<tr><td>Word</td><td>DB_T_UINT16_INT</td></tr>
<tr><td>DWord</td><td>DB_T_UNIT32_INT</td></tr>
<tr><td>Float</td><td>DB_T_FLOAT_INT</td></tr>
<tr><td>Bool</td><td>DB_T_BIT_INT</td></tr>
</table>
</td></tr>
<tr><td>Scale</td><td>Conversion factor. Insert a float number.
If this field is empty, then any conversion is performed.
Write: Internal value = External value
Read: External value = Internal value
If this field is not empty, then type DIRP conversion is applied .
Write: Internal value = External value * Scale
Read: External value = Internal value / Scale</td></tr>
<tr><td>Unit</td><td>Description of measure unit. (Max length 5 character is recommended)
User can insert his unity or select one unit from dialog window.</td></tr>
<tr><td>Short Description</td><td>Short Description of parameter. This message is used for visualizing the parameter in the menu (drive and PC configuration tool).
If this field is empty the drive automatically produces an alternative message containing IPA information. (English language is recommended)</td></tr>
<tr><td>Description</td><td>Description of parameter. This message is used for showing further information on PC configuration tool.</td></tr>
<tr><td>Note</td><td>Note of parameter.</td></tr>
<tr><td>Format</td><td>Display format. Automatically managed by MDPlc in accord to following table:
<table>
<tr><td>Type Par</td><td>Format</td></tr>
<tr><td>Int</td><td>Integer signed : %d</td></tr>
<tr><td>Long</td><td>Integer signed : %d</td></tr>
<tr><td>Word</td><td>Integer unsigned : %u</td></tr>
<tr><td>DWord</td><td>Integer unsigned : %u</td></tr>
<tr><td>Float</td><td>Float : %n.4f</td></tr>
<tr><td>Bool</td><td>Bool: Off-On</td></tr>
<tr><td>Enums_user_def</td><td>Enum: User selection</td></tr>
</table>
User can change automatically set format to one of the following managed by drive:
<table>
<tr><td>Format</td><td>Description</td></tr>
<tr><td>%X</td><td>Hexadecimal. Managed only for Type Par Word and DWord</td></tr>
<tr><td>%f</td><td>Float</td></tr>
<tr><td>%i.df</td><td>Float Managed only for Type Par Float
‘i’ is the number of integer but is ignored.
‘d’ is the number of decimal range 0..4 different values are converts in value 4.</td></tr>
</table>
User can change automatically set format to one of the format represented with standard printf of language C. These format are not managed by drive but only by Pc configuration tool.</td></tr>
</table>

In the following table there are the string length of some items

Item	Len
Parameter short description	20
Variable short description	20
Enum description	14
Alarm description	14
Menu description	15
Eunit description	5

3.1 ENUM

Inside Enum window is possible define Type Par = Enums_user_def.
For every Enums_user_def is possible set following information :
Value : Is the value read and write to parameters from serial line or bus
Description : Is the description read and write to parameters from keypad.

3.2 ALARMS

Inside system drive there are **16 alarms reserved for application.**
Inside Alarms window is possible set application alarm description.

3.3 MENUS

The user parameters and user variables can be organized in menu. In the system menu is available 1 point to hook the application menu. In the application menu is possible to define 1 sub level of menu. **Parameters or variables and sub menu cannot be to the same level, that is to a level of menu there are parameters, variables or sub menu. Parameters inserted in the same menu are visualized in increasing order of Ipa.**

Inside Menus window is possible set menu structure.

When new project is created appear only the item of system menu where is possible to hook the application menu.
The user can change the description of this menu and build your menu structure.
Menu : Is the menu index. Point hook to system menu must be 0. The user can use index from 1..999
Description: Is the description of the menu
Parent: Is the parent of this menu
Level: Must be set to 0.

Example
My Application
Menu 1
Menu 2

Menu	Description	Parent	Level
0	My Application	0	0
1	Menu 1	0	0
2	Menu 2	0	0

3.4 LOAD PARAMETERS FROM FILE

With Mdplc command "*Parameters\Load parameters from file*" the Def value field of parameters are set at Actual value take from PC configuration parameter file. This command is used at the end of the development of an application when you want force the Actual value of the parameters as Def value.

3.5 LOAD PARAMETERS FROM TARGET

See chapter 3.6.

3.6 READ TARGET PARAMETERS

In case of download new Mdplc application with change value on fields **Eu,Def, Min, Max,Scale** of parameters and variables the Actual value of parameter set on drive is loss. The two commands described following help the developer of the application to **preserve the value of parameters** set on the drive during the phase of development of the application.

Inside parameter window of Mdplc for every parameter there are fields Def,Min,Max. Also exists the field Actual value but it is not visible. The field Actual value is set equal to Def value for every new parameter or at every change value on fields Def,Min,Max.
With Mdplc command **"*Parameters\Load parameters from target*"** the Actual value of parameters are set at Actual value take from Drive. This command is useful at compilation time when you want restore the drive Actual value in case of new Mdplc application download. With option "*Project Option\Parameters\Read target parameters*" there is the possibility to force automatic update of Actual value at every compilation command. The Actual value take from drive is saved inside Mdplc project so it is not required to repeat command "*Parameters\Load parameters from target*". The value take from drive is conserved until there is a change value on fields Def,Min,Max.

Example 1:
Inside your application there are four parameters and on the drive is set and saved an Actual value not equal to default.
If you make change to code without change parameters fields (eu, actual value, def, min, max, scale) than the set value are preserved.
If you make change to code and also change parameters fields (eu, actual value, def, min, max, scale) than the set value are not preserved and default value is restored. In order to avoid this behaviour with option "*Project Option\Parameters\Read target parameters*" there is the possibility to force automatic update of Actual value at every compilation command.
Example 2:
Inside your application there are four parameters and on the drive is set and saved a value not equal to default.
If you add a new parameter than the set value are not preserved and default value is restored. In order to avoid this behaviour with option "*Project Option\Parameters\Read target parameters*" there is the possibility to force automatic update of Actual value at every compilation command.

3.7 MULTI LANGUAGE

Inside Mdplc tool there is the possibility to set parameter, menu, Enum, alarm, Eunit description of one language. The use of English language is recommended for maintain first language of Application equal to first language of system.
Inside Mdplc tool there are four dictionary of I,F,D,E where are listed all parameter, menu, Enum, alarm, Eunit description in English language and it is possible insert translated description.
With option "*Project Option\Parameters\Multi language*" it is possible select to download only one language or download more language.
The developer of the application can select number of languages download on the drive. The selection is done with field Active inside Languages window.
If on system in selected language X but this language is not present inside application than application parameter, menu, Enum, alarm, Eunit are displayed in English language.

3.8 APPLICATION PROTECTION + CODE

With option "*Project Option\Parameters\Application protection*" it is possible select to subordinate application execution to a password setting. The password value depend from Regulation serial number.
With option "*Project Option\Parameters\Code*" it is possible force password value depending from this field value also than regulation serial number.

3.9 OUTPUT PARAMETER FILE

The Mdplc tool produces a **parameters file** containing the system parameters, variables plus application parameters, variables. With default configuration Mdplc put the file inside Project directory. If you open this file all html page included inside parameter file are not available. Modifying the option "*Project Option\Parameters\Output parameter file*" (setting directory where there is installed desired target version of PC configuration tool) then it is possible open the included html page.

3.10 PARAMETERS FILE VERSION

Must be set to Version 3.0.

4. SYSTEM PARAMETERS AND VARIABLES

The system parameters and variables are available in **every application** and are part of the APC300 firmware.

These parameters and variables have a parameter index lower than 11000. They configure the drive function.

In some Mdplc tasks (Boot, Background, Parameters, associated read/write routine) using some system built-in functions, functions block is possible to read or to write the parameters and variables system.
In some Mdplc tasks (Boot, Background, Parameters, associated read/write routine) using some system built-in functions, functions block is possible to read or to write the fields (min, max, eu, scale) of parameters and variables system.
Using a built-in functions, functions block it is possible change description displayed on keypad of all system parameters, variables.
It is possible change description on PC configuration tool of only a sub set of parameters, variables.(Pad x, Bit x decomp mon)
Here with, only the parameters relevant for applications development will be analyzed; for all the other parameters, please refer to the ***APC300 Instruction Manual***.

Menu	Par	Description	EU	Type	Def	Min	Max	Acc	Mod
	558	Application select		ENUM	None	0	2	ERWZ	FVS

It selects what IEC 61131-3 application to activate

0 None
1 Application 1
2 Application 2

The drive is furnished with 1 or 2 IEC 61131-3 application already present. Is necessary to select the desired application, perform save parameter, power-off and power-on the drive.

Menu	Par	Description	EU	Type	Def	Min	Max	Acc	Mod
	504	Application ver.rel		UINT16	0	0	0	ER	FVS

On keypad it show the version.release of application
On serial line or Bus inside High byte there is version, inside Low byte there is release.

Menu	Par	Description	EU	Type	Def	Min	Max	Acc	Mod
	506	Application type		UINT16	0	0	0	ER	FVS

It return the type of application .

0 Custom
1 PID
2 Brake

Menu	Par	Description	EU	Type	Def	Min	Max	Acc	Mod
	572	Application key		UINT32	0	0	(2^32)-1	ERW	FVS

It set the enable key for Mdplc application.
To make some Mdplc application enabled in permanent way it is necessary furnish an enable key.
The developer of the Mdplc application decides if it is subject to the enable key.
Check WEG people in order to know is an Mdplc application require key.
If you have an Mdplc application that require enable key but you have a wrong key, than there are 200 Hour of forced enable condition.
At power-on during this time is show a message that explain that time is expiring.
At power-on after this time will be generated a failure and the Mdplc application will be disabled.
Check WEG people in order to know right key.
The key depend from regulation serial number and application type.

5. SYSTEM INTERNAL VARIABLES

APC300 is connected to the TPD32-EV controller as optional card 2 (Option2). The two cards communicate by means of a dual-port ram (DPRAM) that transfers data in Fast (fixed at 1 mSec), Slow (~32msec) or Manual (on request) mode. With the

DPRAM, you can read and write the drive parameters (including parameters connected to analog and digital I/Os and to fieldbus cards) in “Fast”, “Slow” or “Manual” mode

Using the MDPIc compiler supplied by WEG it is possible to have access to all system variables of the APC300 board firmware.

These variables are listed in the ‘**target variables**’ windows inside the MDPIc development environment (Library toolbar) .

It is important to notice that the compiler creates for some of system variables a process image of the variables themselves. Variables without image are normally stated with sysNoImg…

The creation of the process images implies that each PLC program combined to different tasks is supplied with a copy of the variables; such copy is automatically created by the PLC program before the program execution and it is transferred into the corresponding system variables at the end of the program execution.

The value of the variables having a process image does not change, therefore, during the program execution.

The following paragraphs describe the different variable logic groups according to specific subjects.

5.1 SCALING OF TPD32-EV HIGH PRIORITY PARAMETER

When a APC300 is used a subset of the TPD32-EV parameters can be exchanged with the optional card through the automatic synchronous communication. For more details see the APC300 instruction manual.

Parameter	No.	Format	Value			Read/ Write
			min	max	factory	
T current lim + [CURR]	8	U16	0	2 * TOP_CURR	TOP_CUR R	R/W
T current lim - [CURR]	9	U16	0	2 * TOP_CURR	TOP_CUR R	R/W
In use Tcur lim+ [CURR]	10	U16	0	2 * TOP_CURR	-	R
In use Tcur lim- [CURR]	11	U16	0	2 * TOP_CURR	-	R
Current lim red [CURR]	13	U16	0	2 * TOP_CURR	TOP_CUR R	R
T current ref 1 [CURR]	39	I16	-2 * TOP_CURR	+2 * TOP_CURR	0	R/W
T current ref 2 [CURR]	40	I16	-2 * TOP_CURR	+2 * TOP_CURR	0	R/W
T current ref [CURR]	41	I16	-2 * TOP_CURR	+2 * TOP_CURR	-	R
Speed ref 1 [SPD]	42	I16	-32767	32767	0	R/W
Speed ref 2 [SPD]	43	I16	-32767	32767	0	R/W
Ramp ref 1 [SPD]	44	I16	-32767	32767	0	R/W
Ramp ref 2 [SPD]	48	I16	-32767	32767	0	R/W
Control word	55	U16				R/W
Status word	56	U16				R
Ramp ref [SPD]	110	I16	-32767	32767	-	R
Ramp outp [SPD]	113	I16	-32767	32767	-	R
Speed ref [SPD]	118	I16	-32767	32767	-	R
Actual spd [SPD]	122	I16	-32767	32767	-	R
Adap reference [SPD]	183	I16	-32767	32767	4000	R/W
Enc 1 position [ENC_PLS] *	197	I16	-32767	32767	-	R
Enc 2 position [ENC_PLS] *	198	I16	-32767	32767	-	R
Enc 1 last time [ENC_TIM] *	204	U32	0	2^{32}-1	-	R

Parameter	No.	Format	Value			Read/ Write
			min	max	factory	
Enc 1 last time high [ENC_TIM] *	205	U16	0	65535	-	R
Enc 2 last time {ENC_TIM] *	206	U32	0	2^{32}-1	-	R
Enc 2 last time high {ENC_TIM] *	207	U16	0	65535	-	R
Speed reg output [CURR]	236	I16	-2 * TOP_CURR	+2*TOP_CURR	-	R
Lock speed reg	322	U16	0	1	0	R/W
Enc 2 speed [SPD] *	420	I16	-37767	32767	-	R
Enc 1 speed [SPD] *	427	I16	-37767	32767	-	R
Flux current max	467	U16	819	16384	16384	R/W
Flux reference	500	U16	0	16384	16384	R
Pad 0	503	I16	-32768	32767	0	R/W
Pad 1	504	I16	-32768	32767	0	R/W
Pad 2	505	I16	-32768	32767	0	R/W
Pad 3	506	I16	-32768	32767	0	R/W
Pad 4	507	I16	-32768	32767	0	R/W
Pad 5	508	I16	-32768	32767	0	R/W
Pad 6	509	I16	-32768	32767	0	R/W
Pad 7	510	I16	-32768	32767	0	R/W
Pad 8	511	I16	-32768	32767	0	R/W
Pad 9	512	I16	-32768	32767	0	R/W
Pad 10	513	I16	-32768	32767	0	R/W
Pad 11	514	I16	-32768	32767	0	R/W
Pad 12	515	I16	-32768	32767	0	R/W
Pad 13	516	I16	-32768	32767	0	R/W
Pad 14	517	I16	-32768	32767	0	R/W
Pad 15	518	I16	-32768	32767	0	R/W
Bitword pad A	519	U16	0	65535	0	R/W
Bitword pad B	536	U16	0	65535	0	R/W
Dig input term	564	U16	0	65535	0	R
Dig output term	581	U16	0	65535	0	R
Load comp [CURR]	698	I16	-2 * TOP_CURR	+2 * TOP_CURR	-	R
Ind store ctrl	912	U16	0	65535	0	R/W
Index storing	913	U16	0	+2^{32}-1	-	R
Out vlt level	921	U16	0	16384	16384	R/W
F act speed (rpm) [spd]	924	I16	-32768	32767	-	R
F act speed (d) [spd]	925	I16	-32768	32767	-	R
F T curr % [curr]	928	I16	-2 * TOP_CURR	+2 * TOP_CURR	-	R

Parameter	No.	Format	Value			Read/ Write
			min	max	factory	
Speed ratio	1017	I16	0	32767	+10000	R/W
Spd draw out (d) [SPD]	1018	I16	-32768	32767	-	R

Note !

1) [**SPD**] = Speed settings are expressed in *RPM * 4*

2) [**CURR**] = Current settings are expressed in European Drive rated current / 2000 = Motor Amps (2000 is **TOP_CURR**)

3) [**ENC_PLS**] = Encoders positions are expressed in *pulses * 4*

4) [**ENC_TIM**] = Encoders **last time** (s) are expressed in *50ns units* (1=50nS)

5) Encoder 2 parameters (marked with "*" in the table) can be read by the APC200 only if the parameter **Speed fbk sel** = encoder 2

6) Encoder 1 parameters (marked with "*" in the table) can be read by the APC200 only if - the parameter **Speed fbk sel** = encoder 2 and - a digital encoder is used as encoder 1 (interfacing with the converter by means of the DEII card)

7) **Speed reg output [%]** contains valid information even if the speed regulator is disabled (Enable speed reg = Disabled). If Speed reg output is enabled, it contains the sum of actual speed regulator output and **T current ref 2**.

5.2 SCALING OF ADV200 FAST LINK CONTROL VARIABLES

If your application exchange control variable with other drive in internal unit you need to consider the different target.
If the target is another APC300 on TPD32-EV, see the previous chapter for the TPD32-EV High Priority Parameter.
If the other target is an ADV200, you need to use the internal ADV200 conversions.
In following table there are conversion equation of control variables.

ADV200 Ipa	ADV200 Parameter Description
300	DriveRatedCurrentCT_
560	MainsVoltage
680	FullScaleSpeed
2004	RatedSpeed
2010	RatedPower

$$KPI = 3.141592654$$

$$RatedTorque = \frac{RatedPower * 1000}{RatedSpeed * \frac{2 * KPI}{60}}$$

ADV200 Control variable	Eu	Raw width	Eu => Raw	Raw => Eu
CNT_CURRENT	A	32	$Raw = Eu * \frac{2000H * 2\text{^}16}{DriveRatedCurrentCT\_}$	$Eu = Raw * \frac{DriveRatedCurrentCT\_}{2000H * 2\text{^}16}$
CNT_VOLTAGE_AC	V	32	$Raw = Eu * \frac{2000H * 2\text{^}16}{MainsVoltage}$	$Eu = Raw * \frac{MainsVoltage}{2000H * 2\text{^}16}$
CNT_VOLTAGE_DC	V		$Raw = Eu * \frac{800H * 2\text{^}16}{875V}$	$Eu = Raw * \frac{875V}{800H * 2\text{^}16}$
CNT_TORQUE_PERC	%		$Raw = Eu * \frac{2000H * 2\text{^}16}{100\%}$	$Eu = Raw * \frac{100\%}{2000H * 2\text{^}16}$

ADV200 Control variable	Eu	Raw width	Eu => Raw	Raw => Eu
CNT_TORQUE_NM	Nm	32 DIRP	$Raw = Eu * \frac{2000H * 2\text{^}16}{RatedTorque}$	$Eu = Raw * \frac{RatedTorque}{2000H * 2\text{^}16}$
CNT_FREQUENCY	Hz		$Raw = Eu * \frac{7FFFH * 2\text{^}16}{2000Hz}$	$Eu = Raw * \frac{2000Hz}{7FFFH * 2\text{^}16}$
CNT_SPEED	Rpm		$Raw = Eu * \frac{4000H * 2\text{^}16}{FullScaleSpeed}$	$Eu = Raw * \frac{FullScaleSpeed}{4000H * 2\text{^}16}$
CNT_POWER	kW	32 DIRP	$Raw = Eu * \frac{2000H * 2\text{^}16}{RatedPower}$	$Eu = Raw * \frac{RatedPower}{2000H * 2\text{^}16}$
CNT_DEGREES_C	°C		$Raw = Eu * 10$	$Eu = Raw * \frac{1}{10}$
CNT_PLOSS_NEXT_R	%		$Raw = Eu * \frac{4000H * 2\text{^}16}{100\%}$	$Eu = Raw * \frac{100\%}{4000H * 2\text{^}16}$
CNT_VF_SCALE	%		$Raw = Eu * \frac{8000H * 2\text{^}16}{100\%}$	$Eu = Raw * \frac{100\%}{8000H * 2\text{^}16}$
CNT_RAMP_ACC_TIME	Sec		$Raw = Eu * 4000H * 2\text{^}16 * 0.008$	$Eu = Raw * \frac{1}{4000H * 2\text{^}16 * 0.008}$
CNT_RAMP_DEC_TIME	Sec		$Raw = Eu * 4000H * 2\text{^}16 * 0.008$	$Eu = Raw * \frac{1}{4000H * 2\text{^}16 * 0.008}$
CNT_RAMP_ACC_JERK_TIME	Sec		$Raw = Eu * \frac{8000H * 2\text{^}16}{1000 * 1000 * 0.008}$	$Eu = Raw * \frac{1000 * 1000 * 0.008}{8000H * 2\text{^}16}$
CNT_RAMP_DEC_JERK_TIME	Sec		$Raw = Eu * \frac{8000H * 2\text{^}16}{1000 * 1000 * 0.008}$	$Eu = Raw * \frac{1000 * 1000 * 0.008}{8000H * 2\text{^}16}$
CNT_INERTIA	Kgm^2		$Raw = Eu * \frac{8000H * 2\text{^}16}{0.008 * RatedTorque * \frac{60}{2.0 * KPI}}$	$Eu = Raw * \frac{0.008 * RatedTorque * \frac{60}{2.0 * KPI}}{8000H * 2\text{^}16}$
CNT_PERC_1	%		$Raw = Eu * \frac{2000H * 2\text{^}16}{100\%}$	$Eu = Raw * \frac{100\%}{2000H * 2\text{^}16}$
CNT_PERC_2	%		$Raw = Eu * \frac{4000H * 2\text{^}16}{100\%}$	$Eu = Raw * \frac{100\%}{4000H * 2\text{^}16}$
CNT_PERC_3	%		$Raw = Eu * \frac{8000H * 2\text{^}16}{100\%}$	$Eu = Raw * \frac{100\%}{8000H * 2\text{^}16}$

ADV200 Control variable	Unit
CNT_POSITION_VIRT	EncoderPulse * 2^14
CNT_POSITION_TURN	1 Cnt = 1 Rotation
CNT_POSITION_ENC	EncoderPulse * 4

5.3 DATABASE INTERFACE

Following system internal variables are used inside function block associated to read/write event in order to know Ipa, read or write event , value (integer or float).

DATABASE INTERFACE				
Name	**Type**	**Description**	**Unit**	**R/W**
sysDbIpa	INT	Ipa of the accessed parameter (0..65536)	Null	R
sysDbWrRd	INT	Access type Write or Read (0..1) 0 = Rd. 1 = Wr.	Null	R
sysDbValueI	DINT	Integer value (DINT_MIN..DINT_MAX)	Unit of accessed Ipa	R
sysDbValueF	REAL	Float value (FLT_MIN..FLT_MAX)	Unit of accessed Ipa	R

In case of Write event sysDbValueI or sysDbValueF contain the user value.In case of Read event sysDbValueI or sysDbValueF contain value 0.

5.4 ALARMS PLC

Following system internal variables are used for application alarm generation and for configuration of activity executed after alarm.
The application can produce 16 alarms, every bit produces an alarm.
The activity executed after alarm are:

0 Ignore
1 Warning
2 Disable
3 Stop
4 Quick stop

ALARMS PLC				
Name	**Type**	**Description**	**Unit**	**R/W**
SysAlmPlc	DINT	sys Alarm from Plc	Null	R/W
sysAlmPlcXAct	ARRAY[0..15] OF INT	sys Alarm from Plc X activity (0..4)	Null	R/W
SysAlmPlc1Act	INT	sys Alarm from Plc 1 activity (0..4)	Null	R/W
SysAlmPlc2Act	INT	sys Alarm from Plc 2 activity (0..4)	Null	R/W
SysAlmPlc3Act	INT	sys Alarm from Plc 3 activity (0..4)	Null	R/W
SysAlmPlc4Act	INT	sys Alarm from Plc 4 activity (0..4)	Null	R/W
SysAlmPlc5Act	INT	sys Alarm from Plc 5 activity (0..4)	Null	R/W
SysAlmPlc6Act	INT	sys Alarm from Plc 6 activity (0..4)	Null	R/W
SysAlmPlc7Act	INT	sys Alarm from Plc 7 activity (0..4)	Null	R/W
SysAlmPlc8Act	INT	sys Alarm from Plc 8 activity (0..4)	Null	R/W
SysAlmPlc9Act	INT	sys Alarm from Plc 9 activity (0..4)	Null	R/W
SysAlmPlc10Act	INT	sys Alarm from Plc 10 activity (0..4)	Null	R/W
SysAlmPlc11Act	INT	sys Alarm from Plc 11 activity (0..4)	Null	R/W
SysAlmPlc12Act	INT	sys Alarm from Plc 12 activity (0..4)	Null	R/W
SysAlmPlc13Act	INT	sys Alarm from Plc 13 activity (0..4)	Null	R/W
SysAlmPlc14Act	INT	sys Alarm from Plc 14 activity (0..4)	Null	R/W
SysAlmPlc15Act	INT	sys Alarm from Plc 15 activity (0..4)	Null	R/W
SysAlmPlc16Act	INT	sys Alarm from Plc 16 activity (0..4)	Null	R/W

5.5 COUNTERS

Following system internal variables are counter incremented inside Task Fast and Slow.

COUNTERS				
Name	**Type**	**Description**	**Unit**	**R/W**
sys1kHzTimer	DINT	Timer task 1kHz, fast (DINT_MIN..DINT_MAX)	1 cnt = 1 msec	R
sys128HzTimer	DINT	Timer task 128Hz, slow (DINT_MIN..DINT_MAX)	1 cnt = 8 msec	R

5.6 MESSAGES

Following system internal variables are used for application message configuration.

MESSAGES				
Name	**Type**	**Description**	**Unit**	**R/W**
sysMsgPlc1	STRING[19]	Message from Plc 1	Null	R/W
sysMsgPlc2	STRING[19]	Message from Plc 2	Null	R/W
sysMsgPlc3	STRING[19]	Message from Plc 3	Null	R/W
sysMsgPlc4	STRING[19]	Message from Plc 4	Null	R/W

System built-in function **sysKpdWriteMsg** use this system internal variable for Write Keypad Message from task Boot or Background.

5.7 PARAMETERS DESCRIPTIONS

Following system internal variables are used for application message configuration.

PARAMETERS DESCRIPTIONS				
Name	**Type**	**Description**	**Unit**	**R/W**
sysPar1Ipa	INT	Parameter 1 Ipa (0..65536)	Null	R/W
sysPar1Descr	STRING[20]	Parameter 1 description	Null	R/W
SysPar2Ipa	INT	Parameter 2 Ipa (0..65536)	Null	R/W
SysPar2Descr	STRING[20]	Parameter 2 description	Null	R/W
SysPar3Ipa	INT	Parameter 3 Ipa (0..65536)	Null	R/W
SysPar3Descr	STRING[20]	Parameter 3 description	Null	R/W
SysPar4Ipa	INT	Parameter 4 Ipa (0..65536)	Null	R/W
SysPar4Descr	STRING[20]	Parameter 4 description	Null	R/W
SysPar5Ipa	INT	Parameter 5 Ipa (0..65536)	Null	R/W
SysPar5Descr	STRING[20]	Parameter 5 description	Null	R/W
SysPar6Ipa	INT	Parameter 6 Ipa (0..65536)	Null	R/W
SysPar6Descr	STRING[20]	Parameter 6 description	Null	R/W
SysPar7Ipa	INT	Parameter 7 Ipa (0..65536)	Null	R/W
SysPar7Descr	STRING[20]	Parameter 7 description	Null	R/W
SysPar8Ipa	INT	Parameter 8 Ipa (0..65536)	Null	R/W
SysPar8Descr	STRING[20]	Parameter 8 description	Null	R/W
SysPar9Ipa	INT	Parameter 9 Ipa (0..65536)	Null	R/W
SysPar9Descr	STRING[20]	Parameter 9 description	Null	R/W
SysPar10Ipa	INT	Parameter 10 Ipa (0..65536)	Null	R/W
SysPar10Descr	STRING[20]	Parameter 10 description	Null	R/W
SysPar11Ipa	INT	Parameter 11 Ipa (0..65536)	Null	R/W
SysPar11Descr	STRING[20]	Parameter 11 description	Null	R/W
SysPar12Ipa	INT	Parameter 12 Ipa (0..65536)	Null	R/W
SysPar12Descr	STRING[20]	Parameter 12 description	Null	R/W
SysPar13Ipa	INT	Parameter 13 Ipa (0..65536)	Null	R/W
SysPar13Descr	STRING[20]	Parameter 13 description	Null	R/W
SysPar14Ipa	INT	Parameter 14 Ipa (0..65536)	Null	R/W
SysPar14Descr	STRING[20]	Parameter 14 description	Null	R/W
SysPar15Ipa	INT	Parameter 15 Ipa (0..65536)	Null	R/W
SysPar15Descr	STRING[20]	Parameter 15 description	Null	R/W
SysPar16Ipa	INT	Parameter 16 Ipa (0..65536)	Null	R/W
SysPar16Descr	STRING[20]	Parameter 16 description	Null	R/W

System built-in function **sysDBParDescrUpdate**, **sysDBParDescrUpdateEvent** use this system internal variable for Update parameters description from task Background and R/W event.

5.8 TASK PARAMETERS INTERFACE

Following system internal variables are used inside task parameter in order to know Ipa, read or write event , value (integer or float).
It is performed every time the user changes a system parameter or application parameter defined by user.
It is not performed in case of parameter change from task Parameters or read/write event.
It is not performed at the start up during parameter initialization.
It is not performed in case of read of system parameter or application parameter defined by user.

TASK PARAMETERS INTERFACE				
Name	**Type**	**Description**	**Unit**	**R/W**
sysTaskParIpa	INT	Task parameters Ipa (0..65536)	Null	R
sysTaskParWrRd	INT	Task parameters Wr Rd (0..1) 0 = Rd. (Not available) 1 = Wr.	Null	R
sysTaskParValueI	DINT	Task parameters Value integer (DINT_MIN..DINT_MAX)	Unit of accessed Ipa	R
sysTaskParValueF	REAL	Task parameters Value float (FLT_MIN..FLT_MAX)	Unit of accessed Ipa	R

In case of Write event sysTaskParValueI or sysTaskParValueF contain the user value.

5.9 EXE TIME

Following system internal variables are used for read exe time of tasks.
SysExeTimeType select Task information to analyze (Act,Max,Min).

EXE TIME				
Name	**Type**	**Description**	**Unit**	**R/W**
sysExeTimeTask1	DINT	Exe Time Task 1kHz, fast [clk=75MHz]	1 cnt = 13.333 nsec	R
sysExeTimeTask2	DINT	Exe Time Task 128Hz, slow [clk=75MHz]	1 cnt = 13.333 nsec	R
SysExeTimeBackGround	DINT	Exe Time BackGround [clk=75MHz]	1 cnt = 13.333 nsec	R
SysExeTimeType	INT	Select Task analyzed in other variables. (0..2) 0 Actual 1 Max 2 Min	Null	R/W

5.10 FW APPLICATION

Following system internal variables are used inside Mdplc application for communicate to System firmware Application Ver.Rel.Typ.

FW APPLICATION				
Name	**Type**	**Description**	**Unit**	**R/W**
SysFwApplVerRel	INT	Fw application Ver.Rel (Ver-HByte Rel-LByte) (0..65536)	Null	R/W
sysFwApplTyp	INT	Fw application Typ (0..65536)	Null	R/W

5.11 GENERAL USER VARIABLES

Following system internal variables are used for synchronous and congruent data exchange among task Fast and Slow.

GENERAL USER VARIABLES				
Name	**Type**	**Description**	**Unit**	**R/W**
sysUsrVarw0	UINT	User variable w0	Null	R/W
SysUsrVarw1	UINT	User variable w1	Null	R/W
SysUsrVarw2	UINT	User variable w2	Null	R/W
SysUsrVarw3	UINT	User variable w3	Null	R/W
SysUsrVarw4	UINT	User variable w4	Null	R/W
SysUsrVarw5	UINT	User variable w5	Null	R/W
SysUsrVarw6	UINT	User variable w6	Null	R/W
SysUsrVarw7	UINT	User variable w7	Null	R/W
SysUsrVarw8	UINT	User variable w8	Null	R/W
SysUsrVarw9	UINT	User variable w9	Null	R/W
SysUsrVarw10	UINT	User variable w10	Null	R/W
SysUsrVarw11	UINT	User variable w11	Null	R/W
SysUsrVarw12	UINT	User variable w12	Null	R/W
SysUsrVarw13	UINT	User variable w13	Null	R/W
SysUsrVarw14	UINT	User variable w14	Null	R/W
SysUsrVarw15	UINT	User variable w15	Null	R/W
sysUsrVardw0	UDINT	User variable dw0	Null	R/W
SysUsrVardw1	UDINT	User variable dw1	Null	R/W
SysUsrVardw2	UDINT	User variable dw2	Null	R/W
SysUsrVardw3	UDINT	User variable dw3	Null	R/W
SysUsrVardw4	UDINT	User variable dw4	Null	R/W
SysUsrVardw5	UDINT	User variable dw5	Null	R/W
SysUsrVardw6	UDINT	User variable dw6	Null	R/W
SysUsrVardw7	UDINT	User variable dw7	Null	R/W

Following system internal variable are used for display list configuration. Following system internal variable must be initialized before call built-in function **sysKeypadSetDispList**.

GENERAL USER VARIABLE				
Name	**Type**	**Description**	**Unit**	**R/W**
sysDispListIpa	INT	Parameter ipa (0..65536)	Null	R/W
sysDispListDescr	STRING[6]	Parameter description	Null	R/W

5.12 DIGITAL INPUT EXP

Following system internal parameter and variables are relative to digital input exp.

DIGITAL INPUT EXP					
Name	**Type**	**Description**	**Unit**		**R/W**
sysEDIBitWord	DWORD	Exp Digital input word Scheme: "Digital Inputs"	Null	1200	R
sysEDIBitWordBit1	BOOL	Exp Digital input 1 (0..1) Scheme: "Digital Inputs"	Null		R
sysEDIBitWordBit2	BOOL	Exp Digital input 2 (0..1) Scheme: "Digital Inputs"	Null		R
sysEDIBitWordBit3	BOOL	Exp Digital input 3 (0..1) Scheme: "Digital Inputs"	Null		R
sysEDIBitWordBit4	BOOL	Exp Digital input 4 (0..1) Scheme: "Digital Inputs"	Null		R
sysEDIBitWordBit5	BOOL	Exp Digital input 5 (0..1) Scheme: "Digital Inputs"	Null		R
sysEDIBitWordBit6	BOOL	Exp Digital input 6 (0..1) Scheme: "Digital Inputs"	Null		R
sysEDIBitWordBit7	BOOL	Exp Digital input 7 (0..1) Scheme: "Digital Inputs"	Null		R
sysEDIBitWordBit8	BOOL	Exp Digital input 8 (0..1) Scheme: "Digital Inputs"	Null		R
SysEDIBitWordBit9	BOOL	Exp Digital input 9 (0..1) Scheme: "Digital Inputs"	Null		R
SysEDIBitWordBit10	BOOL	Exp Digital input 10 (0..1) Scheme: "Digital Inputs"	Null		R
SysEDIBitWordBit11	BOOL	Exp Digital input 11 (0..1) Scheme: "Digital Inputs"	Null		R
SysEDIBitWordBit12	BOOL	Exp Digital input 12 (0..1) Scheme: "Digital Inputs"	Null		R
SysEDIBitWordBit13	BOOL	Exp Digital input 13 (0..1) Scheme: "Digital Inputs"	Null		R
SysEDIBitWordBit14	BOOL	Exp Digital input 14 (0..1) Scheme: "Digital Inputs"	Null		R
SysEDIBitWordBit15	BOOL	Exp Digital input 15 (0..1) Scheme: "Digital Inputs"	Null		R
SysEDIBitWordBit16	BOOL	Exp Digital input 16 (0..1) Scheme: "Digital Inputs"	Null		R

5.12.1 EXTIO: Digital input

The following system parameters and internal variables are related to **External IO digital input**:

EXTIO					
Name	**Type**	**Description**	**Unit**		**R/W**
sysExtIODigIn0	DWORD	External expansion digital input 0 *Reports directly to the state of the external inputs from 0 to 31*	Null	5400	R
sysExtIODigIn1	DWORD	External expansion digital input 1 *Reports directly to the state of the external inputs from 32 to 63*	Null	5402	R
sysExtIODigInX	ARRAY[0..1] OF DWORD	External expansion digital input X Array [0-1]	Null		R

5.13 DIGITAL OUTPUT EXP

The following system parameters and internal variables are related to **digital output External exp**.:

DIGITAL OUTPUT EXP					
Name	**Type**	**Description**	**Unit**		**R/W**
sysEDOBitWord	DWORD	Read only exp digital output word Scheme: “Digital Outputs”	Null	1400	R

5.13.1 EXTIO: Digital Output

The following system parameters and internal variables are related to **External I/O Digital Output exp**.:

EXTIO					
Name	**Type**	**Description**	**Unit**		**R/W**
sysExtIODigOut0	DWORD	External expansion digital output 0 Reports directly to the state of the external outputs 0 to 31. In practice, the state of the digital outputs 0 .. 7 is not available from MDPlc as it is overwritten by the drive depends on the configuration of the analog output	Null	5454	RW
sysExtIODigOut1	DWORD	External expansion digital output 1 Reports directly to the state of the external outputs 32 to 63	Null	5456	RW
sysExtIODigOutX	ARRAY[0..1] OF DWORD	External expansion digita output X Array [0-1]	Null		RW

5.14 ANALOG INPUT EXP

Following system internal variable show analog input exp.
The following description is relative to a drive with the parameters of the analog input function set to the default value.
:

ANALOG INPUT EXP					
Name	**Type**	**Description**	**Unit**		**R/W**
sysEAI0	DINT	Exp Analog input 0 Scheme: “Analog Inputs Expansion Card”	4000H * 2^16 = 10V	1600	R
sysEAI1	DINT	Exp Analog input 1 Scheme: “Analog Inputs Expansion Card”	4000H * 2^16 = 10V	1650	R

5.14.1 EXTIO: Analog Input

The following system parameters and internal variables are related to **External IO analog input**:

EXTIO					
Name	**Type**	**Description**	**Unit**		**R/W**
sysExtIOAnaIn0	INT	External expansion analog input 0.	Defined by module builder	5410	R
sysExtIOAnaIn1	INT	External expansion analog input 1	Defined by module builder	5412	R
sysExtIOAnaIn2	INT	External expansion analog input 2	Defined by module builder	5414	R
sysExtIOAnaIn3	INT	External expansion analog input 3	Defined by module builder	5416	R
sysExtIOAnaIn4	INT	External expansion analog input 4	Defined by module builder	5418	R
sysExtIOAnaIn5	INT	External expansion analog input 5	Defined by module builder	5420	R
sysExtIOAnaIn6	INT	External expansion analog input 6	Defined by module builder	5422	R
sysExtIOAnaIn7	INT	External expansion analog input 7	Defined by module builder	5424	R
sysExtIOAnaInX	ARRAY[0..7] OF INT	External expansion analog input X Array [0-7]			R

5.15 ANALOG OUTPUT EXP

Following system internal variable show analog output exp.
The following description is relative to a drive with the parameters of the analog output exp function set to the default value.

ANALOG OUTPUT					
Name	**Type**	**Description**	**Unit**		**R/W**
sysEAO0Value	DINT	Exp Analog Output 0 Value Scheme: “Analog Outputs”	4000H * 2^16 = 10V	1866	R
sysEAO1Value	DINT	Exp Analog Output 1 Value Scheme: “Analog Outputs”	4000H * 2^16 = 10V	1868	R

5.15.1 EXTIO: Analog Output

The following system parameters and internal variables are related to **External analog output** exp.:

EXTIO					
Name	**Type**	**Description**	**Unit**		**R/W**
sysExtIOAnaOut0	INT	External expansion analog output 0. In practice MDPlc this variable is not available. The value written by MDPlc is overwritten by the drive depends on the configuration of the analog output.	Defined by module builder	5460	RW
sysExtIOAnaOut1	INT	External expansion analog output 1 In practice MDPlc this variable is not available. The value written by MDPlc is overwritten by the drive depends on the configuration of the analog output.	Defined by module builder	5462	RW
sysExtIOAnaOut2	INT	External expansion analog output 2	Defined by module builder	5464	RW
sysExtIOAnaOut3	INT	External expansion analog output 3	Defined by module builder	5466	RW
sysExtIOAnaOut4	INT	External expansion analog output 4	Defined by module builder	5468	RW
sysExtIOAnaOut5	INT	External expansion analog output 5	Defined by module builder	5470	RW
sysExtIOAnaOut6	INT	External expansion analog output 6	Defined by module builder	5472	RW
sysExtIOAnaOut7	INT	External expansion analog output 7	Defined by module builder	5474	RW
sysExtIOAnaOutX	ARRAY[0..7] OF INT	External expansion analog output X Array [0-7]			RW

5.16 PADS

Following system internal variable show PAD

PAD				
Name	**Type**	**Description**	**Unit**	**R/W**
sysPadX	ARRAY[0..15] OF DINT	General Purpose Pad Variable X Array of 15 DINT	Null	R/W
sysPad1	DINT	General Purpose Pad Variable 1	Null	R/W
SysPad2	DINT	General Purpose Pad Variable 2	Null	R/W
SysPad3	DINT	General Purpose Pad Variable 3	Null	R/W
SysPad4	DINT	General Purpose Pad Variable 4	Null	R/W
SysPad5	DINT	General Purpose Pad Variable 5	Null	R/W
SysPad6	DINT	General Purpose Pad Variable 6	Null	R/W
SysPad7	DINT	General Purpose Pad Variable 7	Null	R/W
SysPad8	DINT	General Purpose Pad Variable 8	Null	R/W
SysPad9	DINT	General Purpose Pad Variable 9	Null	R/W
sysPad10	DINT	General Purpose Pad Variable 10	Null	R/W
sysPad11	DINT	General Purpose Pad Variable 11	Null	R/W
sysPad12	DINT	General Purpose Pad Variable 12	Null	R/W
sysPad13	DINT	General Purpose Pad Variable 13	Null	R/W
sysPad14	DINT	General Purpose Pad Variable 14	Null	R/W
sysPad15	DINT	General Purpose Pad Variable 15	Null	R/W
sysPad16	DINT	General Purpose Pad Variable 16	Null	R/W

5.17 WORD COMP

Following system internal variable show word comp.

WORD COMP				
Name	**Type**	**Description**	**Unit**	**R/W**
sysWComp	DINT	Word Compression Value (0..65535) Scheme: “WComp”	Null	R

5.18 WORD DECOMP

Following system internal variable show word decomp.

WORD DECOMP				
Name	**Type**	**Description**	**Unit**	**R/W**
sysWDecomp	DINT	Word Decompression Value (0..65535)	Null	R/W
sysWDecompBX	ARRAY[0..15] OF BOOL	Word Decompression Bit X Value (0..1) Array of 16 bool	Null	R
sysDecompB0	BOOL	Word Decompression Bit 0 Value (0..1)	Null	R
SysDecompB1	BOOL	Word Decompression Bit 1 Value (0..1)	Null	R
SysDecompB2	BOOL	Word Decompression Bit 2 Value (0..1)	Null	R
SysDecompB3	BOOL	Word Decompression Bit 3 Value (0..1)	Null	R
SysDecompB4	BOOL	Word Decompression Bit 4 Value (0..1)	Null	R
SysDecompB5	BOOL	Word Decompression Bit 5 Value (0..1)	Null	R
SysDecompB6	BOOL	Word Decompression Bit 6 Value (0..1)	Null	R
SysDecompB7	BOOL	Word Decompression Bit 7 Value (0..1)	Null	R
SysDecompB8	BOOL	Word Decompression Bit 8 Value (0..1)	Null	R
SysDecompB9	BOOL	Word Decompression Bit 9 Value (0..1)	Null	R
SysDecompB10	BOOL	Word Decompression Bit 10 Value (0..1)	Null	R
SysDecompB11	BOOL	Word Decompression Bit 11 Value (0..1)	Null	R
SysDecompB12	BOOL	Word Decompression Bit 12 Value (0..1)	Null	R
SysDecompB13	BOOL	Word Decompression Bit 13 Value (0..1)	Null	R
SysDecompB14	BOOL	Word Decompression Bit 14 Value (0..1)	Null	R
SysDecompB15	BOOL	Word Decompression Bit 15 Value (0..1)	Null	R

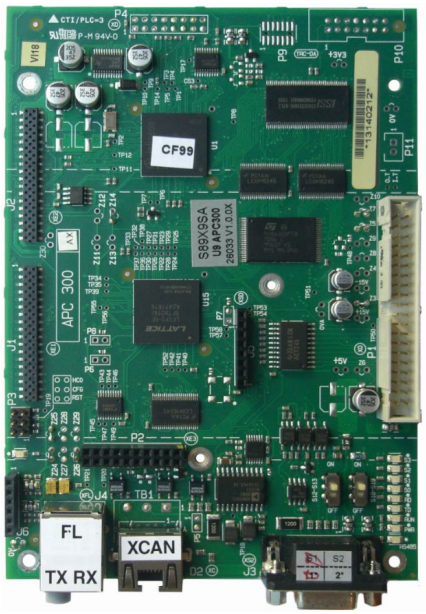
5.19 SYSTEM VARS

Following system internal variable show system vars.

SYSTEM VARS				
Name	**Type**	**Description**	**Unit**	**R/W**
sysFirstAlarm	DWORD	First Alarm Code	Null	R
sysSyncSlaveStatus	DWORD	Slave Synchronization status Pwm Scheme: “Control_FastLink_04” (5720)	Null	R
sysAlarmLO	DWORD	Alarms 1 .. 32 state	Null	R
sysAlarmHI	DWORD	Alarms 33 .. 64 state	Null	R

6. APC300 Ext I/O & FastLink

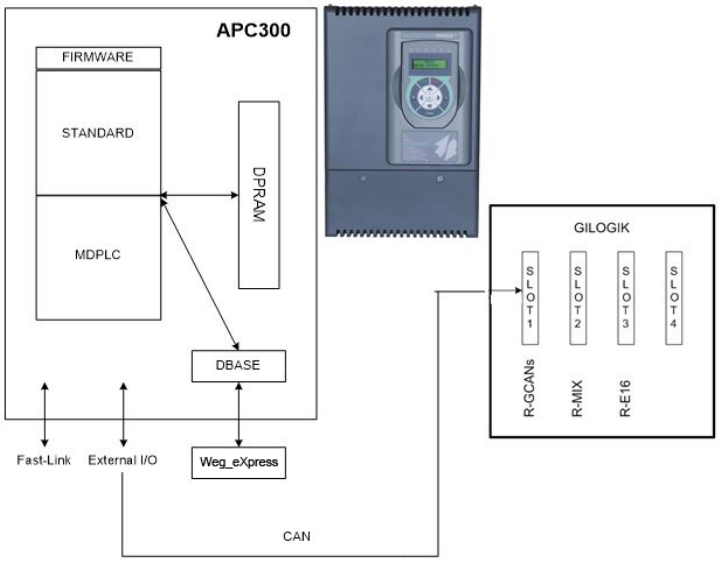
APC300 allow to connect an external I/O expansion (eg. Gilogik, Beckoff..) and or to create a inter-drive fast link communications via optical fiber link and external I / O via CAN.
The two FL and XCAN connector are close to the serial link connector as shown In the figure below.



6.1 EXTERNAL I/O

Through the XCAN connection can be connected an external device for I/O to drive, this increase the number of I / O managed by the drive itself. Communication with the external device is made via CAN using the profile "DS401 Device profile for generic IO modules.
The main features are:

- Communication between drive and external I/O slave
- 4 PDO-RX and 4 PDO-TX
- 64 Dig inp (Max 16 managed by the standard fw, others thru MDPLC)
- 64 Dig Out (Max 8 managed by the standard fw , others thru MDPLC)
- 8 Analog inpt (Max 2 managed by the standard fw , others thru MDPLC)
- 8 Analog Out (Max 2 managed by the standard fw , others thru MDPLC)



Communication between APC300 and external module based on 4 PDO-RX and 4 PDO-TX
The possible I/O configuration is:

64 bit	8 Uint8	4 Uint16	2 Uint32

The information exchanged in the PDO are:

1 PDO-RX	64	Digital Input (bit)
2 PDO-RX	8	Analog Input (Uint16)
1 PDO-RX		Free

1 PDO-TX	64	Digital Output (bit)
2 PDO-TX	8	Analog Output (Uint16)
1 PDO-TX		Free

Through MDPlc you can:

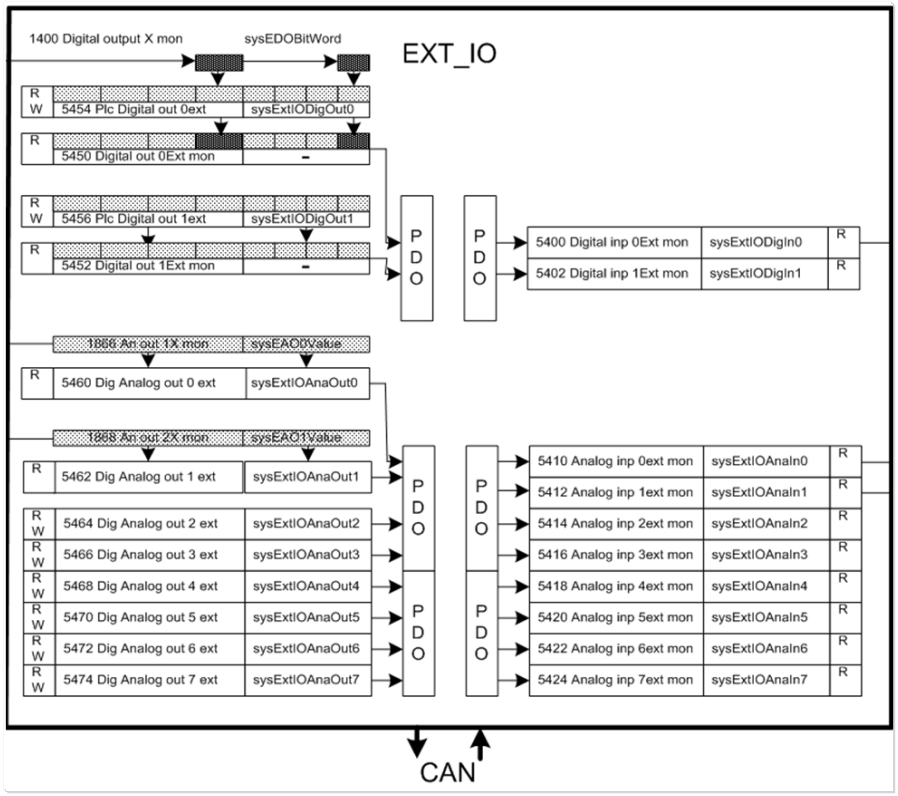
- Read the status of the 5 +1 DI on the control board of the drive via the system variables in the group MDPlc DIGITAL INPUT
- Read the status of the 16 DI external using system variables in the group MDPlc DIGITAL INPUT EXP. Using system variables and sysExtIODigIn0 sysExtIODigIn1 group EXTIO you can read the status of all possible 64 External DI.

- Read the value of the 2 AI on the control board of the drive via the system variables in the group MDPlc ANALOG INPUT
- Read the value of the 2 AI external using system variables MDPlc group ANALOG INPUT EXP.
Using system variables in the group EXTIO sysExtIOAnaIn1 sysExtIOAnaIn0 and you can read the value of all possible external 8 AI.

- Write the status of the 4 DO on the control board of the drive + 8 DO external system variables sysWDecomp (or sysPadX). Obviously the parameters for configuring the DO must be set to function Decomp Word (or PAD).
write the 56 external DO remaining via two system variables and sysExtIODigOut0 sysExtIODigOut1 group EXTIO.

- Write the value of 2 AO on the control board of the drive + 2 external AO system variables sysPadX. Obviously the parameters for the configuration of the AO must be set to the PAD function.
write the remaining 6 AO system through 6 variables sysExtIOAnaOut2 .. sysExtIOAnaOut7 in EXTIO group.

External I/O block diagram:



6.1.1 EXT_IO:

EXT_IO STATE:
The **status** of the **communication** with the external module can be read directly from the application using the following variable:

EXTIO					
Name	**Type**	**Description**	**Unit**		**R/W**
sysExtIOstate	BOOL	External expansion state	Null	5484	R

The value is identical to that of parameter 5484 "ExtIO State", however, the variable is updated immediately in the task.
Vale TRUE when the communication with the module is active. This indicates that the **state** is **Operational** and **were received at least once all the data TPDO slave**.
If a communication cycle (see 5488 "Ext IO Period") is not received all TPDO the variable becomes FALSE and the **alarm** is generated **27 "Opt ExtIO."**

6.2 FAST LINK COMMUNICATIONS

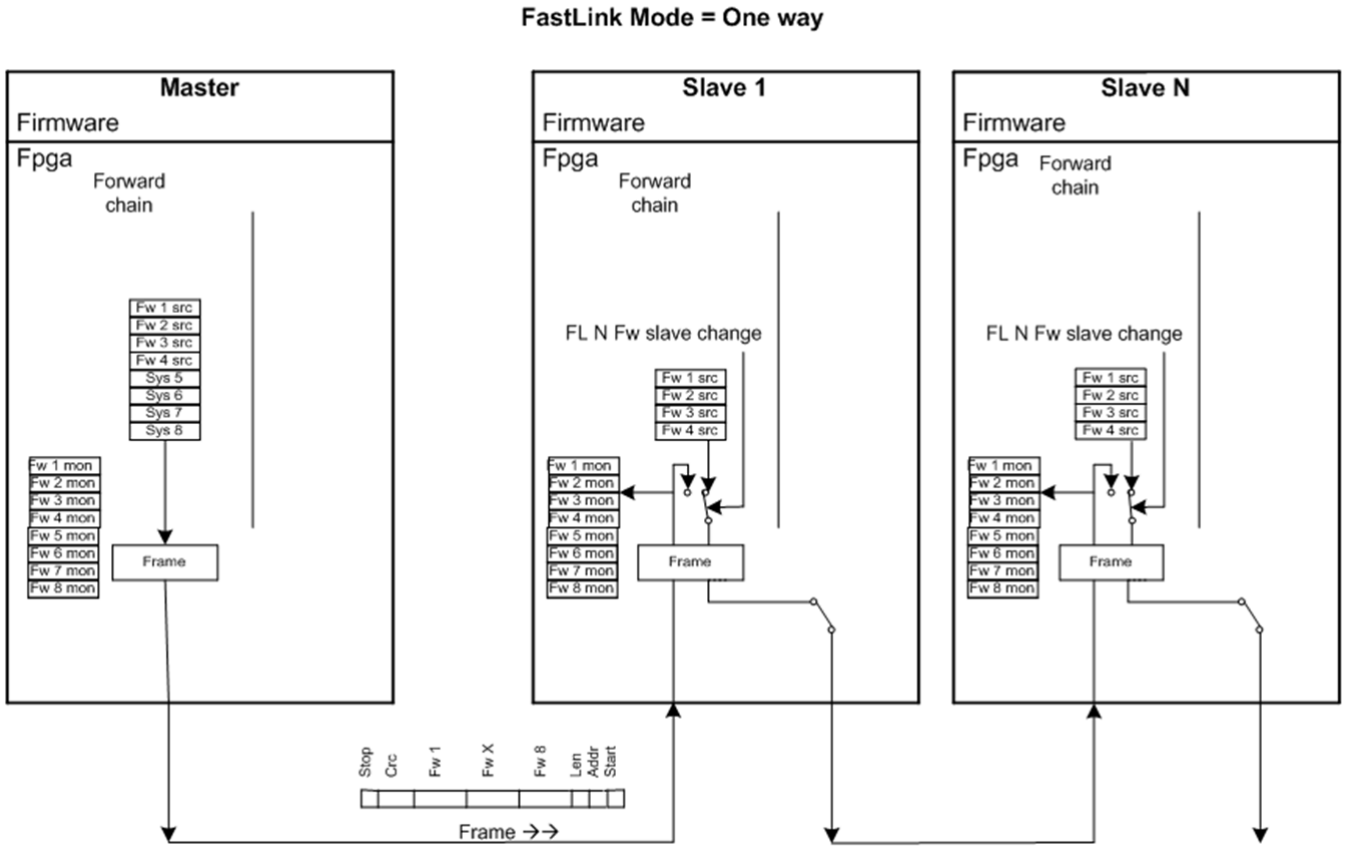
Fast link communications are used in applications where there are multiple drives that work coordinated between them or when it's necessary to transmit information fast to realize synchronization or other functions.
Communication is done with Fiber optics that connect max 16 Partecipants: (1 master + 15 Slave).
Type of transmission data: unidirectional.
Master transmits up to 8 parameters at 32 bit:

- up to 4 parameters in the system menù of the drive
- up to 4 parameters managed by MDPLC

Slave receives all data from the master and through parameter select which can be used
Slave: Slave transmits data to the next. It can replace up to 4 parameters of the system.
Slave: Slave transmits data to the next. It can replace up to 4 parameters of the system.
Data transmission time: 250 usec to 16 participants (including any 'replacement' of the parameters.

Synchronization task: possible value

- No synchronizzations
- Synchronizzation with 125 µSec task
- Synchronizzation with 125 µSec and 1mSec tasks (default)



Parameter:

Menu	Par	Descrizione	UM	Tipo	Fb	Def	Min	Max	Acc	Mod
10.5	5712	FL N Fw slave change		UINT16		0	0	4	ERWZ	FVS

In the unidirectional (one way) mode there are two possible operations:
In the first variant the master send the data frame to the first slave. The first slave take all the data frame and move to the next slave the same frame without making any changes. The same mechanism is repeated by all the slaves.
In the second variant the master send the data frame to the first slave. The first slave take all the data frame and move to the next slave the data frame by replacing a part of the frame. The same mechanism is repeated by all the slaves.
This parameter has no effect on the drive master.
On the slave with this parameter, you can configure the amount of data to be substitutes in the frame.

FL N Fw slave change	Informations substitutes
0	None
1	FL Fw 1
2	FL Fw 1, FL Fw 2
3	FL Fw 1, FL Fw 2, FL Fw 3
4	FL Fw 1, FL Fw 2, FL Fw 3, FL Fw 4

The complete fast link function description and parameters are in the drive manual.

6.2.1 FAST LINK SYSTEM VARIABLES

System Variables in MDPlc environment for the management of FastLink.

FASTLINK					
Name	**Type**	**Description**	**Unit**		**R/W**
sysFL_Fw1_mon	DINT	FastLink Forward 1 monitor Scheme: “Control_FastLink_04”	Null	5750	R
sysFL_Fw2_mon	DINT	FastLink Forward 2 monitor Scheme: “Control_FastLink_04”	Null	5752	R
sysFL_Fw3_mon	DINT	FastLink Forward 3 monitor Scheme: “Control_FastLink_04”	Null	5754	R
sysFL_Fw4_mon	DINT	FastLink Forward 4 monitor Scheme: “Control_FastLink_04”	Null	5756	R
sysFL_Fw5_mon	DINT	FastLink Forward 5 monitor Scheme: “Control_FastLink_04”	Null	5758	R
sysFL_Fw6_mon	DINT	FastLink Forward 6 monitor Scheme: “Control_FastLink_04”	Null	5760	R
sysFL_Fw7_mon	DINT	FastLink Forward 7 monitor Scheme: “Control_FastLink_04”	Null	5762	R
sysFL_Fw8_mon	DINT	FastLink Forward 8 monitor Scheme: “Control_FastLink_04”	Null	5764	R
sysFL_Fw1_inv_mon	DINT	FastLink Forward 1 monitor inverted Scheme: “Control_FastLink_04”	Null	5800	R
sysFL_Fw2_ inv_mon	DINT	FastLink Forward 2 monitor inverted Scheme: “Control_FastLink_04”	Null	5802	R
sysFL_Fw3_ inv_mon	DINT	FastLink Forward 3 monitor inverted Scheme: “Control_FastLink_04”	Null	5804	R
sysFL_Fw4_ inv_mon	DINT	FastLink Forward 4 monitor inverted Scheme: “Control_FastLink_04”	Null	5806	R
sysFL_Fw5_ inv_mon	DINT	FastLink Forward 5 monitor inverted Scheme: “Control_FastLink_04”	Null	5808	R
sysFL_Fw6_ inv_mon	DINT	FastLink Forward 6 monitor inverted Scheme: “Control_FastLink_04”	Null	5810	R
sysFL_Fw7_ inv_mon	DINT	FastLink Forward 7 monitor inverted Scheme: “Control_FastLink_04”	Null	5812	R
sysFL_Fw8_ inv_mon	DINT	FastLink Forward 8 monitor inverted Scheme: “Control_FastLink_04”	Null	5814	R
sysFL_Fw5	DINT	FastLink Forward 5 Scheme: “Control_FastLink_04”	Null	5578	RW
sysFL_Fw6	DINT	FastLink Forward 6 Scheme: “Control_FastLink_04”	Null	5580	RW
sysFL_Fw7	DINT	FastLink Forward 7 Scheme: “Control_FastLink_04”	Null	5582	RW
sysFL_Fw8	DINT	FastLink Forward 8 Scheme: “Control_FastLink_04”	Null	5584	RW

FASTLINK					
Name	**Type**	**Description**	**Unit**		**R/W**
sysFL_Rv1_Slave1_mon	DINT	FaslLink Reverse 1 Slave 1	Null		R
sysFL_Rv1_Slave2_mon	DINT	FaslLink Reverse 1 Slave 2	Null		R
sysFL_Rv1_Slave3_mon	DINT	FaslLink Reverse 1 Slave 3	Null		R
sysFL_Rv1_Slave4_mon	DINT	FaslLink Reverse 1 Slave 4	Null		R
sysFL_Rv1_Slave5_mon	DINT	FaslLink Reverse 1 Slave 5	Null		R
sysFL_Rv1_Slave6_mon	DINT	FaslLink Reverse 1 Slave 6	Null		R
sysFL_Rv1_Slave7_mon	DINT	FaslLink Reverse 1 Slave 7	Null		R
sysFL_Rv1_Slave8_mon	DINT	FaslLink Reverse 1 Slave 8	Null		R
sysFL_Rv1_Slave9_mon	DINT	FaslLink Reverse 1 Slave 9	Null		R
sysFL_Rv1_Slave10_mon	DINT	FaslLink Reverse 1 Slave 10	Null		R
sysFL_Rv1_Slave11_mon	DINT	FaslLink Reverse 1 Slave 11	Null		R
sysFL_Rv1_Slave12_mon	DINT	FaslLink Reverse 1 Slave 12	Null		R
sysFL_Rv1_Slave13_mon	DINT	FaslLink Reverse 1 Slave 13	Null		R
sysFL_Rv1_Slave14_mon	DINT	FaslLink Reverse 1 Slave 14	Null		R
sysFL_Rv1_Slave15_mon	DINT	FaslLink Reverse 1 Slave 15	Null		R
sysFL_Rv1_Slave16_mon	DINT	FaslLink Reverse 1 Slave 16	Null		R
sysFL_Rv2_Slave1_mon	DINT	FaslLink Reverse 2 Slave 1	Null		R
sysFL_Rv2_Slave2_mon	DINT	FaslLink Reverse 2 Slave 2	Null		R
sysFL_Rv2_Slave3_mon	DINT	FaslLink Reverse 2 Slave 3	Null		R
sysFL_Rv2_Slave4_mon	DINT	FaslLink Reverse 2 Slave 4	Null		R
sysFL_Rv2_Slave5_mon	DINT	FaslLink Reverse 2 Slave 5	Null		R
sysFL_Rv2_Slave6_mon	DINT	FaslLink Reverse 2 Slave 6	Null		R
sysFL_Rv2_Slave7_mon	DINT	FaslLink Reverse 2 Slave 7	Null		R
sysFL_Rv2_Slave8_mon	DINT	FaslLink Reverse 2 Slave 8	Null		R
sysFL_Rv2_Slave9_mon	DINT	FaslLink Reverse 2 Slave 9	Null		R
sysFL_Rv2_Slave10_mon	DINT	FaslLink Reverse 2 Slave 10	Null		R
sysFL_Rv2_Slave11_mon	DINT	FaslLink Reverse 2 Slave 11	Null		R
sysFL_Rv2_Slave12_mon	DINT	FaslLink Reverse 2 Slave 12	Null		R
sysFL_Rv2_Slave13_mon	DINT	FaslLink Reverse 2 Slave 13	Null		R
sysFL_Rv2_Slave14_mon	DINT	FaslLink Reverse 2 Slave 14	Null		R
sysFL_Rv2_Slave15_mon	DINT	FaslLink Reverse 2 Slave 15	Null		R
sysFL_Rv2_Slave16_mon	DINT	FaslLink Reverse 2 Slave 16	Null		R

SYSTEM					
Name	**Type**	**Description**	**Unit**		**R/W**
sysSyncSlaveStatus	DWORD	Stato della sincronizzazione dei Pwm Scheme: “Control_FastLink_04”	Null	5720	R

7. APC300 CARD DPRAM Communications

The APC300 card is used to run applications programmable in MdPlc environment.
These applications can affect some of the internal variables of the APC300 (system variables) and the variables declared by the application.
Through some of this system variables and the DPRAM, you can interact with the converter control card or manage other interfaces available on the card (FastLink ExtIO ..).

7.1 DPRAM dialogs between APC300 and Drive

The system variables of the DPRAM menu allow the card to configure the various dialogs and to communicate with the TPD32-EV control card.
There are different communication modes, to be utilized according to the specific application.

MANUAL DIALOG (Asynchronous)	
TPD Par.	**Description**
All	By means of specific functions, the MDPLC application can request the reading or writing of individual TPD32-EV parameters (one at a time).

AUTOMATIC DIALOG (Cyclic)			
Name	**TPD Par.**	**No.**	**Description**
SYNCHRONOUS fast: 1mSec	High Priority (*)	10 A→D 10 D→A	Fast cyclic dialog with data exchanged at 1mSec (but updated by drive every 2ms), settable via assignment of IPA. Data exchanged are in internal unit without any conversions (ex: speed in rpm*4)
ASYNCHRONOUS slow 32mSec	Type Opt2-A/PDC (**)	10 A→D 10 D→A	Slow cyclic dialog with data exchanged at 32 mSec, settable via assignment of IPA.

(*) See list of parameters on table 10.4 “LIST OF HIGH PRIORITY PARAMETERS” in TPD32-EV manual
(**) Only to list of parameters on table of parameters of TPD32-EV that contain a value R or R/W in column Opt2-A/PDC (low-priority parameters)
A→D = from Apc300 to Drive
D→A = from Drive to Apc300

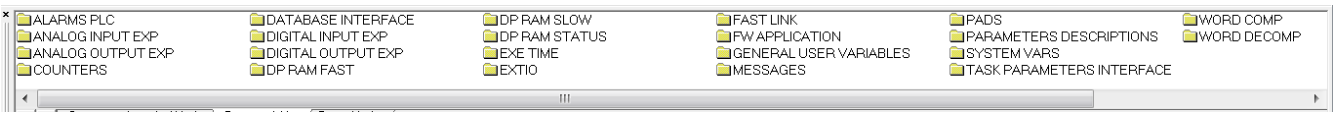
7.2 Type of DPRAM data

In addition to the value of parameters exchanged in manual or automatic mode, the card can access information on the status of the communication and on the type of data exchanged.

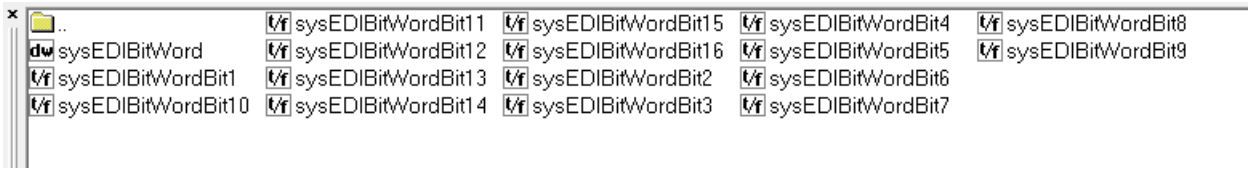
With regard to **data types**, the APC300 - MdPlc variables exchanged in DPRAM have a **32 bit integer format** without sign, whereas the corresponding associated TPD32-EV parameters may have different formats.
Format conversion must be done explicitly by the programmer of the MdPlc application, since only values without conversion or extension of sign are exchanged in DPRAM.

7.3 System variables

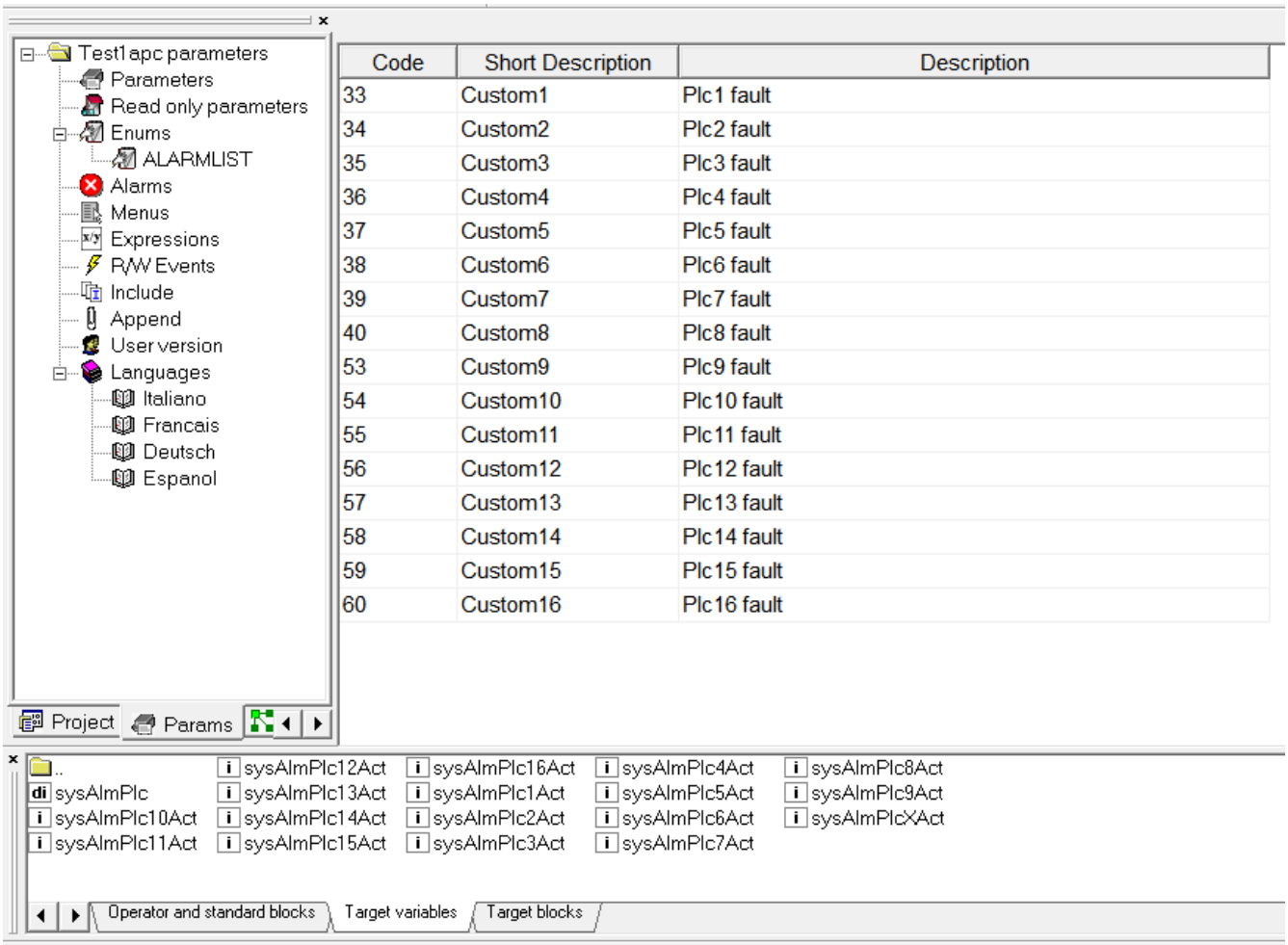
System variables are not only those used in DPRAM data exchange, but also include those used to access internal resources of the APC300 card and internal interfaces.
You can also access APC300 internal data (Pads, Word Comp and Word Decomp, database, counters, etc.).



With system variables, you can manage data exchange over the FastLink line by means of external I/Os.



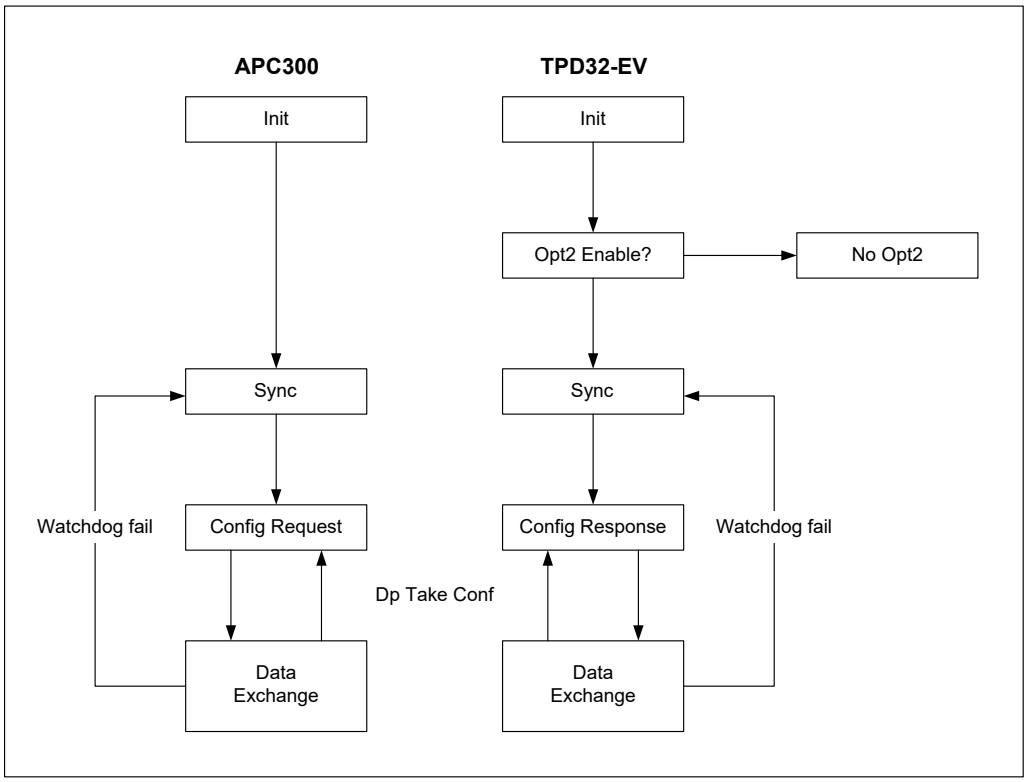
By means of the system variables, the MDPLC application can generate up to 16 alarms corresponding to 16 TPD32-EV alarms. These alarms can be configured as activity and as description by means of the Alarms table in the tab “Params” of MdPlc; in addition, you can detect the TPD32-EV alarm state from the MDPLC.



7.4 APC300 – TPD32-EV synchronization

The synchronization sequence between card and drive is run at startup. If the application has not been loaded or is disabled (parameter 558 “Application select”), even if synchronization fails the firmware system functions are still performed. Therefore, the following are active: data exchange in DPRAM, management via TPD32-EV keypad, FastLink and ExtIO, communication with configurator, etc.

Communication with the TPD32-EV in DPRAM is performed according to the following state machine:



Manual data and alarms are available when synchronization is done.
If Opt2 is not enable, the TPD32-EV never looks for synchronization, therefore the APC300 remains in DPRAM alarm because it cannot synchronize itself and there is no data exchange.
The synchronization procedure indicates the correct initialization of the DPRAM interface between TPD32-EV and APC300, and is run at each start or reset of the APC300.
On the APC300 side, cyclical data exchange takes place only if synchronization has successfully completed.
On the TPD32-EV side, the card goes to data exchange after synchronization.
A watchdog is always active on both sides and trips in case of communication problems.

7.5 List of MdPlc variables for DPRAM

7.5.1 DP RAM FAST:

With regard to MDPLC system variables, **fast data** must be managed only in the MdPlc Fast task @ 1ms.
Ten communication channel are always available, but only those configured with a valid ipa have significance.
TPD32-EV updates the DA channel in asynchronous mode every 2ms.

DP RAM FAST					
NAME	**Type**	**Description**	**Unit**		**R/W**
sysDPfastDAdw	ARRAY [1..10] OF UDINT	Array of Variables 1..10 read by APC300 from Drive using fast channel.	Null		R
sysDPfastDAdw1	UDINT	Variables 1 read by APC300 using fast channel. Correspond to parameters D → A Fast 1	Null	7148	R
sysDPfastDAdw2	UDINT	.. Fast Variables 2 read by APC300. ..	Null	7150	R
sysDPfastDAdw3	UDINT	.. Fast Variables 3 read by APC300. ..	Null	7152	R
sysDPfastDAdw4	UDINT	.. Fast Variables 4 read by APC300. ..	Null	7154	R
sysDPfastDAdw5	UDINT	.. Fast Variables 5 read by APC300. ..	Null	7156	R
sysDPfastDAdw6	UDINT	.. Fast Variables 6 read by APC300. ..	Null	7158	R
sysDPfastDAdw7	UDINT	.. Fast Variables 7 read by APC300. ..	Null	7160	R
sysDPfastDAdw8	UDINT	.. Fast Variables 8 read by APC300. ..	Null	7162	R
sysDPfastDAdw9	UDINT	.. Fast Variables 9 read by APC300. ..	Null	7164	R
sysDPfastDAdw10	UDINT	.. Fast Variables 10 read by APC300. ..	Null	7166	R
sysDPfastDAType	UDINT	Array with types (see TPD32-EV types) of ipa D → A fast	Null		R
sysDPfastDANew	BOOL	Flag indicating updating of fast data by TPD32-EV	Null		R
sysDPfastADdw	ARRAY [1..10] OF UDINT	Array of Fast variables written on TPD32-EV every ms. Correspond to A → D Fast 1..10 dig	Null		R/W
sysDPfastADdw1	UDINT	Fast variables 1 written on TPD32-EV every ms. Correspond to A → D Fast 1	Null	7180	R/W
sysDPfastADdw2	UDINT	Fast variables 2 ..	Null	7182	R/W
sysDPfastADdw3	UDINT	Fast variables 3 ..	Null	7184	R/W
sysDPfastADdw4	UDINT	Fast variables 4 ..	Null	7186	R/W
sysDPfastADdw5	UDINT	Fast variables 5 ..	Null	7188	R/W
sysDPfastADdw6	UDINT	Fast variables 6 ..	Null	7190	R/W
sysDPfastADdw7	UDINT	Fast variables 7 ..	Null	7192	R/W
sysDPfastADdw8	UDINT	Fast variables 8 ..	Null	7194	R/W
sysDPfastADdw9	UDINT	Fast variables 9 ..	Null	7196	R/W
sysDPfastADdw10	UDINT	Fast variables 10 ..	Null	7198	R/W
sysDPfastADType		Array with types (see TPD32-EV types) of 10 ipa A →D fast	Null		R

Example: Fast Task: pA and pB are application parameters.

```
IF sysDp_Sync AND sysDp_Exch THEN
        (*Read*)
        IF sysDPfastDANew THEN
                pA := sysDPfastDAdw1;
        END_IF;
        (*Write*)
        sysDPfastADdw1 := pB;
END_IF;
```

7.5.2 DP RAM SLOW:

Slow data in "DP RAM SLOW" must be managed only in the Slow 8ms task.
Up to 10 data are available, but only those with a valid ipa are exchanged. TPD32-EV updates them in asynchronous mode every 32ms

DP RAM Slow					
NAME	**Type**	**Description**	**Unit**		**R/W**
sysDPslowDAdw	ARRAY [1..10] OF UDINT	Array of Variables 1..10 read by APC300 from Drive every 8 mSec slow channel.	Null		R
sysDPslowDAdw1	UDINT	Variables 1 read by APC300 using fast channel. Correspond to parameters D → A Slow 1	Null	7212	R
sysDPslowDAdw2	UDINT	.. Slow Variables 2 read by APC300. .. D → A Slow 2	Null	7214	R
sysDPslowDAdw3	UDINT	.. Slow Variables 3 read by APC300. ..	Null	7216	R
sysDPslowDAdw4	UDINT	.. Slow Variables 4 read by APC300. ..	Null	7218	R
sysDPslowDAdw5	UDINT	.. Slow Variables 5 read by APC300. ..	Null	7220	R
sysDPslowDAdw6	UDINT	.. Slow Variables 6 read by APC300. ..	Null	7222	R
sysDPslowDAdw7	UDINT	.. Slow Variables 7 read by APC300. ..	Null	7224	R
sysDPslowDAdw8	UDINT	.. Slow Variables 8 read by APC300. ..	Null	7226	R
sysDPslowDAdw9	UDINT	.. Slow Variables 9 read by APC300. ..	Null	7228	R
sysDPslowDAdw10	UDINT	.. Slow Variables 10 read by APC300. .. D → A Slow 10	Null	7230	R
sysDPslowDAType	UDINT	Array with types (see TPD32-EV types) of ipa D → A Slow	Null		R
sysDPslowDANew	BOOL	Flag indicating updating of Slow data by TPD32-EV	Null		R
sysDPslowADdw	ARRAY [1..10] OF UDINT	Array of Slow variables written on TPD32-EV every ms. Correspond to A → D Slow 1..10 dig	Null		R/W
sysDPslowADdw1	UDINT	Slow variables 1 written on TPD32-EV every ms. Correspond to A → D Slow 1	Null	7244	R/W
sysDPslowADdw2	UDINT	Slow variables 2 ..	Null	7246	R/W
sysDPslowADdw3	UDINT	Slow variables 3 ..	Null	7248	R/W
sysDPslowADdw4	UDINT	Slow variables 4 ..	Null	7250	R/W
sysDPslowADdw5	UDINT	Slow variables 5 ..	Null	7252	R/W
sysDPslowADdw6	UDINT	Slow variables 6 ..	Null	7254	R/W
sysDPslowADdw7	UDINT	Slow variables 7 ..	Null	7256	R/W
sysDPslowADdw8	UDINT	Slow variables 8 ..	Null	7258	R/W
sysDPslowADdw9	UDINT	Slow variables 9 ..	Null	7260	R/W
sysDPslowADdw10	UDINT	Slow variables 10 ..	Null	7262	R/W
sysDPslowADType		Array with types (see TPD32-EV types)of 10 ipa A→D Slow	Null		R

7.5.3 DPRAM STATUS:

Diagnostics data (read only) in "DP RAM STATUS"

DP RAM Slow					
NAME	**Type**	**Description**	**Unit**		**R/W**
sysDp_Sync	BOOL	Synchronization with TPD32-EV active	Null	7000	
sysDp_Exch	BOOL	Data exchange active (at least one fast or slow word programmed; if more than one, all programmed correctly)	Null	7002	
sysDp_AztAlarm	UDINT	TPD32-EV alarm code (see TPD32-EV manual)	Null		

APC300->TPD32-EV manual dialog
Executed requests are in the **boot and background** task by the MdPlc application toward TPD32-EV parameters. Requests can be executed only if sysDP_Sync = TRUE.

Reading TPD32-EV parameter (Getpar)
The function returns a value indicating the result of the request; if it returns value other than zero the request failed and the read/written data is not valid.

fbSysDpGetIntegerPar
ipa
result
val

fbSysDpGetFloatPar
ipa
result
val

I.e. :

- read integer parameter *fbSysDpGetInteger*
- fb is called by background task
- reads Pad0 of TPD32-EV, ipa 503 + 8192 = 8695

```
IF sysDp_Sync THEN
        (*Reading a parameter *)
fbSysDpGetInteger.ipa := 8695;
fbSysDpGetInteger();
IF fbSysDpGetInteger.result = 0 THEN
        pA := fbSysDpGetInteger.val;
END_IF;
END_IF;
```

Writing TPD32-EV parameter (PutPar)

res : =SysDpPutIntegerPar(ipa,val)
res := SysDpPutFloatPar(ipa,val)

Embedded functions:

Orange LEDs S0 and S1 can be managed directly by the MdPlc application by means of embedded functions *sysLedS0* and *sysLedS1* .

res := sysLedS0(val);
res := sysLedS1(val);

the value “val” is a dword, if 0 the corresponding LED is off, if 1 the LED is on.

Manual dialog TPD32-EV → APC300:

This is used for asynchronous requests from TPD32-EV to APC. If you use the profibus card, in CC dialogs you can read and write single parameters of the APC card by using manual dialogs.

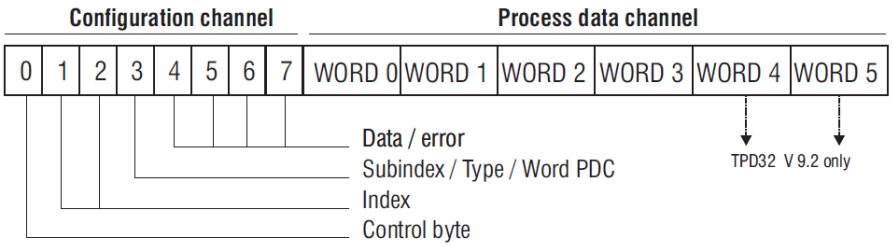
The errors do not follow the table of CC requests to the TPD32-EV, but instead are the ones shown below:

Function	Code	Description
Read	6	DbReadPar failed
Write	7	DbWritePar
All	5	Checksum error
All	8	Unknown command

For example, if you use a **profibus** card as Option 1 (SBI-PDP32), two bits of the SBI-PDP32 card control byte are used. Some information on profibus communication; the following data types are to be used:

Type	Value	Note
INT16	3	
INT32	4	
UINT16	6	
UINT32	7	
FLOAT	8	

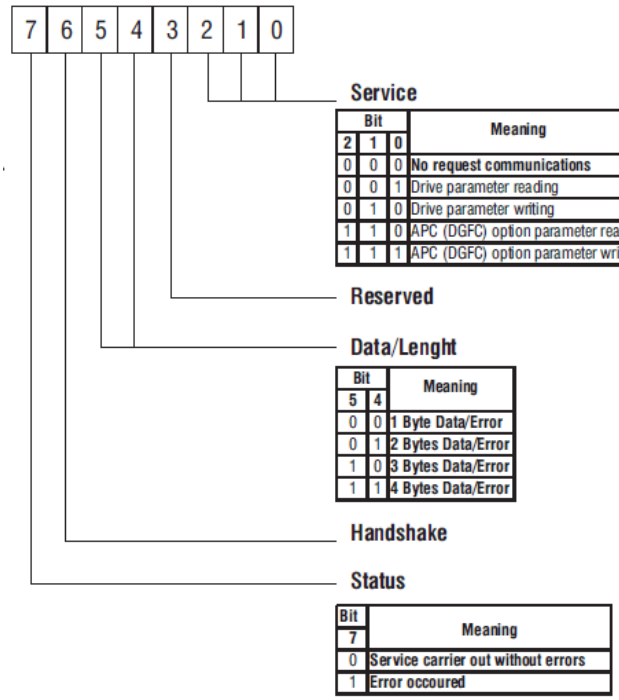
The SBI card uses a 20-byte frame. The first 8 bytes represent the configuration channel for the noncyclic data exchange, the other 12 (8) are the process data channel for cyclic exchange.



The bytes assignment is as follows:

Subindex/Type/ Word PDC: In case of service carried out towards the APC option, this field should contain the data type (see table type).

Control byte setting



Bit 2	Bit 1	Bit 0	Meaning
0	0	0	No request communications
0	0	1	Drive parameter reading
0	1	0	Drive parameter writing
1	1	0	APC (DGFC) option parameter reading
1	1	1	APC (DGFC) option parameter writing

Bit 5	Bit 4	Meaning
0	0	1 Byte Data/Error
0	1	2 Bytes Data/Error
1	0	3 Bytes Data/Error
1	1	4 Bytes Data/Error

Bit 7	Meaning
0	Service carrier out without errors
1	Error occoured

8. SYSTEM BUILT-IN FUNCTIONS AND FUNCTIONS BLOCKS

The APC300 firmware exposes some built-in functions, functions block that can be useful in order to speed-up program execution and to perform some tests that couldn't be developed with IEC-61131 languages.
These functions are listed in the '**target blocks"** windows inside the MDPlc development environment (Library toolbar).
Below is a list of these functions together with the related explanation;
INT MAX= +32767, DINT MAX= 0x7FFFFFFF; UINT MAX= 65535, UDINT MAX = 0xFFFFFFFF.

8.1 DBASE INTERFACE (1)

Following functions, functions block are used how interface with Database process.
Following functions, functions block are available only in tasks Boot, Background.

Type	"ST" invocation	Description
F	DINT:=**sysDBWriteIntegerPar**(DINT ipa, DINT val);	Write database integer parameter value
F	DINT:=**sysDBWriteIntegerFieldMin**(DINT ipa, DINT min);	Write database integer parameter field minimum
F	DINT:=**sysDBWriteIntegerFieldMax**(DINT ipa, DINT max);	Write database integer parameter field maximum
F	DINT:=**sysDBWriteIntegerFieldEu**(DINT ipa, DINT eu);	Write database integer parameter field engineering unit
F	DINT:=**sysDBWriteIntegerFieldScale**(DINT ipa, REAL scale);	Write database integer parameter field scale
F	DINT:=**sysDBWriteIntegerField**(DINT ipa, DINT field, DINT ival, REAL fval);	Write database integer parameter field x
F	DINT:=**sysDBWriteRealPar**(DINT ipa, REAL val);	Write database float parameter value
F	DINT:=**sysDBWriteRealFieldMin**(DINT ipa, REAL min);	Write database float parameter field minimum
F	DINT:=**sysDBWriteRealFieldMax**(DINT ipa, REAL max);	Write database float parameter field maximum
F	DINT:=**sysDBWriteRealFieldEu**(DINT ipa, DINT eu);	Write database float parameter field engineering unit
F	DINT:=**sysDBWriteRealFieldScale**(DINT ipa, REAL scale);	Write database float parameter field scale
F	DINT:=**sysDBWriteRealField**(DINT ipa, DINT field, DINT ival, REAL fval);	Write database float parameter field x

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	long val;
	}
	void **sysDBReadIntegerPar**(_FB_PAR * frame);
		Read database integer parameter value

Type	"ST" invocation	Description

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	long min;
	}
	void **sysDBReadIntegerFieldMin**(_FB_PAR * frame);
		Read database integer parameter field minimum

FB	typedef struct _FB_PAR { /* VAR_INPUT */ long ipa; /* VAR_OUTPUT */ long result; long max; void **sysDBReadIntegerFieldMax**(_FB_PAR * frame);	Read database integer parameter field maximum

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	long eu;
	}
	void **sysDBReadIntegerFieldEu**(_FB_PAR * frame);
		Read database integer parameter field engineering unit

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	float scale;
	}
	void **sysDBReadIntegerFieldScale**(_FB_PAR * frame);
		Read database integer parameter field scale

FB	typedef struct _FB_PAR { /* VAR_INPUT */ long ipa; long field; /* VAR_OUTPUT */ long result; long ival; float fval;	Read database integer parameter field x

Type	"ST" invocation	Description
	} void **sysDBReadIntegerField**(_FB_PAR * frame);	
FB	/* VAR_INPUT */ long ipa; /* VAR_OUTPUT */ long result; float val; } void **sysDBReadRealPar**(_FB_PAR * frame);	Read database float parameter value
FB	typedef struct _FB_PAR { /* VAR_INPUT */ long ipa; /* VAR_OUTPUT */ long result; float min; void **sysDBReadRealFieldMin**(_FB_PAR * frame);	Read database float parameter field minimum

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	float max;
	}
	void **sysDBReadRealFieldMax**(_FB_PAR * frame);
		Read database float parameter field maximum

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	long eu;
	}
	void **sysDBReadRealFieldEu**(_FB_PAR * frame);
		Read database float parameter field engineering unit

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	float scale;
	}
	void **sysDBReadRealFieldScale**(_FB_PAR * frame);
		Read database float parameter field scale

Type	"ST" invocation	Description

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	long field;
	/* VAR_OUTPUT */
	long result;
	long ival;
	float fval;
	}
	void **sysDBReadRealField**(_FB_PAR * frame);
		Read database float parameter field x

8.2 DBASE INTERFACE (2)

Following functions, functions block are used how interface with Database process.
Following functions, functions block are available only in tasks Parameters, read/write event.

Type	"ST" invocation	Description
F	DINT:=**sysDBPutIntegerPar**(DINT ipa, DINT val);	Write database integer parameter value
F	DINT:=**sysDBPutIntegerFieldMin**(DINT ipa, DINT min);	Write database integer parameter field minimum
F	DINT:=**sysDBPutIntegerFieldMax**(DINT ipa, DINT max);	Write database integer parameter field maximum
F	DINT:=**sysDBPutIntegerFieldEu**(DINT ipa, DINT eu);	Write database integer parameter field engineering unit
F	DINT:=**sysDBPutIntegerFieldScale**(DINT ipa, REAL scale);	Write database integer parameter field scale
F	DINT:=**sysDBPutIntegerField**(DINT ipa, DINT field, DINT ival, REAL fval);	Write database integer parameter field x **Field** / **Code** SCALE / 16
F	DINT:=**sysDBPutRealPar**(DINT ipa, REAL val);	Write database float parameter value
F	DINT:=**sysDBPutRealFieldMin**(DINT ipa, REAL min);	Write database float parameter field minimum
F	DINT:=**sysDBPutRealFieldMax**(DINT ipa, REAL max);	Write database float parameter field maximum
F	DINT:=**sysDBPutRealFieldEu**(DINT ipa, DINT eu);	Write database float parameter field engineering unit
F	DINT:=**sysDBPutRealFieldScale**(DINT ipa, REAL scale);	Write database float parameter field scale
F	DINT:=**sysDBPutRealField**(DINT ipa, DINT field, DINT ival, REAL fval);	Write database float parameter field x **Field** / **Code** SCALE / 16
FB	typedef struct _FB_PAR { /* VAR_INPUT */	Read database integer parameter value

Type	"ST" invocation	Description
	long ipa; /* VAR_OUTPUT */ long result; long val; } void **sysDBGetIntegerPar**(_FB_PAR * frame);	
FB	/* VAR_INPUT */ long ipa; /* VAR_OUTPUT */ long result; long min; void **sysDBGetIntegerFieldMin**(_FB_PAR * frame);	Read database integer parameter field minimum

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	long max;
	}
	void **sysDBGetIntegerFieldMax**(_FB_PAR * frame);
		Read database integer parameter field maximum

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	long eu;
	}
	void **sysDBGetIntegerFieldEu**(_FB_PAR * frame);
		Read database integer parameter field engineering unit

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	float scale;
	}
	void **sysDBGetIntegerFieldScale**(_FB_PAR * frame);
		Read database integer parameter field scale

FB	typedef struct _FB_PAR { /* VAR_INPUT */ long ipa; long field;	Read database integer parameter field x **Field** / **Code** VAL / 12 DEF / 13

Type	"ST" invocation	Description
	/* VAR_OUTPUT */ long result; long ival; float fval; } void **sysDBGetIntegerField**(_FB_PAR * frame);	<table><tr><td>MIN</td><td>14</td></tr><tr><td>MAX</td><td>15</td></tr><tr><td>SCALE</td><td>16</td></tr></table>

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	float val;
	}
	void **sysDBGetRealPar**(_FB_PAR * frame);
		Read database float parameter value

FB	typedef struct _FB_PAR
	{
	
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	float min;
	}
	void **sysDBGetRealFieldMin**(_FB_PAR * frame);
		Read database float parameter field minimum

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	float max;
	}
	void **sysDBGetRealFieldMax**(_FB_PAR * frame);
		Read database float parameter field maximum

FB	typedef struct _FB_PAR
	{
	/* VAR_INPUT */
	long ipa;
	/* VAR_OUTPUT */
	long result;
	long eu;
	}
	void **sysDBGetRealFieldEu**(_FB_PAR * frame);
		Read database float parameter field engineering unit

<table>
<tr><th>Type</th><th>“ST” invocation</th><th>Description</th></tr>
<tr><td>FB</td><td>/* VAR_INPUT */
long ipa;
/* VAR_OUTPUT */
long result;
float scale;
void sysDBGetRealFieldScale(_FB_PAR * frame);</td><td>Read database float parameter field scale</td></tr>
<tr><td>FB</td><td>typedef struct _FB_PAR
{
/* VAR_INPUT */
long ipa;
long field;
/* VAR_OUTPUT */
long result;
long ival;
float fval;
}
void sysDBGetRealField(_FB_PAR * frame);</td><td>Read database float parameter field x
<table><tr><th>Field</th><th>Code</th></tr><tr><td>VAL</td><td>12</td></tr><tr><td>DEF</td><td>13</td></tr><tr><td>MIN</td><td>14</td></tr><tr><td>MAX</td><td>15</td></tr><tr><td>SCALE</td><td>16</td></tr></table></td></tr>
</table>

8.3 KEYPAD INTERFACE

Following functions, functions block are used how interface with Keypad process.
Following functions, functions block are available only in some Task, for every function verify in what task is available.

Type	“ST” invocation	Description
F	DINT:=**sysKpdWriteMsg**(WORD message, WORD code, WORD timeout);	Write Keypad Message from task Boot or Background
F	DINT:=**sysDBParDescrUpdate**(WORD dummy);	Update parameters description from task Background. Update parameter and description configured with system internal variable sysPar1Ipa , sysPar1Descr .. sysPar16Ipa , sysPar16Descr .
F	DINT:=**sysDBParDescrUpdateEvent**(WORD dummy);	Update parameters description from R/W Events. Update parameter and description configured with system internal variable sysPar1Ipa , sysPar1Descr .. sPar16Ipa , sysPar16Descr .
F	DINT:= **sysKeypadStartupParList** (WORD ipa, WORD listindex);	Update Start-up parameters list from task Boot or Background.
F	DINT:= **sysKeypadSetDispList** (UINT mode, UINT pos); Update mode WORD mode, function code: 0 = append, add at end list 1 = insert, insert before "pos" (1=first element) 2 = delete, delete "pos" element 3 = reset list and append WORD pos, position for insert and delete	Set keypad display list from task Boot or Background. A new element is written by the mdplc application in the sysDispListIpa and sysDispListDescr before calling this function. The max number of elements is MAX_DISPL_PAR (16). Extra elements are discarded.

sysKpdWriteMsg
Parameters:
sysKpdWriteMsg.message. It selects one of the messages configured with system internal variable sysMsgPlc1.. sysMsgPlc4.
sysKpdWriteMsg.code. It sets number showed on display together with the message.
sysKpdWriteMsg.timeout. It sets the time for which the message is visualized.

sysKeypadStartupParList
If there is a Mdplc Application inside Start-up Wizard there is the item “Set application par ?”
Using this function is possible set the parameters inserted inside this submenu. This submenu is an array of size 10.
After power-on all items of this array are 0xffff (marker for ipa not available).
The application have to write required Ipa of parameter defined by user or system parameter.
Ipa of not existing parameter are managed like ipa not available.
The first item set like ipa not available provokes exit from submenu.
Parameters:
sysKeypadStartupParList.ipa. It set index parameter
sysKeypadStartupParList.listindex. It set list index (0..9)
Return value:
0 => No error
1 => List index greater than array size

8.4 LED INTERFACE

Following **Embedded functions:**
Orange LEDs S0 and S1 can be managed directly by the MdPlc application by means of embedded functions *sysLedS0* and *sysLedS1* .

```
res := sysLedS0(val);
res := sysLedS1(val);
```

the value “val” is a dword, if 0 the corresponding LED is off, if 1 the LED is on.

F	DINT:=**sysLedS0**(DINT dummy);	Manage S0 led of the APC300 Board. Input vars num: 1 val : DINT (* 0 = Off , 1 = On *) Result type: DINT
F	DINT:=**sysLedS1**(DINT dummy);	Manage S1 led of the APC300 Board. Input vars num: 1 val : DINT (* 0 = Off , 1 = On *) Result type: DINT

8.5 ARITHMETIC 16 BITS

Following functions, functions block are available in every task.

Type	"ST" invocation	Description
F	INT:=**ADD16_LIM**(INT op1, INT op2);	Sum of two signed 16 bit, limited to signed 16 bit
F	WORD:=**ADDU16_LIM**(WORD op1, WORD op2);	Sum of two unsigned 16 bit, limited to unsigned 16 bit
F	DINT:=**ADD32_LIM**(DINT op1, DINT op2);	Sum of two signed 32 bit, limited to signed 32 bit
F	DWORD:=**ADDU32_LIM**(DWORD op1, DWORD op2);	Sum of two unsigned 32 bit, limited to unsigned 32 bit
F	INT:=**SUB16_LIM**(INT op1, INT op2);	Difference of two signed 16 bit, limited to signed 16 bit
F	WORD:=**SUBU16_LIM**(WORD op1, WORD op2);	Difference of two unsigned 16 bit, limited to unsigned 16 bit
F	DINT:=**SUB32_LIM**(DINTop1, DINT op2);	Difference of two signed 32 bit, limited to signed 32 bit
F	DWORD:=**SUBU32_LIM**(DWORD op1, DWORD op2);	Difference of two unsigned 32 bit, limited to unsigned 32 bit
F	DINT:=**MUL32**(INT op1, INT op2);	Mult of two signed 16 bit to signed 32 bit
F	DWORD:=**MULU32**(WORD op1, WORD op2);	Mult of two unsigned 16 bit unsigned 32 bit
F	DINT:=**MUL32_LIM**(DINT op1, DINT op2);	Mult of two signed 32 bit, limited to signed 32 bit
F	DWORD:=**MULU32_LIM**(DWORD op1, DWORD op2);	Mult of two unsigned 32 bit, limited to unsigned 32 bit
F	INT:=**DIV32**(DINT x, INT y);	Divide 32 bit signed by 16 bit signed , result is limited to 16 bit
F	WORD:=**DIVU32**(DWORD x, WORD y);	Divide 32 bit signed by 16 bit signed , result is limited to 16 bit
F	INT:=**MOD32**(DINT x, INT y);	Module 32 bit signed over 16 bit signed , result is limited to 16 bit
F	WORD:=**MODU32**(DWORD x, WORD y);	Module 32 bit over 16 bit signed , result is limited to 16 bit
F	INT:= **DIV32_LIM**(DINT x, INT y);	Divide 32 bit signed by 16 bit signed , result is limited to 16 bit
F	WORD:=**DIVU32_LIM**(DWORD x, WORD y);	Divide 32 bit unsigned by 16 bit unsigned , result is limited to 16 bit
F	DINT:=**SHA_R**(DINT inp, INT cnt);	Arithmetic shift right. input signed 32 bits, output signed 32 bits
F	INT:=**HI_WORD**(DINT inp);	Return high word (16 bits) of input value (32 bits)
F	INT:=**LO_WORD**(DINT inp);	Return low word (16 bits) of input value (32 bits)

9. ALARM GENERATION

System alarms

System alarms include alarms generated after a problem detected by the firmware (for example, loss of synchronization in DPRAM), FastLink communication errors, etc.
These alarms are generated and reported by parameters:

[4770] “First alarm”
[4840] “Alarm state lo”.
[4842] “Alarm state hi”.

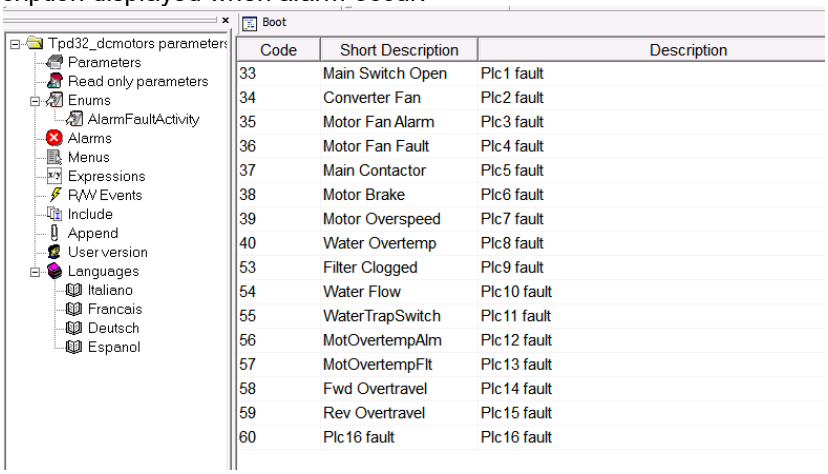
The “First alarm” code is transmitted to the TPD32-EV, which generates an alarm “Opt2 with code” = code of First Alarm.

Example: if APC300 has “FastLink” alarm active → TPD32-EV shows “Opt2 code 28”.
These alarms are also reported by means of system parameters:

- sysFirstAlarm
- sysAlarmHi (Status Hi)
- sysAlarmLo (Status Lo)

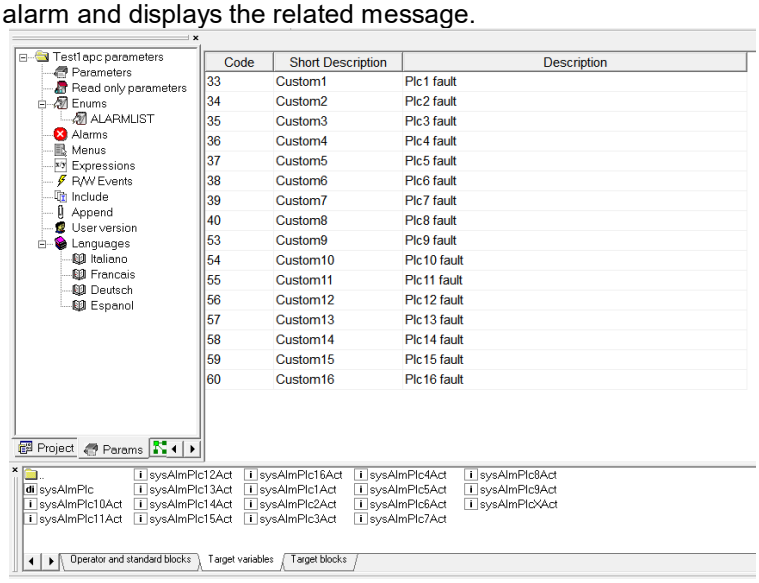
Application Alarms:

The APC300 card manages 16 application alarms. The MdPlc application generates inside application Tasks these by means of **sysAlarmsPlc** variables.
It is possible change description displayed when alarm occur.



Code	Short Description	Description
33	Main Switch Open	Plc1 fault
34	Converter Fan	Plc2 fault
35	Motor Fan Alarm	Plc3 fault
36	Motor Fan Fault	Plc4 fault
37	Main Contactor	Plc5 fault
38	Motor Brake	Plc6 fault
39	Motor Overspeed	Plc7 fault
40	Water Overtemp	Plc8 fault
53	Filter Clogged	Plc9 fault
54	Water Flow	Plc10 fault
55	WaterTrapSwitch	Plc11 fault
56	MotOvertempAlm	Plc12 fault
57	MotOvertempFlt	Plc13 fault
58	Fwd Overtravel	Plc14 fault
59	Rev Overtravel	Plc15 fault
60	Plc16 fault	Plc16 fault

Information on alarms is reported by the APC300 via First Alarm and 4842 Alarm state hi.
The TPD32-EV is also transmitted the string with a description of the alarm inserted in the MdPlc application. The TPD32-EV then generates an MDPLC alarm and displays the related message.



Code	Short Description	Description
33	Custom1	Plc1 fault
34	Custom2	Plc2 fault
35	Custom3	Plc3 fault
36	Custom4	Plc4 fault
37	Custom5	Plc5 fault
38	Custom6	Plc6 fault
39	Custom7	Plc7 fault
40	Custom8	Plc8 fault
53	Custom9	Plc9 fault
54	Custom10	Plc10 fault
55	Custom11	Plc11 fault
56	Custom12	Plc12 fault
57	Custom13	Plc13 fault
58	Custom14	Plc14 fault
59	Custom15	Plc15 fault
60	Custom16	Plc16 fault

Alarm Reset

If DPRAM dialogs are active (synchronization active), resetting the alarms on the TPD32-EV also resets the alarms on the APC300. If the causes of the alarm are removed, resetting the alarms deletes all of the alarms on the TPD32-EV as well as on the APC300.
Resetting the alarms with the APC300 configurator or APC300 parameter can reset the FirstAlarm but a TPD32-EV remains until an alarm reset is done on the TPD32-EV .

Alarm Activity

It is possible writing a system internal variable configure the activity performed by drive after alarm activation.

The available Activity are:

Code	Activity
0	Ignore
1	Warning
2	Disable
3	Stop
4	Quick stop

Alarm Codes

Alarms can be system alarms or application alarms (see Managing Alarms). The alarm codes (except for PLC codes, which are programmable) appear on the TPD32-EV keypad. In case of APC alarm, the first line shows “OPT2 failure,” while the code is on the second line.
The alarm codes also appear on the WEG_eXpress connected to the APC300.

List of APC300 alarm codes:

No..	Name	Description
21	External fault	An Ext-dig. input is programmed as an external alarm
27	ExtIO fault	Communication problem with external module
28	FastLink fault	Communication problem with FastLink
29	Dpram fault	Communication problem with DPRAM
33	Plc1 fault	Alarm generated by MDPLC application. For more information, see the documentation for the specific application.
34	Plc2 fault	
35	Plc3 fault	
36	Plc4 fault	
37	Plc5 fault	
38	Plc6 fault	
39	Plc7 fault	
40	Plc8 fault	
41	Watchdog	System alarm. May be due to a change in card configuration (example: downloading of a PLC application, etc.) or card malfunction
42	Trap error	
43	System error	
44	User error	
45	Param error->Press ESC	Error during activation of parameters database saved in flash
46	Load def par ->Press ESC	Load factory parameters, normal if at first startup or loading of new firmware. If it appears when the card is already operative, it means that there is a problem in the parameters database saved in Flash
47	Plc cfg error ->Press ESC	During loading of MDPLC application, if displayed the MDPLC application is not executed.
48	Load def plc ->Press ESC	Normal if at first startup or loading of new firmware. If it appears when the card is already operative, it means that there is a problem in the parameters database saved in Flash
49	Key failed->Press ESC	Wrong PLC key.
53	Plc9 fault	Alarm generated by MDPLC application. For more information, see the documentation for the specific application.
54	Plc10 fault	
55	Plc11 fault	
56	Plc12 fault	
57	Plc13 fault	
58	Plc14 fault	
59	Plc15 fault	
60	Plc16 fault	
100 (*)	DPRAM fault TPD	DPRAM error detected by TPD control

(*) 100 managed only on TPD32-EV side (TPD32-EV keypad or alarm).

10. MULTI LANGUAGE

Inside Mdplc tool there is the possibility to set parameter, menu, Enum, alarm, Eunit description of one language. The use of English language is recommended for maintain first language of Application equal to first language of system.
Inside Mdplc tool there are four dictionary of I,F,D,E where are listed all parameter, menu, Enum, alarm, Eunit description in English language and it is possible insert translated description.
With option "*Project Option\Parameters\Multi language*" it is possible select to download only one language or download more language.
The developer of the application can select number of languages download on the drive. The selection is done with field Active inside Languages window.
If on system in selected language X but this language is not present inside application than application parameter, menu, Enum, alarm, Eunit are displayed in English language.

11. APPENDIX

11.1 UNITS OF MEASURE

Units description
The units of measure are described in the system internal variable chapter, see chapter 5.1 e 5.2 SCALING OF …. CONTROL VARIABLES.

11.2 APPLICATION EXAMPLE

In preparation..

For a simple application example see How to Write MDPLC applications with ADV200.

11.3 STANDARD LIBRARY

Some libraries have been developed containing Functions and FB (Function Block), which can be easily recalled inside the PLC program.
Each library is oriented towards specific operations.

For APC300 labrary is in preparation..

For a simple application example see ADV200 Library How to Write MDPLC applications with ADV200.

You can import and use some Functions or functions Block of the ADV200 libraries.

11.4 SOFTWARE STRUCTURE

Software Structure

11.4.1 Code generated by the Mdplc

Through the MDPlc tool the user can write the code contained in the available tasks, see chapter "**TASKS**". For each task the MDPlc compiler generates three different object codes:

- EXE code which performs the operations specified by the programmer in the task. With the Fast Plc task, this code is performed in the Task1 interrupt of system firmware. With Slow Plc task, this code is performed inside the Task2 interrupt. With the Parameters Plc task, this code is performed Dbase process of system firmware. With Background Plc task, this code is performed in the Background process of system firmware. With Boot Plc task, this code is performed in the Init process of system firmware.
- IN code which is generated automatically by the MDPlc compiler.
  This code copies the task input variables which must have an image. For Fast, Parameters, Background Plc Task,this code is performed before execution of EXE code. For Slow This code is performed in Task1 interrupt system firmware only when there is synchronization with Task2 interrupt after Fast Plc Task management.
- OUT code which is generated automatically by the compiler. The OUT code copies the content of the image variables inside the output variables defined inside the PLC program. The generated EXE code, in fact, works with an image of the output variables (if the variables have an image). For Fast, Parameters, Background Plc Task,this code is performed after execution of EXE code. For Slow This code is performed in Task1 interrupt system firmware only when there is synchronization with Task2 interrupt before Fast Plc Task management.

11.4.2 DOWNLOAD CODE

In this chapter there are the sequence to download a MdPlc application in a APC300 Board.
The first step is to download the application firmware (eg. "ApplSampVxx_yyyy") in the drive, the firmware module available to download in the APC300 board.

In your project, after the compilation if you execute the MDPLC command "Project/Generate application files" , MDPLC generate four file:

```
c:\MdplcProjects\test>echo off
```

```
'
'
' Batch generazione applicazione
'
'
Generated output file: APC300_1_0_0_Fw_Lang_tpd32_dcmotors___A1.fl2
Generated output file: APC300_1_0_0_Fw_Lang_tpd32_dcmotors___A2.fl2
Generated output file: tpd32_dcmotors___A1.fl2
Generated output file: tpd32_dcmotors___A2.fl2
```

The files are generate in the same project folder:

1- "APC300_1_0_0_Fw_Lang_tpd32_dcmotors___A1.fl2" contain APC300 fw V1_0_0 code + "test" program code in the application 1 memory.
2- "APC300_1_0_0_Fw_Lang_tpd32_dcmotors___A2.fl2" contain APC300 fw V1_0_0 code + "test" program code in the application 2 memory.
3- "test___A1.fl2" contain the core ready to be loaded in the in application 1 memory with APC300 fw V1_0_0
4- "test___A2.fl2" contain the core ready to be loaded in the in application 2 memory APC300 fw V1_0_0

To download and start APC300 fw V1_0_0 with the new application there are two steps:

Step 1: download the firmware you can use WEG_eXpress:

With download command you can download all the files in the table.
The procedure is the following:

1.Power on the drive. Check that there are no message that stop the drive starting sequence, (eg. "Load default"). In this case unlock the situation (eg. press "Esc").

2. Start WEG_eXpress and connect to APC300 board.

3. Select "Service->Download Firmware"

4. select the file to download with the button "Browse"

5. Press the button "Download". In the download window there are first the message "Erasing Flash" and then the message "loading". In the right part there are a download counter, all the download take several minutes (about 5-6min).

6. Wait the message "Load complete" and press OK.

7. The drive restart, if there are a message that stop the starting sequence press Esc to finish the starting sequence.

Step2: download the correct application default parameter configuration:.

To perform this with WEG_eXpress the sequence is:

1. Execute a "Load default drive value" command then a "Save parameter into target" and a "Drive reset".

2. Menù "DRIVE CONFIG\554-Access" easy, check or select "expert".

3. Menù "DRIVE CONFIG\558-Applicaion select" none check or select "Application 1" o "Application 2".

4. Execute a "Save parameter into target" and "Drive reset". Now the application is running..

Note:
If drive is enabled the mdplc show following message:
"System stand-by require".
Disable drive and repeat command.

11.4.3 Parameter generated by MDPLC

The WEG_eXpress configurator allows the user to carry out the following tasks: parameterization of the application downloaded in the drive of the target, supervision of drive activities, diagnostics and service functions. To carry out these functions, the WEG_eXpress configurator uses the parameter files produced by MDPlc during compilation.

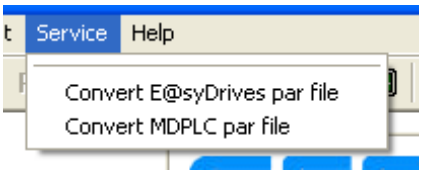
NOTE:

With APC300, WEG _eXpress must be used. The parameter file generated by MDPLC in old format *.par, must be converted in the *.Gfs format. The procedure is in the WEG _eXpress manual, here the most important information:

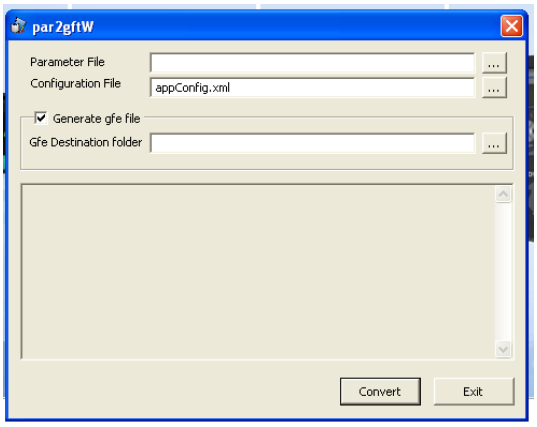
Convert MDPLC par file

From WEG _eXpress:

1:



2:



This tool is to convert a par file created by MDPLC.
This program has the following input boxes:

- Parameter File : path of the input file (.par)
- Configuration File : path of the configuration file (AppConfig.xml)
- gfe destination folder: path of the folder containing the .gfe output file (optional)

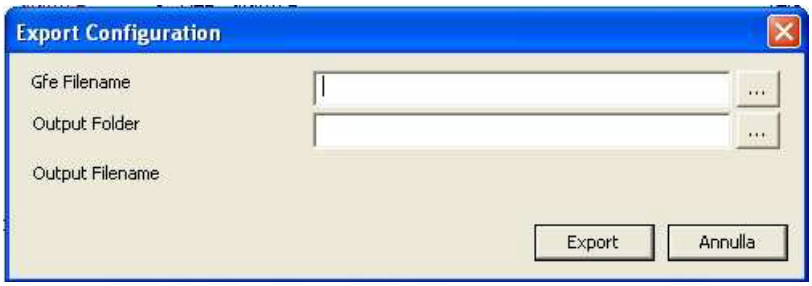
It builds a .xml file extended as .gfe (Output File) starting from a .par file, and a .gft file of the MDPLC application.
Configuration File path is already set.

11.4.4 MDPLC WEG _eXpress Parameter Export

If you want to distribute a MDPLC application parameter file in WEG _eXpress it's necessary to Export and Import the Configuration.
The complete procedure is in the WEG _eXpress Manual, here the most important informations:

Import Export configuration

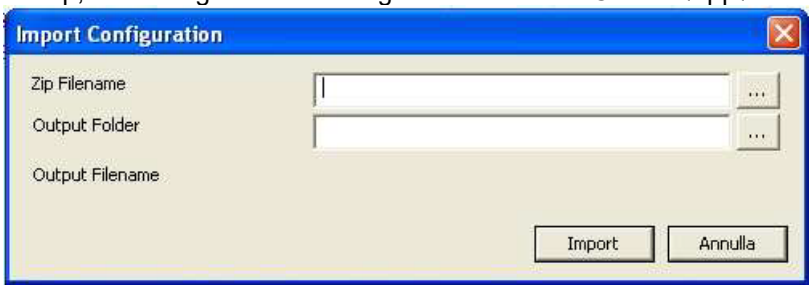
With the "Import Configuration" and "Export Configuration" commands on the "File" menu, you can import and export GFE files with assigned GFT in a single ZIP packet.
This function lets you reuse the configuration and the definition of a specific custom device created by the user.
It is used to export to another PC a device not inserted in the standard catalog, typically a device written with MDPLC program.



This window tool has the following input boxes:

- Gfe FileName : path of the input file (.gfe)
- Output Folder: path of the folder containing the .zip output file

It builds a .zip file containing the .gfe file and the relative .gft file, inclusive of all over folders until Catalog\.
For example, if the saved file.gfe refers to .gft file ADV200_5_X_0.gft, located in C:\Programmi\Weg\Catalog\Custom\App\ADV200_5_X_0, then the program will generate a .zip file named saved file.zip, containing the file saved.gfe and the folder Custom\App\ADV200_5_X_0\ADV200_5_X_0.gft



This window tool has the following input boxes:

- Zip FileName : path of the input file (.zip), already made by exportation option
- Output Folder: path of the .gfe file just extracted

It unzips the .zip file checked, then places the .gft file in the original position and saves the .gfe file in the desired path. All the existing files will be overwritten.

How to write the applications with the MDPlc on APC300
Series: APC300
Revision: 0.2
Date: 15-12-2022
Code: 1S5WMDEN

WEG Automation Europe S.r.l.
Via Giosuè Carducci, 24
21040 Gerenzano (VA) · Italy

www.weg.net

Driving efficiency and sustainability