

EzProg-I Tools

(Version 4.6)

EzProg



ICP DAS CO., LTD.

Warranty

All products manufactured by ICPDAS Inc. are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICPDAS Inc. assumes no liability for damages consequent to the use of this product. ICPDAS Inc. reserves the right to change this manual at any time without notice. The information furnished by ICPDAS Inc. is believed to be accurate and reliable. However, no responsibility is assumed by ICPDAS Inc. for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Copyright 1997-2009 by ICPDAS Inc., LTD. All rights reserved worldwide.

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

License

The user can use, modify and backup this software on a single machine. The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

Technical Support

If you have problems about using the product, please contact ICP DAS Product Support.

Email: Service@icpdas.com

Table of Contents

EzProg-I Utilities

1	Introduction.....	5
1.1	EzProg-I Software Development Resources.....	9
2	EzProg-I framework.....	13
2.1	Application framework.....	13
2.1.1	EzProg-I Register.....	14
3	EzTemplate.....	17
3.1	EzTemplate Setting.....	17
3.1.1	Pages.....	17
3.1.2	Programming interfaces.....	19
3.2	Priority Levels.....	23
4	EzHMI.....	26
4.1	EzHMI ActiveX overview.....	26
4.2	Settings.....	32
4.2.1	Visual Studio 2008 IDE.....	32
4.2.2	Refresh Time.....	35
4.2.3	Language Switching.....	36
4.2.4	Register linking.....	38
4.3	EzHMI ActiveX Controls.....	39
4.3.1	LED Control.....	39
4.3.2	Switch.....	43
4.3.3	EzHMI Lable.....	49
4.3.4	EzHMI ColorEdit.....	54
4.3.5	EzHMI ButtonST.....	58
4.3.6	EzHMI Image Control.....	63
4.3.7	EzHMI ColorCheck.....	73
4.3.8	EzHMI ColorRadio.....	78
4.3.9	EzHMI EzKnob.....	83
4.3.10	EzHMI EzSlider.....	87
4.3.11	EzHMI EzList.....	91
4.3.12	EzHMI Position.....	94
5	EzConfig Utility.....	95
5.1	Introduction.....	95
5.2	Main properties.....	96
5.3	Slot scan and IO register mapping.....	98
5.4	Module and channel configuration.....	100
5.4.1	Digital input configuration.....	100
5.4.2	Digital output configuration.....	102
5.4.3	Digital IO configuration.....	104
5.4.4	FRnet configuration.....	106
5.4.5	Analog input configuration.....	109
5.4.6	Analog output configuration.....	111
5.5	Default startup settings.....	115

5.6	Registry Key Editor	117
6	EzGo	120
6.1	Introduction.....	120
6.2	Using EzGo.....	120
6.2.1	Selection window.....	121
6.2.2	Initialization and configuration window.....	122
6.2.3	Basic Operation: Independent axis motion.....	132
6.2.4	Advance motion features: Multi-axis interpolation	137
7	EzMake	143
7.1	Introduction.....	143
7.1.1	Main user interface	143
7.2	Initial Table.....	144
7.2.1	Create a new initialization table.....	145
7.2.2	Modifying a initialization table.....	146
7.2.3	Open an initialization table file.....	148
7.2.4	Remove an initialization table from the tree view	149
7.2.5	Downloading of an initialization file	149
7.3	Macro Program Files (MP Files)	150
7.3.1	Create a new macro file	150
7.3.2	Adding a macro form to a macro file.....	151
7.3.3	Open a macro file.....	153
7.3.4	Writing macros (motion commands)	154
7.3.5	Downloading and executing a macro file	157
7.4	Interrupt Service Routine (ISR) macro	159
7.5	Project Files	160
7.5.1	Create a new project file	160
7.5.2	Downloading a project file.....	163
7.6	Macro motion commands	164
7.6.1	Basic Setting Functions.....	165
7.6.2	Status Functions.....	166
7.6.3	FRnet DIO Functions.....	166
7.6.4	Auto Home Functions.....	166
7.6.5	Axis Move Functions.....	167
7.6.6	Interpolation Functions.....	167
7.6.7	Synchronous Action Functions.....	169
7.6.8	Continuous Interpolation Functions.....	169
7.6.9	Interrupt Control Functions.....	170
7.6.10	Other Functions.....	170
7.6.11	Macro Program Functions.....	171

1 Introduction

EzProg

Advanced PAC solution for machine automation



EzProg-I development suit is a software package for the windows CE 5.0 /6.0 platform consisting of numerous utilities and libraries which assist the system developer in developing a control system on the PAC in a short period of time. The development suit has got the following advantages for the user:

1. Simple and visually appealing I/O monitoring HMI can be created by dragging and dropping of ActiveX objects, without the need of any programming.
2. Plc like programming by providing a predefined multitasking structure.
3. Extensive motion control libraries and utilities for writing motion control macros which make it easy to write application programs for ICPDAS's multi-axis motion control cards.

The **EzProg-I development suite** comprises:

- Human machine interface (HMI) objects (ActiveX)
- Configuration tools and utilities
- IO libraries
- Motion control tools and libraries
- Libraries for serial and network communications

Powerful and Easy to Use

EzProg-I is a total solution for system configuration, logic programming and HMI design for manufacturers or control system designers. Engineers who are familiar with programming PLC systems can easily migrate to EzProg-I and become acquainted with the software solution. EzProg-I makes it much easier for customers to integrate PLC and IT technologies.

The EzProg-I package contains many kinds of development tools and libraries such as EzConfig, EzGo, EzMake, EzHMI, EzLIB and EzCore. Based on these development resources, customers can directly configure and test the PAC's I/O channels and motion control modules without any additional programming effort. Moreover, EzProg-I simplifies the I/O instructions and provides I/O mapping table equivalent to the PLC system. It assists the system developer to design, construct and test its control system.

EzProg-I Applications

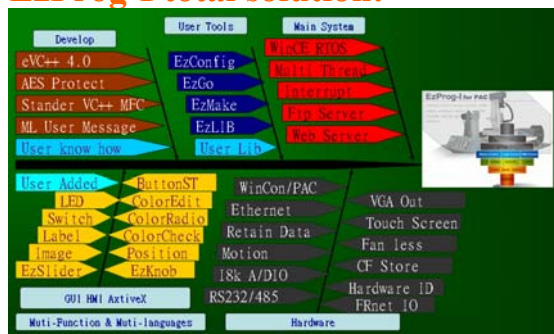
I/O monitoring and logic control:

Building automation, SCADA, factory automation

Motion control:

Carving machine, Cutter, Laser carving, Laser cutting, Graphic plotter, Tapping machine, Dispensing machine, Welding machine, Drilling machine, Punch machine, X-Y-Z table control, coil wiring machine, automatic machine control

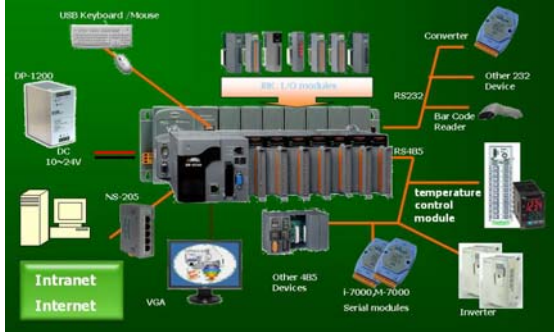
EzProg-I total solution:



- WinCE 5.0/6.0 real time OS
- Embedded low power consumption controller
- EzProg-I development tools: EzConfig, EzHMI, EzGo, EzMake and EzLIB.
- Microsoft VC++ IDE development environment (Visual Studio 2008).
- Software protection and license management

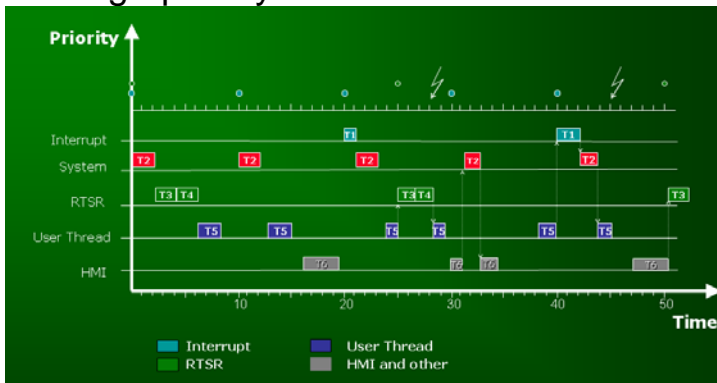
Integration of software and hardware:

The EzProg-I was integrated on the hardware PAC (Programmable Automation Controller), The PAC offer a lot of IO modules, communication interfaces, computer interfaces, data storage device and so on. That can satisfy the need of industrial automatic applications.



- Standard USB, VGA interface
- Touch screen support
- Data storage (1GB CF or Micro SD)
- Great number of IO modules available
- Multi-axis motion control modules
- Easy to use API command
- Various communication interfaces: Ethernet, RS-232 and RS-485.

Real time framework: EzProg-I priority level



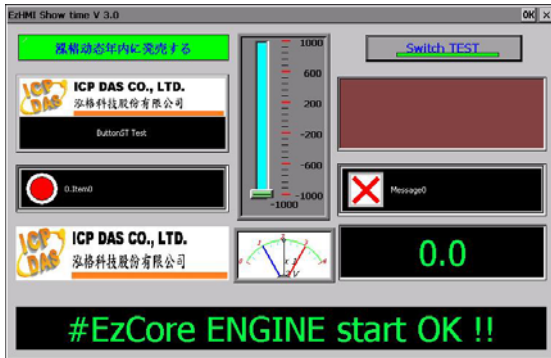
The EzProg-I is a real time development framework. The framework supports hardware interrupt, eight software interrupt service routine RTSR and eight user threads. That can design multi-thread control procedures on the WinCE RTSO. The user can design control function easily. Base on this framework, those multi-thread control procedures will be executed by priority one by one.

EzProg-I priority levels:

1. Interrupt
2. System task
3. RTSR task
4. User thread
5. HMI

Easy use to design GUI (HMI):

EzHMI provides a number of ActiveX controls which allows the programmer to create a graphic interface on a WinCE system which an operator can use to control and monitor machines and plants. A user interface can therefore be designed within a short period of time especially when it comes to displaying digital and analog I/O data of the ICPDAS 8000 series.



EzHMI ActiveX properties:

- Direct display of IO status
- Direct display of register values
- Multilanguage support
- Support flashing (alarm)
- Picture movement
- Font setting
- Color setting
- Enable/disable objects

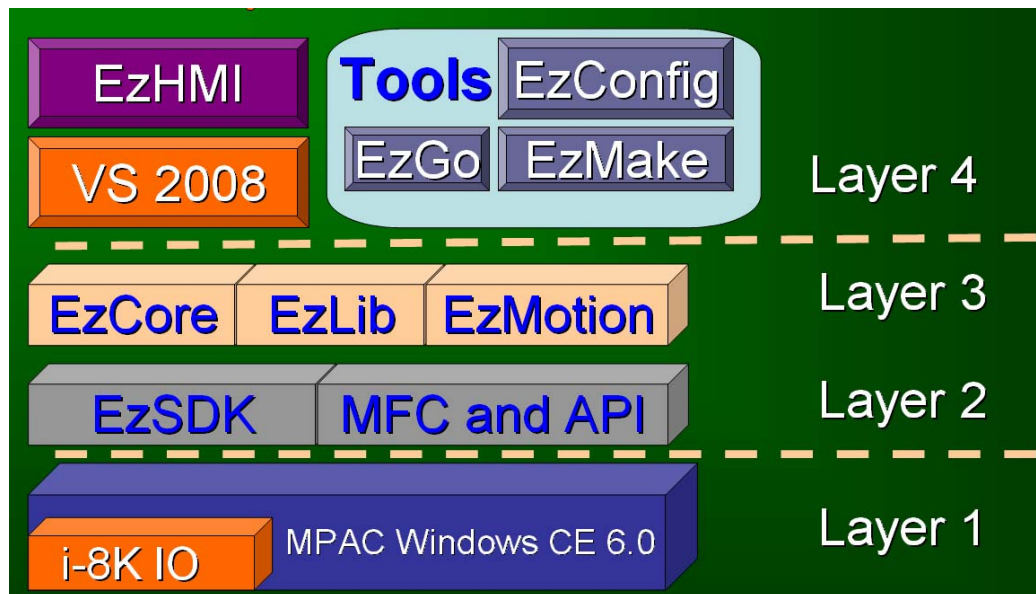
1.1 EzProg-I Software Development Resources

Requirements such as high performance, easy integration, extensibility, fast software development and short time to market are an increasing demand of the industry. ICPDAS offers the hardware and software solution to meet the demands. The EzCore engine provided by ICPDAS is a development kit to simplify and reduce the software development expenditure.

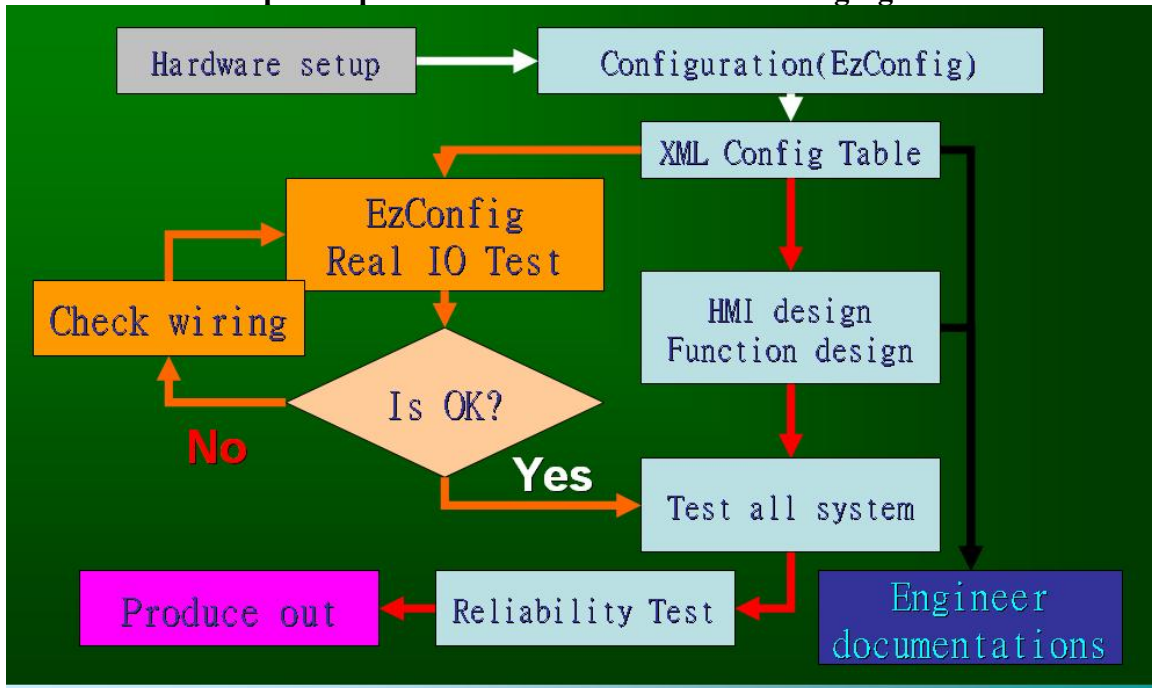
EzProg-I is a developing toolkit consisting of libraries, utilities and HMI controls for the Windows CE 6.0 platform.

The basic structure of EzProg-I:

- The lowest level consists of the real time WinCE operation system and the hardware driver
- IO APIs and the MFC APIs are part of the next layer
- One layer up is the main application layer comprising the EzCore, EzLib and EzMotion libraries.
- The top layer is made up of the control program compiled by VS2008, the human machine interface (EzHMI) and configuration and testing utilities



The normal development procedure is shown on the following figure.

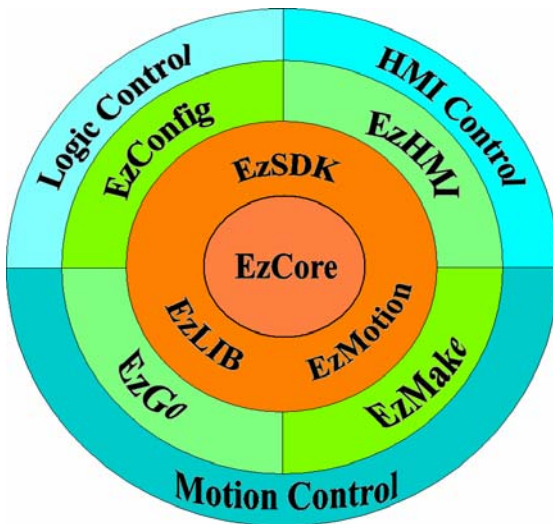


Libraries and utilities

EzCore

EzCore is the engine driving EzProg-I and it is responsible for the communication between the hardware (I/O), HMI controls and the registers. All libraries are compatible with Visual Studio 2008.

EzCore has a real time scanning engine which supports hardware interrupt and direct access of IO registers and other system related registers. EzCore offers eight real time interrupt service routine RTSR (similar to a task in a multitasking programmable logic controller) and eight user threads.



EzCore system variables:

	Specification	Register	Register number	Data type	Size	Range	
Slot Device Register	Digital Input	X	Local DI:	0 ~ 777	bit	1 bit	true / false
			FRNet DI:	1000 ~ 7777			
	Digital Output	Y	Local DO:	0 ~ 777	bit	1 bit	true / false
			FRNet DO:	1000 ~ 7777			
Analog Output	AO	Local AO:	0 ~ 511	float	4 bytes	3.4E +/- 38	
Analog Input	AI	Local AI:	0 ~ 511	float	4 bytes	3.4E +/- 38	
Software Register	Timer	T	None Retain:	1 ~ 299	bit	1 bit	true / false
	Counter	C	None Retain:	1 ~ 511	bit	1 bit	true / false
			Retain:	512 ~ 1023			
	Flag	M	None Retain:	1 ~ 6999	bit	1 bit	true / false
			Retain:	8192 ~ 15999			
	Step	S	None Retain:	1 ~ 8191	bit	1 bit	true / false
	long integer	D	None Retain:	1 ~ 3599	long integer	4 bytes	-2,147,483,648 to 2,147,483,647
			Retain:	4096 ~ 7999			
	BYTE	B	None Retain:	1 ~ 699	unsigned char	1 byte	0 to 255
			Retain:	1024 ~ 2047			
	WORD	W	None Retain:	1 ~ 1023	unsigned short	2 bytes	0 to 65,535
			Retain:	1024 ~ 1999			
DWORD	DW	None Retain:	1 ~ 4095	unsigned long	4 bytes	0 to 4,294,967,295	
		Retain:	4096 ~ 8191				
Float	F	None Retain:	1 ~ 1899	float	4 bytes	3.4E +/- 38	
		Retain:	2048 ~ 3999				
Special Type	DB	None Retain:	1 ~ 49				
Message	MSG	Retain:	1 ~ 249	30 wchar_t	60 bytes	30 unicode char	

EzCore main function:

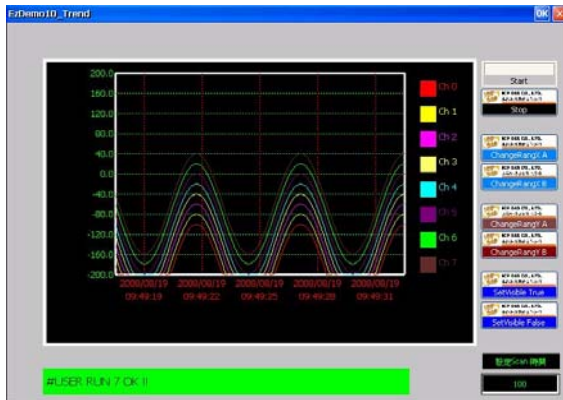
EzCore has a real time scanning engine which supports hardware interrupt and direct access of IO registers and other system related registers. EzCore offers eight real time interrupt service routine RTSR (similar to a task in a multitasking programmable logic controller) and eight user threads.

EzHMI ActiveX

EzHMI ActiveX controls allows the programmer to design a user interface on a WinCE system for monitoring and controlling purposes. The ActiveX controls can be directly linked to IO registers for displaying or manipulating of IO data. In addition the ActiveX controls support Multilanguage. EzHMI objects use the EzCore platform to update data at system run time.

EzLIB

EzLib is a collection of reusable software components and assists software developers to write application programs for the Window CE platform.



APIs for:

- Data format transformation
- Date and time
- Read/Write file
- Context drawing
- FTP communication
- TCP/IP communication
- Trend line graph

Motion Control development resources

EzProg-I supports the following 2 axis and four axis motion modules:
i8092F, i8094, i8094F, i8094A, i8094H.

EzGo and EzMake are utilities for configuring, programming and testing the motion modules. For more information refer to the manuals for motion control.

2 EzProg-I framework

2.1 Application framework

The EzProg-I main framework has the following layout:

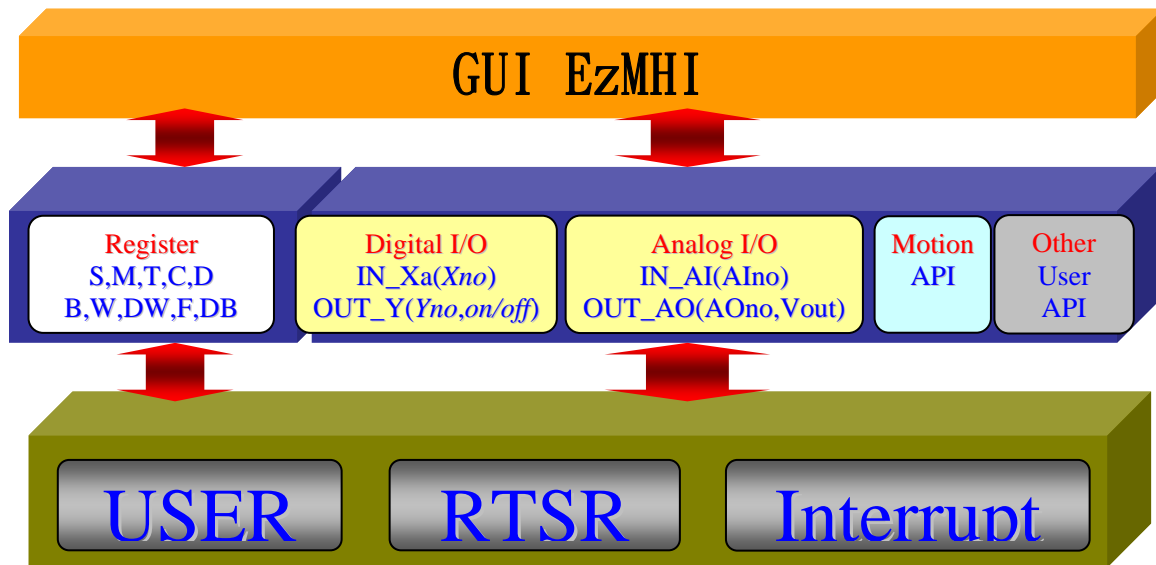


Figure 1: EzProg-I structure

System consists of three parts:

1. Top layer:
EzHMI provides a number of ActiveX controls which allows the programmer to design a user interface on a WinCE system for monitoring and controlling purposes. The ActiveX controls can be directly linked to IO registers for displaying or manipulating of IO data. In addition the ActiveX controls support Multilanguage.
2. Middle layer
EzProg-I has a section of memory called register where bit, integer, floating point and string values can be stored and accessed. These registers can be accessed by the upper and bottom layer. EzHMI controls can be linked directly to a register type and register number to access data.
3. Bottom layer
 - a. User thread: Programmer has to add code to the function. The code will be executed only once

- b. RTSR: Similar to plc coding. This API will be called at fixed time intervals.
- c. Hardware interrupt: The Interrupt has got the highest priority and immediately interrupts the execution of other tasks. After the interrupt handler has completed its execution code a function with a lower priority will be executed.

2.1.1 EzProg-I Register

EzProg-I has a section of memory called register where bit, integer, floating point and string values are stored and accessed. The register table is divided in two main parts: one part stores IO data from the slot modules (like the X,Y, AO, AI registers) and the other part holds non IO related values (e.g. M, D,F, MSG). The IO registers are updated every milliseconds, that means the EzProg-I engine scans every millisecond all D/A input modules in the slots to update the X, AI registers and writes D/A data from the Y, AO registers to the output modules. Use the EzConfig utility assist the system designer to map IO channels to register. EzHMI controls can directly access data from a register by linking the control to a register type and number. EzHMI uses data from the register to change the attributes of the on-screen controls.

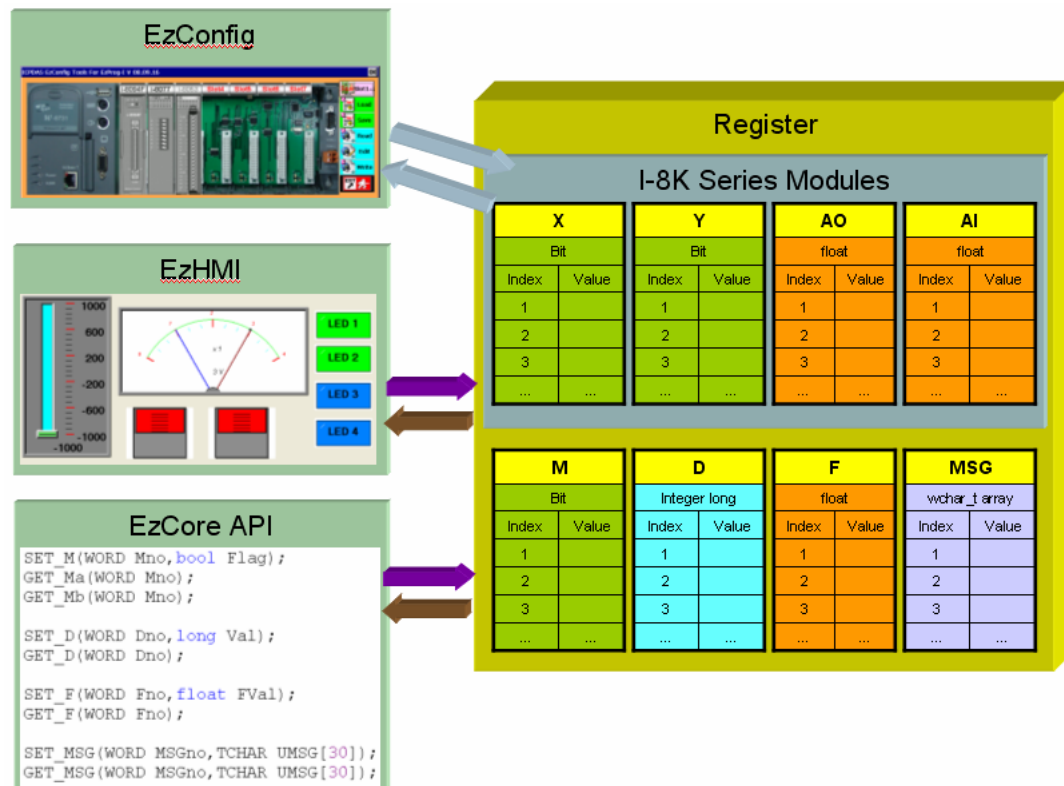


Figure 2: Registers in the EzProg-I framework

The registers supported by EzProg-I are listed in the following table:

	Specification	Register	Register number	Data type	Size	Range
Slot Device Register	Digital Input	X	Local DI: 0 ~ 777	bit	1 bit	true / false
			FRNet DI: 1000 ~ 7777			
	Digital Output	Y	Local DO: 0 ~ 777	bit	1 bit	true / false
			FRNet DO: 1000 ~ 7777			
Analog Output	AO	Local AO: 0 ~ 511	float	4 bytes	3.4E +/- 38	
Analog Input	AI	Local AI: 0 ~ 511	float	4 bytes	3.4E +/- 38	
Software Register	Timer	T	None Retain: 1 ~ 299	bit	1 bit	true / false
	Counter	C	None Retain: 1 ~ 511	bit	1 bit	true / false
			Retain: 512 ~ 1023			
	Flag	M	None Retain: 1 ~ 6999	bit	1 bit	true / false
			Retain: 8192 ~ 15999			
	Step	S	None Retain: 1 ~ 8191	bit	1 bit	true / false
	long integer	D	None Retain: 1 ~ 3599	long integer	4 bytes	-2,147,483,648 to 2,147,483,647
			Retain: 4096 ~ 7999			
	BYTE	B	None Retain: 1 ~ 699	unsigned char	1 byte	0 to 255
			Retain: 1024 ~ 2047			
	WORD	W	None Retain: 1 ~ 1023	unsigned short	2 bytes	0 to 65,535
			Retain: 1024 ~ 1999			
DWORD	DW	None Retain: 1 ~ 4095	unsigned long	4 bytes	0 to 4,294,967,295	
		Retain: 4096 ~ 8191				
Float	F	None Retain: 1 ~ 1899	float	4 bytes	3.4E +/- 38	
		Retain: 2048 ~ 3999				
Special Type	DB	None Retain: 1 ~ 49				
Message	MSG	Retain: 1 ~ 249	30 wchar_t	60 bytes	30 unicode char	

Table 1: Register types

The next table lists the APIs for accessing the registers:

Specification	Register	Read from Register	Write to Register
Digital Input	X	IN_Xa (X_RegisterNo);	-
Digital Output	Y	GET_Ya (Y_RegisterNo);	OUT_Y (Y_RegisterNo, Flag);
Analog Output	AO	GET_AO (AO_RegisterNo);	OUT_AO (AO_RegisterNo, Value);
Analog Input	AI	IN_AI (AI_RegisterNo);	-
Timer	T	GET_T (T_RegisterNo);	SET_T (T_RegisterNo, Flag, Value);
Counter	C	GET_C (C_RegisterNo);	SET_C (C_RegisterNo, Flag, Value);
Flag	M	GET_Ma (M_RegisterNo);	SET_M (M_RegisterNo, Flag);
Step	S	GET_S (S_RegisterNo);	SET_S (S_RegisterNo);
long integer	D	GET_D (D_RegisterNo);	SET_D (D_RegisterNo, Value);
BYTE	B	GET_B (B_RegisterNo);	SET_B (B_RegisterNo, Value);
WORD	W	GET_W (W_RegisterNo);	SET_W (W_RegisterNo, Value);
DWORD	DW	GET_DW (DW_RegisterNo);	SET_DW (DW_RegisterNo, Value);
Float	F	GET_F (F_RegisterNo);	SET_F (F_RegisterNo, Value);
Message	MSG	GET_MSG (MSG_RegisterNo, TCHAR UMSG[30]);	SET_MSG (MSG_RegisterNo, TCHAR UMSG[30]);

Table 2: APIs for accessing registers

3 EzTemplate

The EzTemplate is a template for Visual Studio 2008 to enable a user which is neither familiar with MFC nor Visual Studio 2008 to easily create a Window CE 6.0 application. The EzTemplate provides all the necessary tools to start writing a real time logic controller with a human machine interfaces. Direct DIO control via the user interface, for example for displaying or setting a digital channel status, can be done without any programming.

All the necessary libraries and header files are included in the project for the system designer to make full use of the power of the EzHMI controls. Initialization and IO scanning engine are activated by default. Easy to use programming interfaces are at user's disposal to unburden the programmer from understanding MFC and graphic programming. Several user threads and real time service routine (multitasking) are available to run the control program in a multitasking and deterministic environment. Task cycle time and task priority can be set with ease. The EzTemplate includes 50 dialog pages (design windows). Dialog pages for the following screen resolutions are provided:

- 640x480
- 800x600
- 1024x760

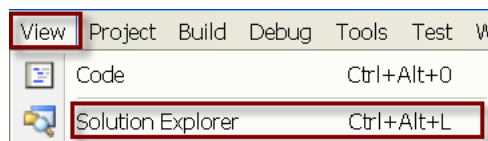
3.1 EzTemplate Setting

3.1.1 Pages

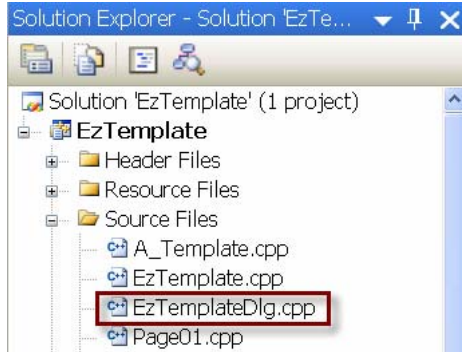
The EzTemplate provides up to 50 user windows (pages). The program developer has to decide how many pages he needs in order to provide the operator with all the necessary information and input capabilities to operate the control system.

The following steps describe procedure of setting of the number of windows:

STEP 1: Open the Solution Explorer: *View* → *Solution Explorer*.



STEP 2: Double click *Ez_TemplateDlg.cpp* in the Solution Explorer.



STEP 3: The value of the PageCount definition represents the number of independent dialog windows to be used for the program. The default number of dialog is ten.

```

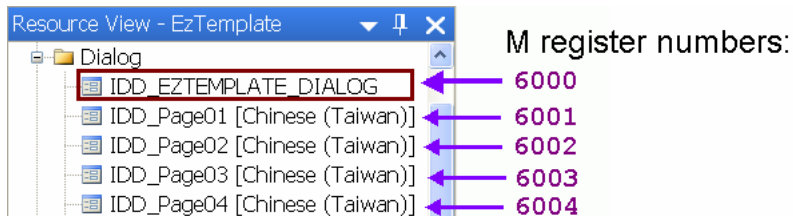
////////////////////////////////////
////////////////////////////////////
#define      PageCount      10    // 0~50 : maximum 51 pages

```

STEP 4:

Each dialog window is linked to an M register number. If the register of a corresponding window is set to true, the window will be visible on the screen. The main page “IDD_EZTEMPLATE_DIALOG” is linked by default to M register 6000. For each page following the main page the corresponding M register number is incremented by one.

```
#define      M_Page          6000
```



The user can change the default M register setting. By using

```
#define      M_Page          5000
```

The main page “IDD_EZTEMPLATE_DIALOG” is linked to M register 5000. The register numbers following the main window are automatically assigned to the remaining windows.

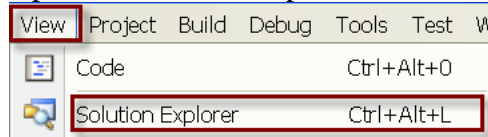
3.1.2 Programming interfaces

To make the programming easier all the necessary programming interfaces are provided in the *A_Template.cpp* file. These interfaces are a great help especially for plc and c programmers who are not familiar with MFC and graphic programming. They significantly reduce the programming time by relieving the user from the burden of studying Windows CE APIs. All the code for initialization and controlling has to be implemented in this file. Programming code can be added to the following functions:

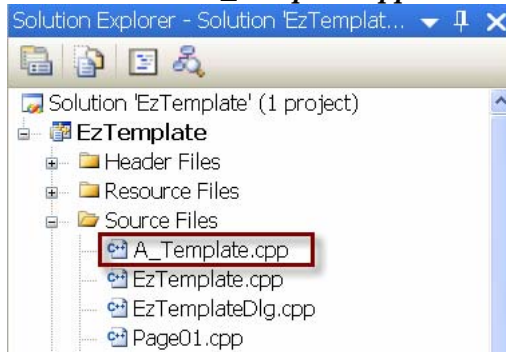
- USER_INITIAL
- USER_THREAD 0 to 7
- Real Time Service Routine (RTSR 0 to 7)

To open the *A_Template.cpp* file in VS2008 go ahead as follows:

STEP 1: Open the Solution Explorer: *View* → *Solution Explorer*



STEP 2: Double click *A_Template.cpp* in the Solution Explorer



3.1.2.1 Initialization

The main purpose of the USER_INITIAL() function is to initialize registers and variables at program start. The following example illustrates this.

```
void USER_INITIAL()  
{  
    SET_M(2,true); //Set M register number 2 to true  
    SET_D(333, 10000); //Set D register number 333 to 1000  
    SET_MSG(44, L"Hello"); //enter a string to MSG register  
                           //number 44  
}
```

3.1.2.2 User Thread

The user thread API allows the programmer to write low priority code like updating the user interface or other none time critical operations. The eight user threads each execute their code in a separate thread. The thread priority of the user threads are fixed and can not be changed in the program. The priority level decreases with increasing thread number. `USER_THREAD_0` has the highest and `USER_THREAD_7` the lowest priority. The user thread can be called anywhere in the program by calling the `START_USER_THREAD` API.

The following steps describe how to call the user thread at program start.

STEP 1: Open the *EzTemplateDlg.cpp* file

STEP 2: Go to the `CEzTemplateDlg::OnInitDialog()` in the *EzTemplateDlg.cpp* file and remove the forward slashes `//` from the user thread you want to call at program start. In the following `USER_THREAD_3` is activated:

```
//===== Start USER_THREAD =====  
//ret= START_USER_THREAD(0, USER_THREAD_0); //Run background  
//ret= START_USER_THREAD(1, USER_THREAD_1); //Run background  
//ret= START_USER_THREAD(2, USER_THREAD_2); //Run background  
ret= START_USER_THREAD(3, USER_THREAD_3); //Run background  
//ret= START_USER_THREAD(4, USER_THREAD_4); //Run background  
//ret= START_USER_THREAD(5, USER_THREAD_5); //Run background  
//ret= START_USER_THREAD(6, USER_THREAD_6); //Run background  
ret= START_USER_THREAD(7, Tree_Page); //Run background
```

STEP 3: Open the *A_Template.cpp* file and add your code to the `USER_THREAD_3` function

```
//===== Run UserThread3=====  
unsigned long USER_THREAD_3(void *)  
{  
    //Add your code here  
  
    //=====  
    END_USER_THREAD(3);  
    return _NO_ERROR;  
}
```

3.1.2.3 Real Time Service Routine (RTSR)

This RTSR function is very similar to the tasks of a multitasking PLC. Every RTSR function is being called at a set time interval. The priority settings of the eight RTSRs are fixed. The priority level is decreasing from RTSR 0 to 7. RTSR_0 has the highest priority followed by RTSR_1 and RTSR_7 has the lowest priority level.

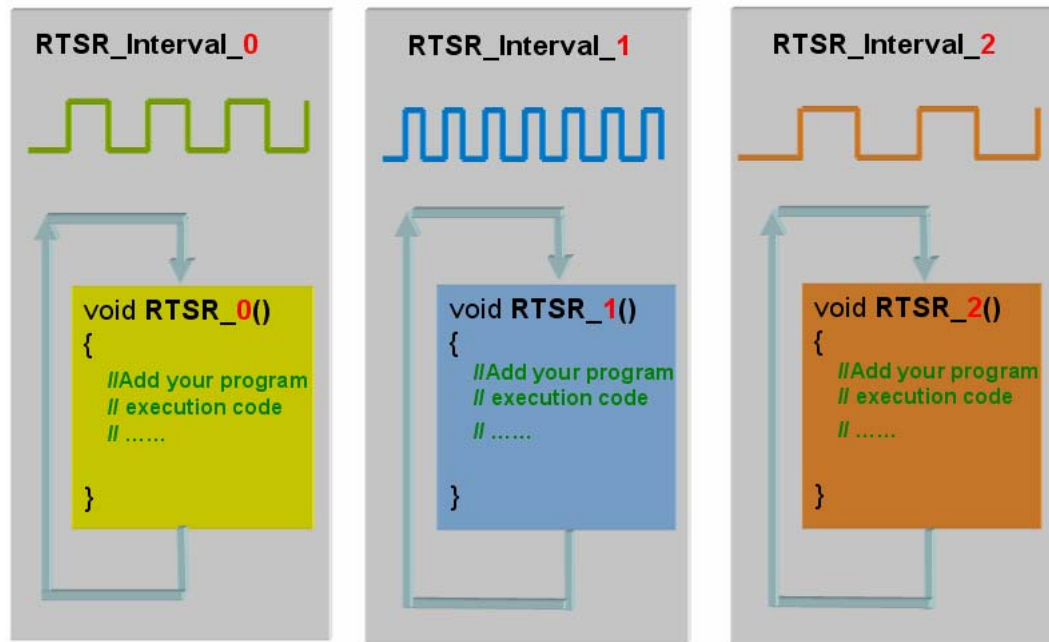


Figure 3: Real time service routines (RTSR) with different priorities

3.1.2.3.1 RTSR Activation

The following steps demonstrate the activation of a RTSR in the EzTemplate. In this example the activation of the RTSR_4 function will be shown.

STEP 1: Open the *EzTemplateDlg.cpp* file.

STEP 2: The RTSR_4 calling interval time is defined in the `#define` section at the beginning of the file. The default interval time for the `RTSR_Interval_4` constant is 100 milliseconds. The user can change this value to suit his requirements. The smallest interval value supported is 2 milliseconds. In this example we assign `RTSR_Interval_4` 150 milliseconds.

```
////////// For RTSR ////////////////////////////////////////////
#define RTSR_Interval_0 5 // 5 ms RTSR Interval
#define RTSR_Interval_1 10 // 10 ms RTSR Interval
#define RTSR_Interval_2 20 // 20 ms RTSR Interval
#define RTSR_Interval_3 50 // 50 ms RTSR Interval
#define RTSR_Interval_4 150 // 100 ms RTSR Interval
#define RTSR_Interval_5 200 // 200 ms RTSR Interval
```

```
#define RTSR_Interval_6 500 // 500 ms RTSR Interval
#define RTSR_Interval_7 1000 // 1000 ms RTSR Interval
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

STEP 3: Go to the `CEzTemplateDlg::OnInitDialog()` in the `EzTemplateDlg.cpp` file and remove the forward slashes `//` from the `//ret= START_RTSR(4);` comment line

```
//===== Start RTSR =====
//ret= START_RTSR(0); //if you want use RTSR 0 remove this "/" comment
//ret= START_RTSR(1); //if you want use RTSR 1 remove this "/" comment
//ret= START_RTSR(2); //if you want use RTSR 2 remove this "/" comment
//ret= START_RTSR(3); //if you want use RTSR 3 remove this "/" comment
ret= START_RTSR(4); //if you want use RTSR 4 remove this "/" comment
//ret= START_RTSR(5); //if you want use RTSR 5 remove this "/" comment
//ret= START_RTSR(6); //if you want use RTSR 6 remove this "/" comment
//ret= START_RTSR(7); //if you want use RTSR 7 remove this "/" comment
```

STEP 4: Add your control code to the `RTSR_4()` function

```
void RTSR_1()
{
    // Add your code here
}
```

For each RTSR the scan time can be set. To ensure deterministic behavior of the RTSR it is important to make sure that the code in the RTSR is executed within the set time scan interval. Therefore it is suggested not to call the Sleep API or use an endless loop inside a RTSR.

If the interval is set to three milliseconds and a Sleep API is being called in the RTSR to suspend the execution of the RTSR thread for more than three seconds the behavior of the RTSR becomes nondeterministic. The same applies to an endless while loop, which causes the execution of one RTSR cycle to run indefinitely. Also care should be taken when loops are used which requires a lot of CPU resources. The execution time of a RTSR can be Within the The interval time can be guaranteed execution time of a RTSR can be reduced by avoiding Sleep, large “for” loops and by implementing small execution code or by setting the time interval to a larger value.

```
void RTSR_1()
{
    for (int i=1, i<= 100; i++)
    {
        Sleep(10); ← AVOID using Sleep inside a RTSR

        // Execute some code
        //.....
    }
}
```

```
void RTSR_1()
{
    while(true) ← Do NOT use a infinite loop inside a RTSR
    {
```

```

        // Execute some code
        //.....
    }
}

```

An infinite loop is a sequence of instructions in a computer program which loops endlessly, either due to the loop having no terminating condition, having one that can never be met, or one that causes the loop to start over. An infinite loop in a RTSR with a high priority causes the entire system to become unresponsive as the loop consumes all available processor time. The only way to end the loop is to power off the console.

EzCore provides the function *GET_RTISR_TIME(BYTE RTISRno)* to check whether the execution time of the previous RTSR call exceeded the interval time. The following example shows how to implement this function: compare the execution time with the interval time, if the execution time exceeds the interval time generate an error message is generated and abort execution of the remaining RTSR code.

```

void RTISR_4()
{
    //Read the execution time of the previous RTISR_4 call
    if(GET_RTISR_TIME(4)> RTISR_Interval_4)
    {
        SET_MSG(1, L"RTISR_4 interval time exceeded!");
        return;
    }

    // Add your code here
}

```

3.2 Priority Levels

The EzCore has got the following priority levels:

- Interrupt
- System task
- RTSR task
- User thread
- HMI

All priorities are fixed and can not be change by the programmer.

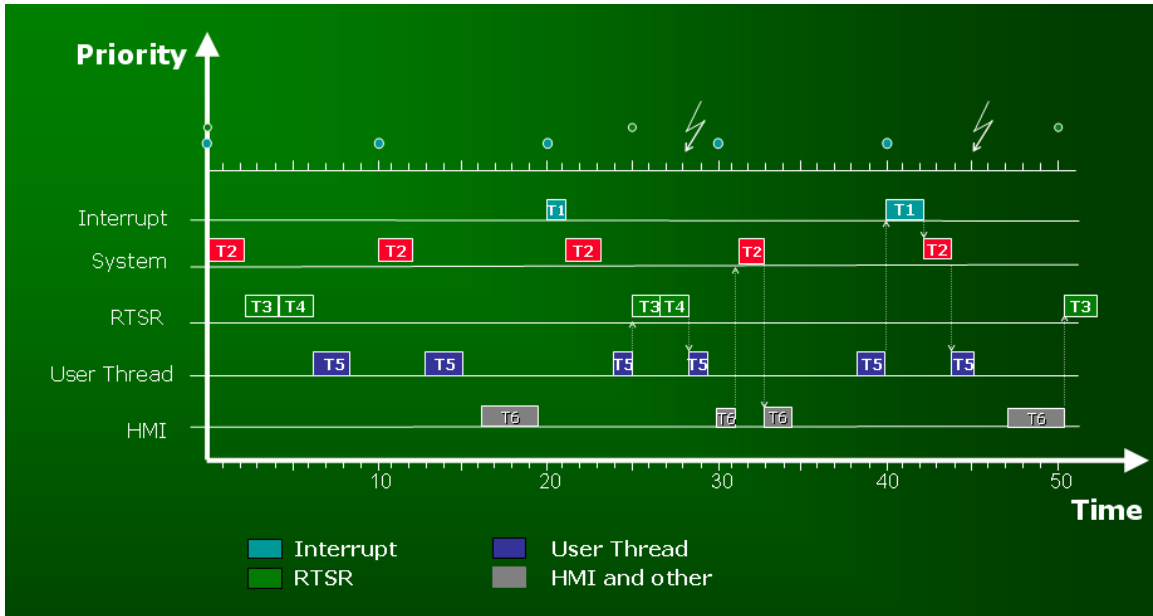


Figure 4: Priority levels of EzCore

Interrupt by hardware

A hardware interrupt causes the processor to save its state of execution and begin execution of an interrupt handler. Hardware interrupts is a way to avoid wasting the processor time in polling loops, waiting for external events. The Interrupt has got the highest priority and immediately interrupts the execution of other tasks. After the interrupt handler has completed its execution code a function with a lower priority will be executed. Two APIs are provided for the interrupt:

- **SET_INT()**: assigning the interrupt a interrupt handler
- **START_INT()**: setting the interrupt mode: falling or rising edge

The hardware interrupt is supported by the following modules

- DI module: 8094H in slot 1
- Motion module: 8094A, 8094F, 8092F (in slot 1 to 3)

Example:

```
#define INTP_MODE_Rising 1
#define INTP_MODE_Falling 2

RET = SET_INT(0, &(ptTSRFunc)INTP_RUN0);
RET = START_INT(0, INTP_MODE_Rising);

// Interrupt handler:
void INTP_RUN0()
{
    SET_D(1, GET_D(1)+1);
    OUT_Y(0, GET_Yb(0));
}
```


System task

The system task scans every millisecond the I/O modules in slot 1 to 7 and update the corresponding register.

RTSR task

The RTSR task is being called at fixed time intervals. The time interval can be set for every RTSR task. Eight RTSR tasks with predefined priority levels are provided. The priority level decreases from 0 to 7. The minimum interval time is 2 milliseconds.

- RTSR_0 () → **highest priority**
- RTSR_1 ()
- RTSR_2 ()
- RTSR_3 ()
- RTSR_4 ()
- RTSR_5 ()
- RTSR_6 ()
- RTSR_7 () → **lowest priority**

User thread task

The task priority is decreasing with increasing user thread number.

- USER_THREAD_0 () → **highest priority**
- USER_THREAD_1 ()
- USER_THREAD_2 ()
- USER_THREAD_3 ()
- USER_THREAD_4 ()
- USER_THREAD_5 ()
- USER_THREAD_6 ()
- USER_THREAD_7 () → **lowest priority**

HMI

The updating the user interface has the lowest priority.

4 EzHMI

EzHMI is a package of ActiveX controls for the Windows Embedded CE 6.0 environment containing different kind of buttons, switches, knobs, sliders, gauges which can be used in many industry applications and simulation environments. It is designed to visualize and integrate the real-time process and production data. EzHMI releases the developer from designing graphic components and thereby speeds up the project developments involving SCADA systems, HMI and simulations. It allows the developer to fully focus on programming the logic control. The user interface can be fully customer configurable which supports customer-provided picture. Developers can design a graphic interface on a WinCE system to their dialog boxes, using a variety of shapes, surfaces, textures, bitmaps and icons, colors and fonts.

4.1 EzHMI ActiveX overview

Symbol	Name	Picture
--------	------	---------



LED



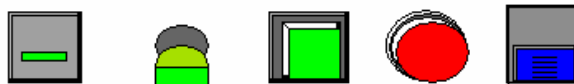
The LED is used to visualize boolean discrete status:

- ON/OFF
- True/False
- Active/Inactive
- Open/Closed

A text label can be added to the control.



SWITCH



The switch is used to input and visualize two statuses at runtime:

- ON/OFF
- True/False
- Active/Inactive
- Pressed/ not pressed.

The switch can be labeled with a text.



Label



You can enter one or several lines of text in a LABEL box and define the font width and color. You can add a background color or pattern to a text box.

A LABEL control can be set to display at runtime

- Name or caption
- Value
- Text message



ColorEdit



A ColorEdit box have the following runtime functions:

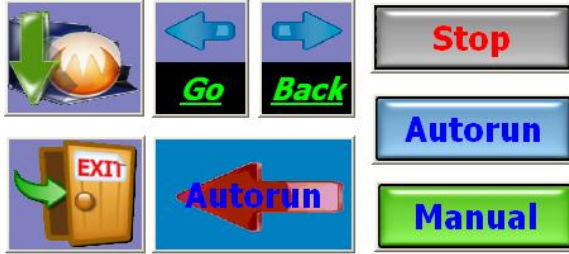
- Input/output of values
- Input/output of text

You can define the limits for the input values.

User input can be disabled at runtime.



ButtonST



The ButtonSt triggers an event, notification or acknowledgement when it is being clicked or released. The implementation of the button event has to be done in c or c++ programming language. The operator can use a button to control a process. Images can be added to the button without any programming effort.



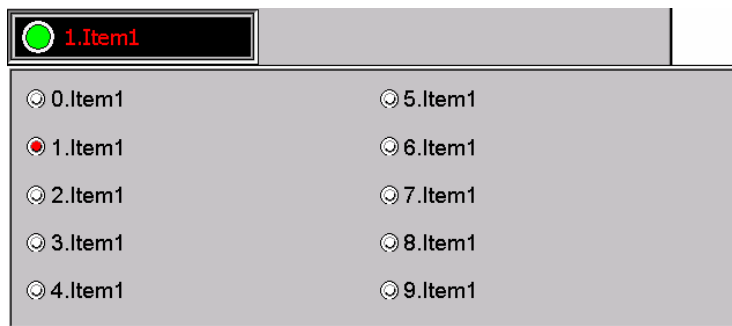
Image



The Image control displays graphic objects on the screen which has been created by a graphic software. Only graphic images saved as ".bmp" can be shown. In addition more than one image can be attached to a control. During runtime the image itself can be replaced by another image and the position of the image can be changed. Functions are provided to move the image across the screen.



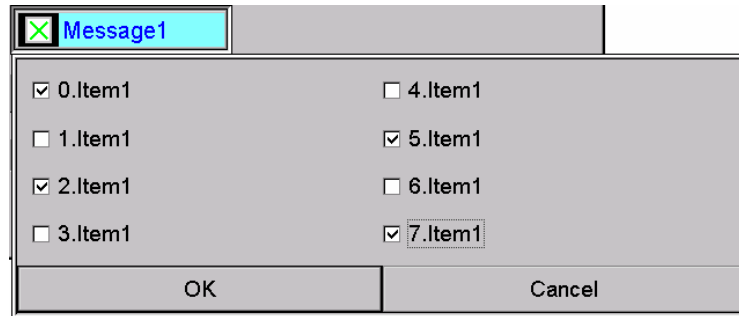
ColorRadio



A ColorRadio control allows the operator to choose only one of a predefined set of options. When the operator selects an option, any previously selected option in the same group becomes deselected.



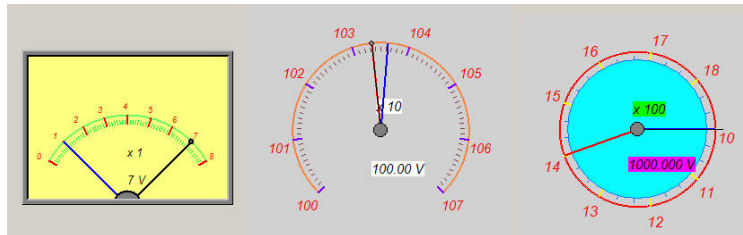
ColorCheck



The ColorCheck allows the selection of several items.



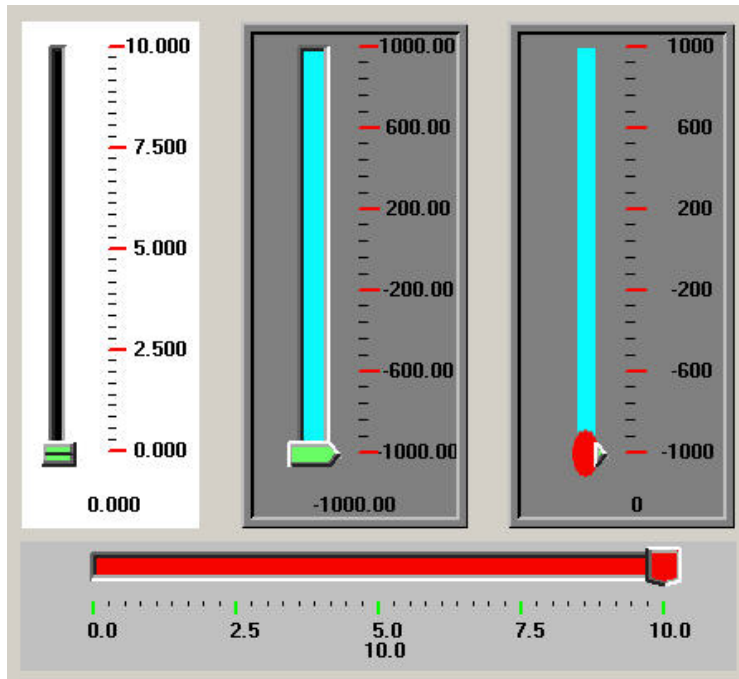
EzKnob



The EzKnob visualizes data like pressure, temperature, volt, etc. in form of a multi-needle gauge. The EzKnob object also allows the user to manually input values by dragging the needle (via mouse or touch screen) to the desired position on the scale. This ActiveX can be directly linked to analog I/O channel.



EzSlider



The EzSlider represents a process value in the form of a scaled bar. The slider bar allows you to visualize or enter dynamic values (temperature, filling levels, pressure, etc.). New values are entered by sliding the indicator to required value.



EzList



The EzList outputs messages during runtime. Each message can be provided with a date and time stamp. New messages are automatically added to top of the list.



Position



The Position control is used for motion control applications. It can be set to display one of the following motion parameters:

- Logic Position
- Encoder Position
- Velocity
- Acceleration

Table 3: EzHMI controls

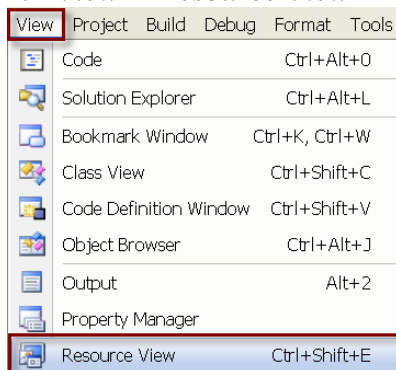
4.2 Settings

4.2.1 Visual Studio 2008 IDE

4.2.1.1 Adding EzHMI ActiveX controls

In order to use the EzHMI controls for the EzTemplate project the EzHMI ActiveX have to be added to the VS2008 toolbox. The following steps describe the procedure:

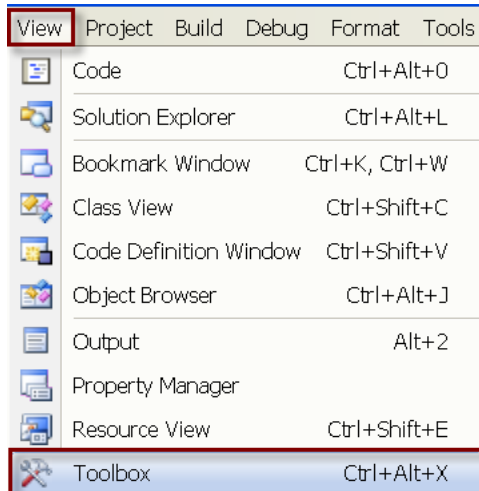
STEP 1: Click *View* → *Resource View*



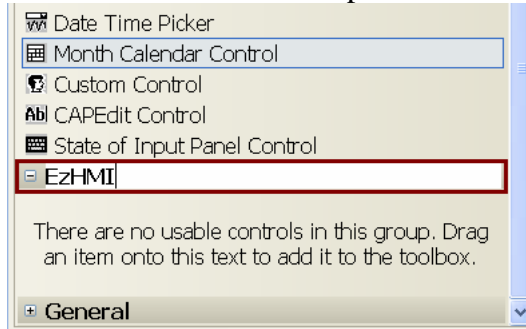
STEP 2: Double click the “IDD_EZTEMPLATE_DIALOG” resource. The dialog sheet will appear.



STEP 3: Click *View* → *Toolbox* to display the toolbox

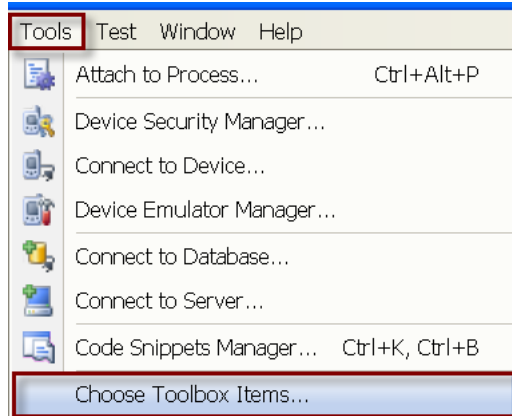


STEP 4: Right click the toolbox and select “Add Tab” from the popup window. Enter “EzHMI” as a tab name and press enter.



STEP 5: Click on the EzHMI tab

STEP 6: Click *Tools* → *Choose Toolbox Items ...*



It will take some time before the “Choose Toolbox Items” dialog window pops up.

STEP 7: Click the “*COM Components*” tab.

STEP 8: Select from the “*COM Components*” list the 12 EzHMI ActiveX controls and click “*OK*”.

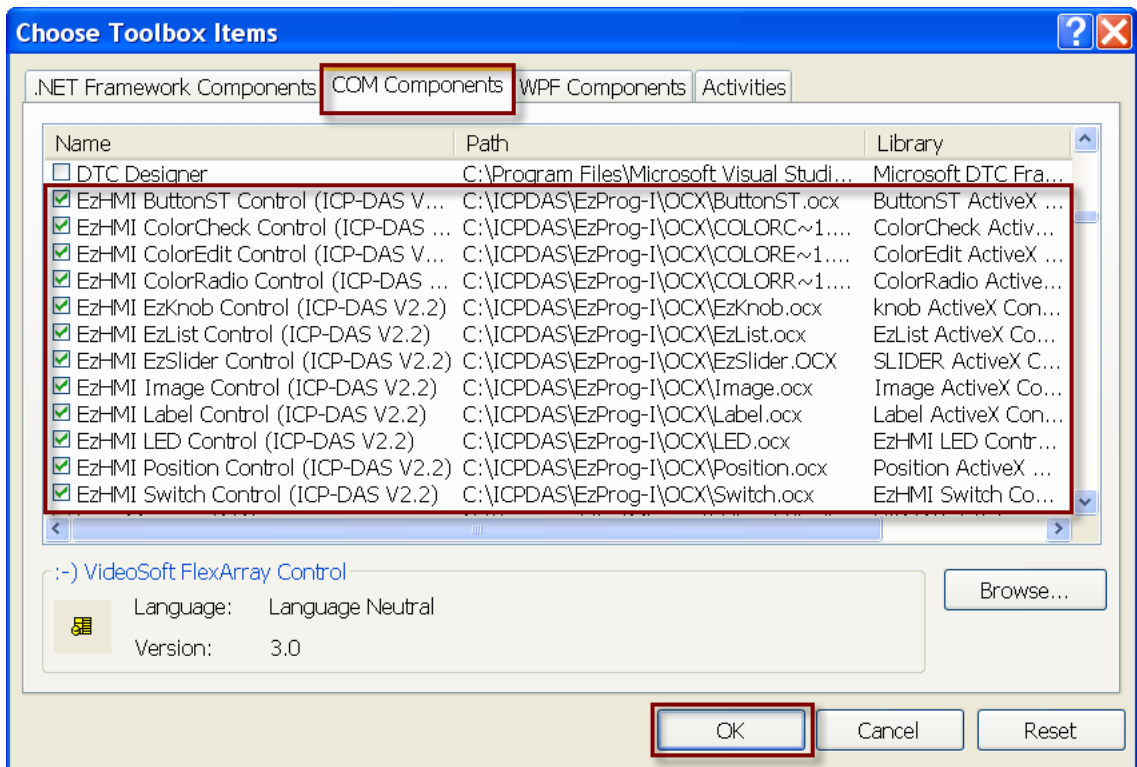


Figure 5: Load EzHMI controls to the toolbox of VS2008

All the selected controls will be displayed beneath the EzHMI tab.

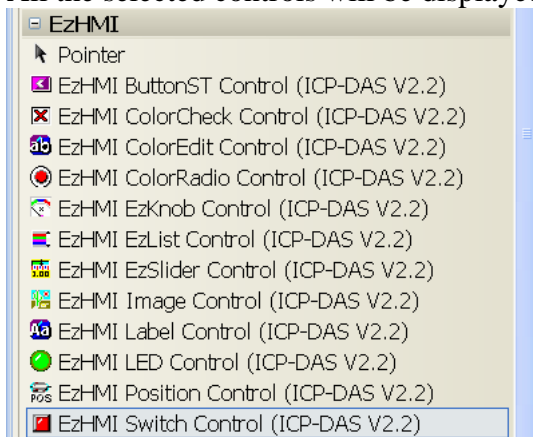


Figure 6: EzHMI controls in the toolbox

4.2.1.2 Adding ActiveX controls to dialog resource

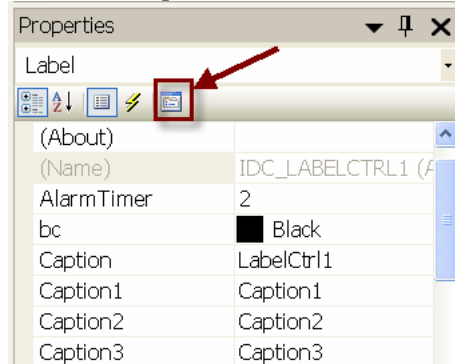
To add a control to the design surface and set their properties follow the next steps:

STEP 1: Select from the toolbox the required control by clicking once on it.

STEP 2: Indicate with a click on the design surface the upper left position of the control.

STEP 3: Right click the control and select from the pop-up menu “*Properties*”.

STEP 4: On the “*Properties*” window click the “*Properties Pages*” icon.



STEP 5: Now you can do the necessary configuration on the “*Properties Pages*”.

Note: Do NOT set the EzHMI control properties directly in the “*Properties*” window. Always do the setting on the “*Properties Pages*”.

4.2.2 Refresh Time

In order to increase the performance of the system it is important to define refresh time carefully. Here are some guidelines for configuring the refresh time:

- Select the refresh time in such a way that it reflects the rate at which the process variables change. Each EzHMI control can be set to a different refresh time. For example, displaying the ambient temperature the scan rate can be set to more than 1 minute as the temperature changes slowly and is not likely to change suddenly. On the other hand, a digital point that is monitoring the on/off state of a valve would need a shorter scan rate to accurately reflect the current valve status.
- As a good engineering practice, select the slowest possible scan rate that is acceptable for your application. This will help prevent the system from being overloaded by needlessly scanning processes.
- If the user interface has a large amount of EzHMI controls select a lower scanning speed otherwise the graphic interface will not be updated within the refresh time.
- Always remember that the priority for updating the EzHMI controls is lower than the priority for the controlling process (RTSR or ISR). Therefore if your

controlling process is very CPU intensive less CPU resource will be available for handling the graphic refreshing. In this case it is more reasonable to use a low scan rate.

4.2.3 Language Switching

EzHMI allows you to configure a multilingual project. Up to eight languages can be loaded simultaneously onto the HMI device. You can switch between the individual languages at runtime. Language switching can be used on almost any EzHMI control. By default language switching is not activated. To enable language switching on a control, first add a EzHMI control to the design surface and in the property panel check the “Display Status Text” checkbox. Make sure that the update rate (“Flash Timer 0,1,2 ...”) is set to a value greater than zero.

The following EzHMI controls support Multilanguage:

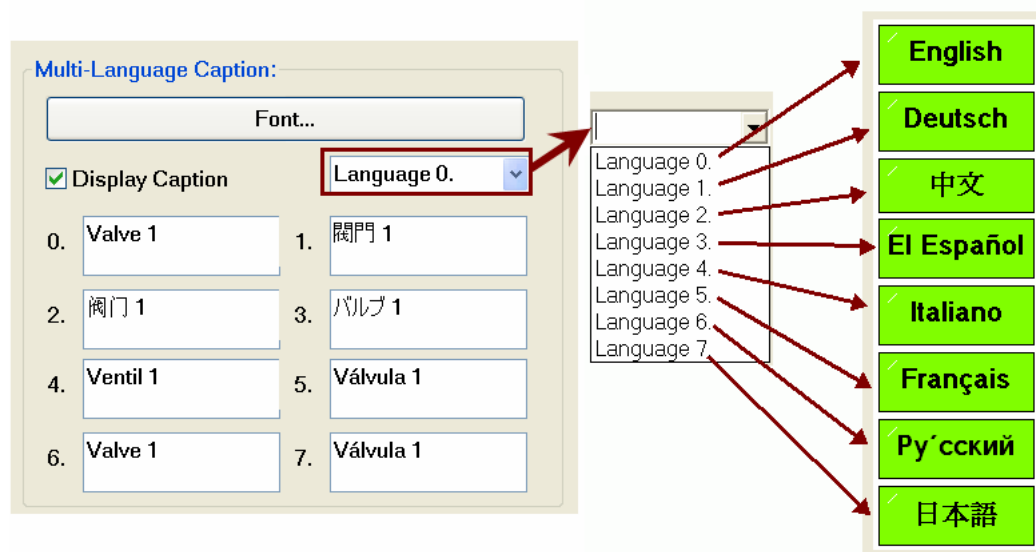
- LED
- SWITCH
- Label
- ButtonST
- ColorCheck

At design time you have to determine how many languages your program needs to support and assign each language an index from 0 to 7. It is suggested to create a table which lists the different languages with their assigned indexes. Every EzHMI controls which supports Multilanguage input provides edit boxes with labels from 0 to 7. Make always sure that the text is edited in the correct edit box by comparing the language index and text field index. Both indexes have to be identical. Throughout the project it is required to stick to one language numbering otherwise there will be a mismatch of languages.

Language Index	Language	Edit Box Index
0	English	0.
1	Traditional Chinese	1.
2	Simplified Chinese	2.
3	Japanese	3.
4	German	4.
5	Spanish	5.

6	French	6.
7	Portuguese	7.

Table 4: Language mapping



Enter the text to be displayed on the control in different languages to the textboxes (0 to 7) of the “*Properties Pages*”. Each language textbox (0 to 7) represent a different language. The default text is set by selecting a language from drop-down list. In the figure above language “Language 0.” is selected as the default language.

During runtime one of the eight languages (0 to 7) can be selected by

1. using EzHMI color radio control (see tutorial)
2. calling the `SET_D()` API in c code (see tutorial).

```
SET_D(WORD Dno, long Val);
```

with

Dno = 8000

Val = textbox number 0 to 7

4.2.4 Register linking

EzHMI parameters and some properties have to be linked to registers during design time. Register linking can be one way or bi-directional, that is to say that some parameters and properties can only read data and others can both read and write data to the assigned registry. The table below gives an overview of the register types the different EzHMI controls can be linked to.

Property	Registertype	LED	SWITCH	Label	ColorEdit	ButtonST	Image
Status Register (on/off)	Digital Input: X	Read	-	-	-	-	-
	Digital Output: Y	Read	Read/Write	-	-	-	-
	Timer: T	Read	-	-	-	-	-
	Counter: C	Read	-	-	-	-	-
	Flag: M	Read	Read/Write	-	-	Read/Write	Read/Write
	Step: S	Read	-	-	-	-	-
Value Register	Analog Input AI	-	-	Read	-	-	-
	Analog Output AO	-	-	Read	Read/Write	-	-
	Long Integer: D	-	-	Read	Read/Write	-	Read
	Float F	-	-	Read	Read/Write	-	-
	BYTE B	-	-	-	-	-	-
String Register	Message MSG	-	-	Read	Read/Write	-	-
Language Setting	D8000 D	Read	Read	Read	-	Read	-
Disable ActiveX	Flag M	-	Read	-	Read	Read	-
Hide Control	Flag M	-	-	-	-	-	Read

Property	Registertype	ColorRadio	ColorCheck	EzKnob	EzSlider	EzList	Position
Status Register (on/off)	Digital Input: X	-	-	-	-	-	-
	Digital Output: Y	-	-	-	-	-	-
	Timer: T	-	-	-	-	-	-
	Counter: C	-	-	-	-	-	-
	Flag: M	Write	Write	-	-	Read/Write	-
	Step: S	-	-	-	-	-	-
Value Register	Analog Input AI	-	-	Read	Read	-	-
	Analog Output AO	-	-	Read/Write	Read/Write	-	-
	Long Integer: D	Read/Write	-	Read/Write	Read/Write	-	-
	Float F	-	-	Read/Write	Read/Write	-	-
	BYTE B	-	Read/Write	-	-	-	-
String Register	Message MSG	-	-	-	-	Read	-
Language Setting	D8000 D	-	Read	-	-	-	-
Disable ActiveX	Flag M	Read	Read	Read	Read	-	-
Hide Control	Flag M	-	-	-	-	-	Read

Figure 7: Registers accessed by EzHMI objects

4.3 EzHMI ActiveX Controls

At present ICPDAS provides the following EzHMI controls:

4.3.1 LED Control

4.3.1.1 Description

The LED control can be used for indicating the following statuses

- on/off status,
- digital I/O status,
- over / under limit status,
- alarm, emergency,
- event,
- flag,

Setting:

In the following the configuration interface of the LED control will be discussed in more details:

The configuration interface for the LED control is divided into several sections:

- Appearance Styles:** Includes 'LED Type' (set to RECTANGLE) and 'Border Style' (set to NONE).
- Color:** Includes color selection for 'Status Text On Color' (black), 'Status Text Off Color' (blue), 'ON Color' (red), 'OFF Color' (green), and 'BackGround Color' (white).
- Multi-Language Caption:** Includes a 'Font...' button, a 'Display Caption' checkbox, and a 'Language 0.' dropdown. Below are eight caption input fields labeled 0 through 7.
- Register Type and Number Assignment:** Includes a 'Select X/Y/M/S/T/C' dropdown (set to RealDI X), an input field for 'X/Y/M/S/T/Cno -->LED(On/Off)' (set to 00000000), and a 'Refresh Interval (Unit 50ms):' section with a 'Flash Timer 0,1,2.....' input field (set to 00000002).

4.3.1.2 LED Appearances

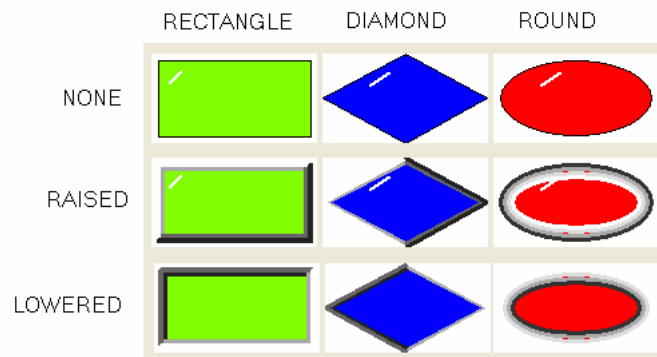
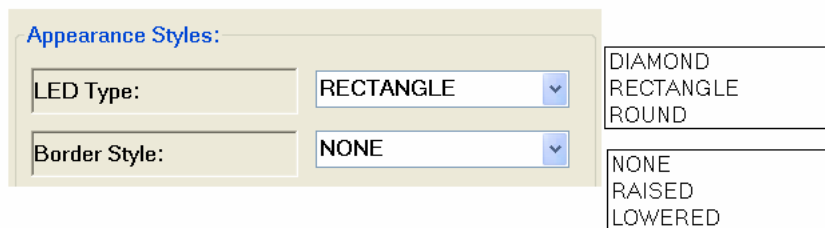
The LED control supports three types of shapes

- Rectangle
- Diamond
- Round

And the following border styles:

- None
- Raised
- Lowered

The color of the LED for displaying the ON and OFF status can be selected by clicking the color box next to the “ON Color” and “OFF Color”. In addition the background color of the LED is selectable.

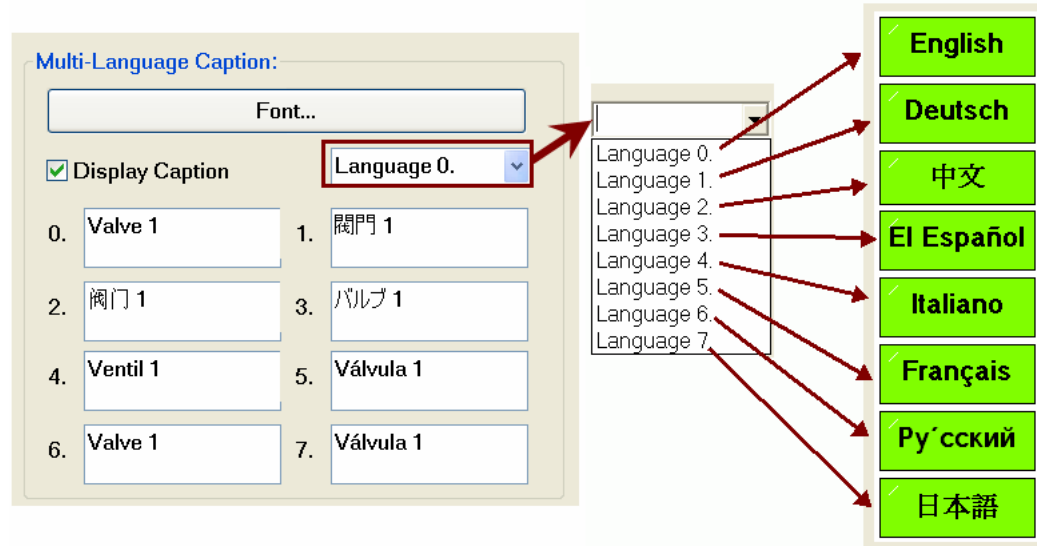


4.3.1.3 Caption

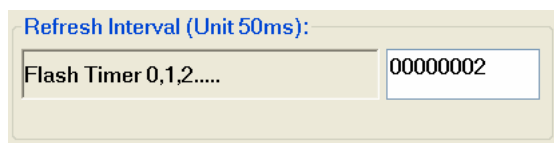
Each LED can be labeled with a caption text in 8 different languages. During runtime the caption text can be changed to a different language. To display a caption on the LED

- Check the “*Display Caption*” check box

- Type in one or more text boxes (0 to 7) the caption text in different languages
- Set the caption text font and size by clicking the “*Font...*” button
- Set the text color for the ON and OFF status by clicking the color button next to “*Status Text On Color*” and “*Status Text Off Color*” and selecting the desired color.
- Selected from the language combo box the default language.



4.3.1.4 Flash Timer



The task of the flash timer is to update the status of the LED display at fixed time intervals. The unit of the flash timer is 50 milliseconds. A flash timer value of two will update the LED status every 100 milliseconds.

If the flash timer setting is zero, the LED control will not be updated during the program execution time and the system language setting can not be changed during runtime. The control will not read any register it has been assigned to and therefore will not react to any register status change. To ensure that the LED control is updated in regular intervals an integer number greater than 0 has to be entered. The default value is 100 milliseconds.

4.3.1.5 Register assignment

A LED control can only read a status register (X/Y/M/S/T/C). The following steps describe the linking of a LED control to a register:

- STEP 1: The first step is to select a proper register type for the LED control. Select from the combo box “Select X/Y/M/S/T/C” the required register type.
- STEP 2: Then enter in the “X/Y/M/S/T/Cno →LED(on/off)” a register number according to the register table.

Specification	Register	Register numbers
Digital Input	X	Local DI: 0 ~ 777
		FRNet DI: 1000 ~ 7777
Digital Output	Y	Local DO: 0 ~ 777
		FRNet DO: 1000 ~ 7777
Analog Output	AO	Local AO: 0 ~ 511
Analog Input	AI	Local AI: 0 ~ 511
Timer	T	None Retain: 1 ~ 299
Counter	C	None Retain: 1 ~ 511
		Retain: 512 ~ 1023
Flag	M	None Retain: 1 ~ 6999
		Retain: 8192 ~ 15999
Step	S	None Retain: 1 ~ 8191

X and Y register stores data of digital IO modules plugged in slot 1 to 7. The mapping of the register number with the digital IO modules is done by the EzConfig utility. To choose a correct X or Y register number please consult the IO_Table generated by the EzConfig utility. No additional coding is required when a LED control is directly linked to a IO module register (X, Y).

Choosing register types T, C, M and S for the LED control requires the user to write code in the UserThread, RTSR or ISR to set the register to true or false. This is being done by calling the following APIs:

- SET_T (WORD Tno, bool Flag, DWORD Tval)
- SET_C (WORD Cno, bool Flag, DWORD COUNT);
- SET_M (WORD Mno, bool Flag)
- SET_S (WORD Sno);

If the LED control is not mapped to a valid register number it can not update its status as it has no valid source.

Notice that the XY register start from zero.

Make sure that a register number greater than zero is assigned for Register M, Register S, Register T or Register C otherwise the LED control is not mapped correctly and can not update its status.

4.3.2 Switch

4.3.2.1 Description

The EzHMI SWITCH control displays two statuses: “ON” (true) or “OFF” (false).

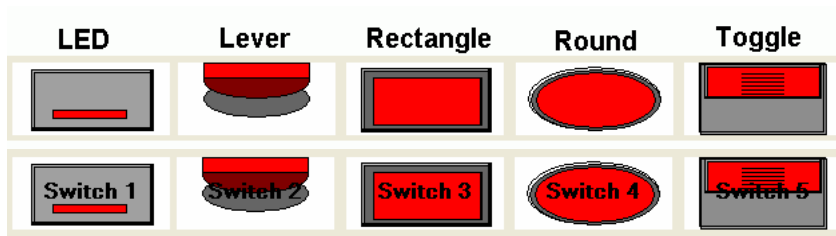
A SWITCH control reads, sets and displays the current status of a Y, M register. The mapped register can be set either to true or false. Clicking the EzHMI SWITCH causes the switch to write the opposite status to the register. If for example the current status of a register is true and the switch is activated, it will change the register to false.

A switch mapped to a digital output Y register can read and change the output state of a device (e.g. a motor, valve, etc.) connected to the mapped output channel. Clicking the SWITCH on the HMI automatically changes the state of the connected device.

Layout

The following types of switches are supported:

- LED
- Lever
- Rectangle
- Round
- Toggle



Setting:

In the following the configuration interface of the SWITCH control will be discussed in more details:

Appearance Styles:

Switch Type: SWITCH LED

Color:

Status Text On Color	Black
Status Text Off Color	Blue
ON Color	Red
OFF Color	Green
BackGround Color	White

Multi-Language Caption:

Font...

Display Caption Language 0.

0. Caption0	1. Caption1
2. Caption2	3. Caption3
4. Caption4	5. Caption5
6. Caption6	7. Caption7

Register Type and Number Assignment:

Y/M Enable

Select Y/M RealDO Y

Switch(On/Off) --> Y/Mno 00000000

Mno(On) --> Lamp 00000000

Control Status:

Mno(On) --> Disable ActiveX 00000000

Refresh Interval (Unit 50ms):

Flash Timer 0,1,2..... 00000002

4.3.2.2 SWITCH Appearances

The SWITCH control supports the following shapes:

- LED
- Lever
- Rectangle
- Round
- Toggle

Appearance Styles:

Switch Type: SWITCH LED

- SWITCH LED
- SWITCH LEVER
- SWITCH RECTANGLE
- SWITCH ROUND
- SWITCH TOGGLE



The color for displaying the ON and OFF status can be selected by clicking the color box next to the “ON Color” and “OFF Color”. In addition the background color of the SWITCH is selectable.

4.3.2.3 Caption

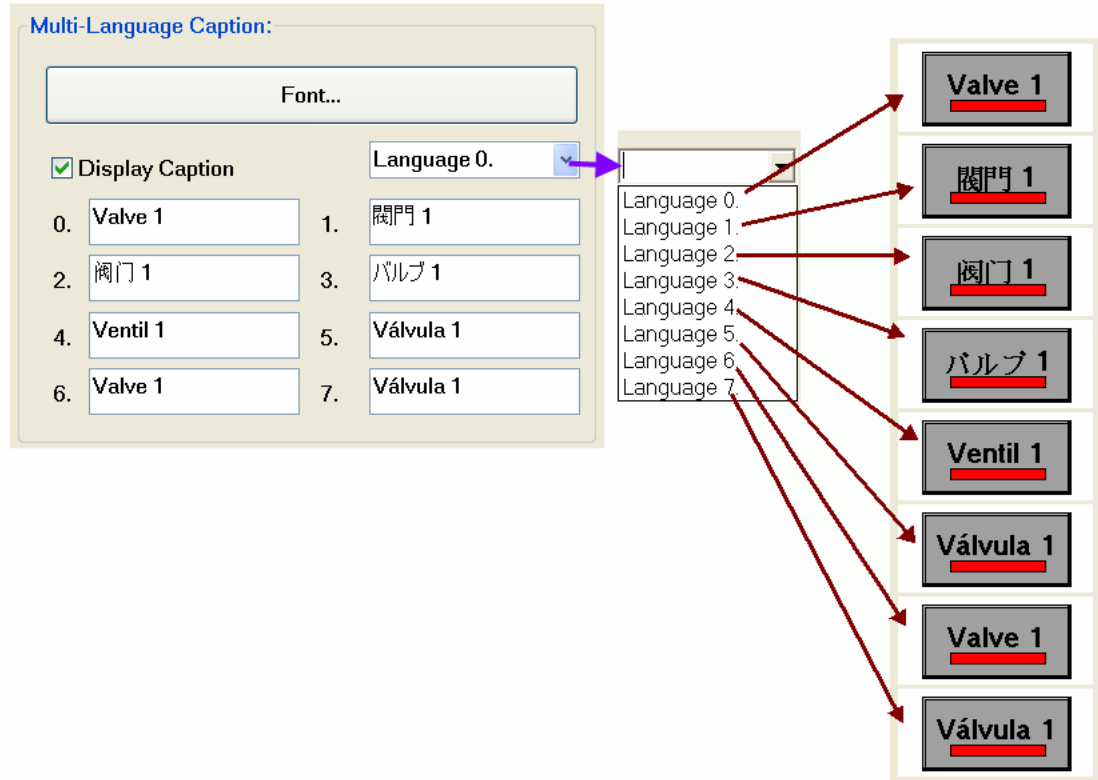
Each Switch can be assigned with a caption enabling the user to clearly identify the LED on the HMI. During runtime the caption text can be changed to a different language.

Different Switch styles with caption:

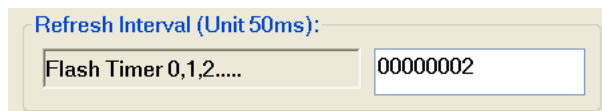


Approach to display a caption on a SWITCH object:

- Check the “Display Caption” check box.
- Type in one or more text boxes (0. to 7.) the caption text in different languages.
- Set the font and size of the caption text by clicking the “Font...” button
- Set the text color for the ON and OFF status by clicking the color button next to “Status Text On Color” and “Status Text Off Color” and selecting the desired color.
- Selected from the language combo box the default language.



4.3.2.4 Flash Timer



The SWITCH control flash timer has basically two tasks:

1. Switch state has been changed: Update the corresponding register to the new switch status.
2. Register state has been changed: Redraw the SWITCH to display the new register state.

The flash timer reads from and writes to the status register at the set time interval. The unit of the flash timer is 50 milliseconds. A flash timer value of two will update the SWITCH display and register status every 100 milliseconds.

If the flash timer setting is zero:

- the SWITCH control will not be updated during the program execution time,
- the language setting can not be changed during runtime,
- the control will neither read from nor write to the register it has been assigned to.

To ensure that the SWITCH control and the register are updated in regular intervals an integer number greater than 0 has to be entered.

4.3.2.5 Register assignment

A SWITCH control can read from and write to a Y/M status register. The following steps describe the linking of a SWITCH control to a status register:

- STEP 1: Select a proper register type for the SWITCH control. Select from the combo box “Select Y/M” the required register type.
- STEP 2: Then enter in the “Y/Mno→LED(on/off)” a register number according to the register table.

Register Type and Number Assignment:

Y/M Enable

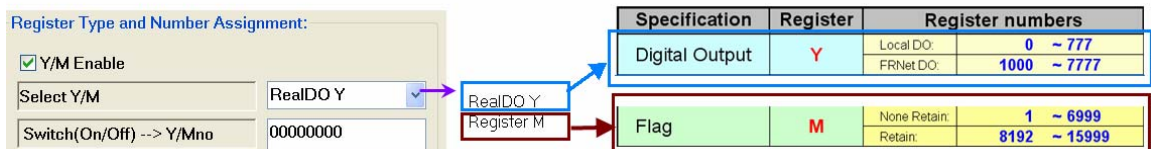
Select Y/M: RealDO Y

Switch(On/Off) --> Y/Mno: 00000000

Mno(On) --> Lamp: 00000000

Control Status:

Mno(On) --> Disable ActiveX: 00000000



X and Y register stores data of the physical DIO modules in slot 1 to 7. The mapping of the register number with the DIO modules is done by the EzConfig utility. To choose a correct X or Y register number please consult the IO_Table generated by the EzConfig utility. No additional coding is required when a LED control is directly assigned to a DIO module register (X, Y).

Choosing register types T, C, M and S for the LED control requires the user to write code in the UserThread, RTSR or ISR to set the register to true or false. This is being done by calling the following APIs:

- SET_T (WORD Tno, bool Flag, DWORD Tval)
- SET_C (WORD Cno, bool Flag, DWORD COUNT);

- SET_M (WORD Mno, bool Flag);
- SET_S (WORD Sno);

If the LED control is not mapped to a valid register number it can not update its status as it has no valid source.

4.3.3 EzHMI Lable

4.3.3.1 Description

A LABLE control can be set to display at runtime

- the current value of an assigned register (AI, AO, D, F)
- or a text message of an assigned register (MSG)
- or a fixed name or caption.

The LABLE control can have one of the following two states:

- Flashing:
The LABLE object starts to flash at a set time interval when a set condition has been met.
- Static:
A new value or text message will be displayed in the set color.

Setting:

In the following the configuration interface of the LABLE control will be discussed in more details:

Appearance Styles:

Frame: Client edge
 Static edge
 Modal frame

Align Text: DT_CENTER

Multi-Language Caption:

Language 0. 1.
2. 3.
4. 5.
6. 7.

MSG/AI/AO/D/F Enable

Color:

Back Color:

Font Color:

Register Type and Number Assignment:

Select MSG/AI/AO/D/F Register D
MSG/AI/AO/D/Fno --> Label
Decimal Point: 0
Upper Limit: 2000000000
Lower Limit: -2000000000

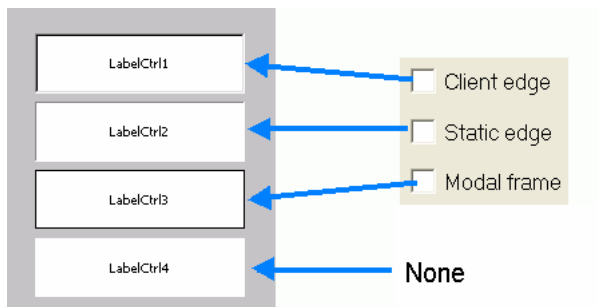
Refresh Interval (Unit 50ms):

Alarm Timer 0,1,2,.....
Flash Timer 0,1,2,.....

4.3.3.2 LABEL Appearances

The LABEL control supports the following border styles:

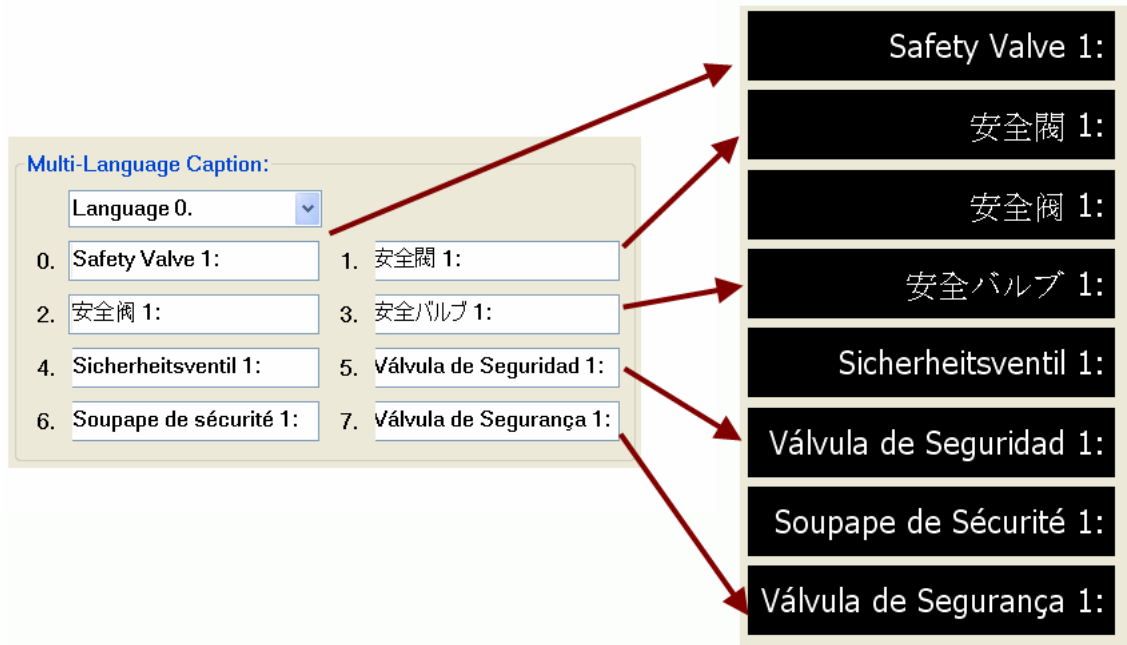
- None
- Client edge
- Static edge
- Modal frame



4.3.3.3 Caption

Each LABEL support caption text in 8 different languages. During runtime the caption text can be changed to different languages. To display a caption on the LABEL

- Uncheck the “*MSG/AI/AO/D/F Enable*” check box.
- Type in one or more text boxes (0 to 7) the caption text in different languages.
- Set the caption text font and size by clicking the “*Fonts*” tab
- Set the text and background color by clicking the color button next to “*Back Color*” and “*Font Color*” and selecting the desired color.
- Selected from the language combo box the default language text (0 to 7) to be displayed.
- Align the text by selecting DT_RIGHT, DT_CENTER or DT_LEFT from the combo box
- Set the “*Alarm Timer 0, 1, 2...*” to zero for a static caption. A blinking or flashing text to indicate an alarm is achieved by entering a number greater than zero for the “*Alarm Timer 0, 1, 2...*”. The time unit is 50 milliseconds. For example: a value of 4 causes the LABEL to flash every 200 milliseconds.



4.3.3.4 Register assignment

A LABEL control can read values or text messages from the assigned register (AI, AO, D, F, MSG). The following steps describe the mapping of a LABEL control to a register:

STEP 1: Check the “*MSG/AI/AO/D/F Enable*” check box

STEP 2: Select a proper register type for the LABEL control. Select from the combo box “*Select MSG/AI/AO/D/F*” the required register type.

Specification	Register	Register numbers
Analog Output	AO	Local AO: 0 ~ 511
Analog Input	AI	Local AI: 0 ~ 511
long integer	D	None Retain: 1 ~ 3599
		Retain: 4096 ~ 7999
Float	F	None Retain: 1 ~ 1899
		Retain: 2048 ~ 3999
Message	MSG	Retain: 1 ~ 249

STEP 3: Enter in the “*MSG/AI/AO/D/Fno →Label*” edit box a register number according to the register table.

STEP 4: Set the LABEL control update rate in the “Flash Timer 0,1,2...” edit box. The Flash timer reads values or text messages from the mapped register at the set time interval. The unit of the flash timer is 50 milliseconds. A flash timer value of two will update the LABEL display every 100 milliseconds. If the flash timer setting is zero, the control will not be updated during the program execution time. The language setting can also not be changed during run time.

STEP 5: In case a float or long integer register type (“AI/AO/D/F”) has been selected:

- Set the number of decimal places to display

STEP 6: If you want the control to flash or blink to indicate an alarm the following additional settings have to be made:

- Enter a blinking frequency value for the “*Alarm Timer 0, 1, 2...*” (Unit: 50 milliseconds). If this timer is set to zero the blinking property is disabled.
- In case a *float* or *long integer* register type (“AI/AO/D/F”) has been selected:

- Enter a safe range by assigning a value for the upper and lower limit. Once the value in the mapped register exceeds the range the control starts to flash.
- In case a **message** register type (“MSG”) has been selected:
 - Only messages in the MSG register which starts with a “#” character will flash in the LABEL control. A text is written to the MSG register by calling the following EzCore API:

```
SET_MSG(WORD MSGno, TCHAR UMSG[30]);
```

MSGno - MSG register number
UMSG[30] - Unicode message string

Example:

```
SET_MSG(100, L"# Overvoltage");
```

This function call writes the text message “# Overvoltage” to the MSG register at index 100. As the message starts with the “#” character the LABEL mapped to the MSG register 100 will blink at the interval set at “*Alarm Timer 0, 1, 2...*”

4.3.4 EzHMI ColorEdit

4.3.4.1 Description

The main task of the ColorEdit control is to enable the user to write values or strings to mapped register. When used as data entry, the data can be validated on entry to check for minimum and maximum values. The entered value is then only accepted if the value lies within the set limits.

At runtime a ColorEdit box

- writes values (float, integer) to and reads values from the assigned register (AO, D, F),
- writes a text strings of maximum 30 unicode characters to the assigned register (MSG) and reads strings from the register,
- displays a virtual keyboard for keying characters or values via touch screen.

Setting:

In the following the configuration interface of the ColorEdit control will be discussed in more details:

The screenshot shows the configuration interface for the ColorEdit control, divided into several sections:

- Appearance Styles:** Includes an "Align Test:" dropdown menu currently set to "DT_CENTER".
- Color:** Includes "Back Color:" and "Text Color:" fields. The "Text Color:" field is currently set to black.
- Register Type and Number Assignment:** Includes a "Select Input MSG/AO/D/F" dropdown menu set to "Register MSG", and three input fields for "ColorEdit --> MSG/AO/D/Fno" (00000000), "Upper Limit:" (1000000000), and "Lower Limit:" (-1000000000).
- Control Status:** Includes an "Mno(On) --> Disable ActiveX" input field set to 00000000.
- Refresh Interval (Unit 50ms):** Includes a "Flash Timer 0,1,2,...." input field set to 00000002.

4.3.4.2 Flash Timer

Refresh Interval (Unit 50ms):

Flash Timer 0,1,2.....

The ColorEdit control flash timer has basically two tasks:

1. ColorEdit input has been changed: Update the corresponding register to the new input.
2. Register has been changed: Display the new register content.

The flash timer writes to and reads from the assigned ColorEdit register at the set time interval. The unit of the flash timer is 50 milliseconds. A flash timer value of two will update the ColorEdit display and register status every 100 milliseconds.

If the flash timer setting is zero, the ColorEdit control will not be updated during the program execution time. To ensure that the ColorEdit control and the register are updated in regular intervals an integer number greater than 0 has to be entered.

4.3.4.3 Register assignment

A ColorEdit control writes to and reads from a register (MSG, AO, D, F). The following steps describe the mapping procedure:

- STEP 1: Select a proper register type for the control. Select from the combo box “*Select MSG, AO, D, F*” the required register type.
- STEP 2: Then enter in the “*ColorEdit → MSG, AO, D, Fno*” a register number according to the register table.
- STEP 3: For the register types AO, D and F limit values can be set. If an entered value exceeds the configured limit value the value is rejected.

Register Type and Number Assignment:

Select Input MSG/AO/D/F

ColorEdit --> MSG/AO/D/Fno

Upper Limit:

Lower Limit:

Control Status:

Mno(On) --> Disable ActiveX

Specification	Register	Register numbers
Analog Output	AO	Local AO: 0 ~ 511
long integer	D	None Retain: 1 ~ 3599
		Retain: 4096 ~ 7999
Float	F	None Retain: 1 ~ 1899
		Retain: 2048 ~ 3999
Message	MSG	Retain: 1 ~ 249

The control can be disabled by assigning “*Mno(on) →Disable ActiveX*” a register number and setting this register to true either by using a EzHMI SWITCH control or by calling the EzCore API:

- `SET_M (WORD Mno, bool Flag);`

This prevents the user from entering any value or character to the ColorEdit box.

4.3.4.4 Virtual Keyboard

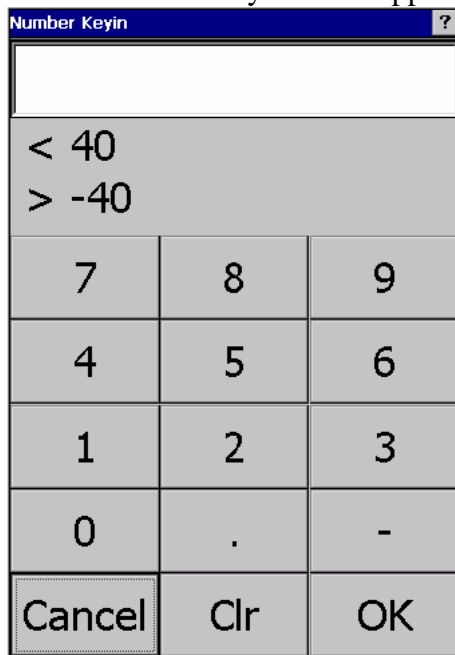
The virtual keyboard is an on-screen keyboard and can usually be operated with multiple input devices, such as the actual keyboard, a computer mouse and a touch screen. Two types of virtual keyboards are provided: a number keyboard and a character keyboard.

To enable the virtual keyboards the M8000 Register has to be set to true by either calling the EzCore API

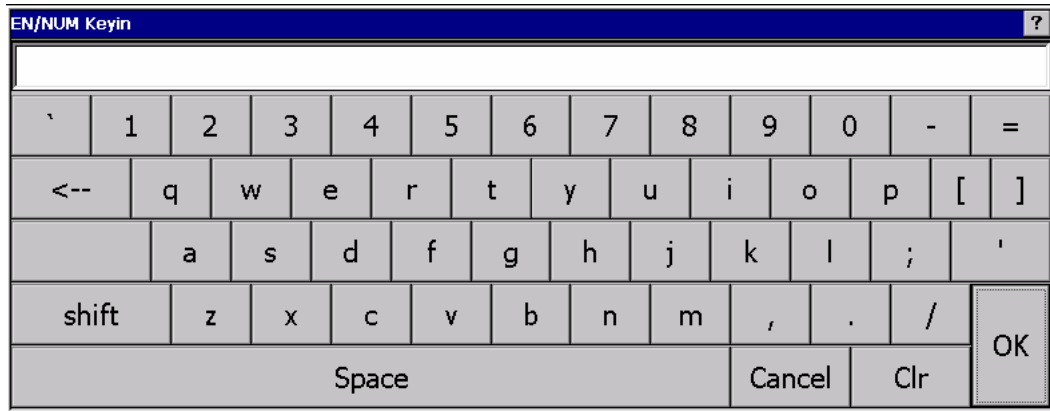
- `SET_M (8000, true);`

or by using the EzHMI SWITCH control.

If the MSG has been selected as register type for the ColorEdit control the character keyboard pops up when the control receives the focus. A selection of AO, D or F register type causes the number keyboard to appear as soon as the control is activated by the user.



Number keyboard



Character keyboard

4.3.5 EzHMI ButtonST

4.3.5.1 Description

The ButtonST control sets the flag of its assigned M register to true. It only writes data to the M register. An event is triggered when you click the button. C code has to be implemented to process the event.

The button sets the flag to true

- after it has been clicked or
- after it has been released or
- while it is being pressed (useful for jogging)

The ButtonST control supports the following layouts:

- Button with multilingual caption: The caption should describe the type of event being triggered by the button click.
- Button with graphic: The button can be provided with two bitmaps, one for indicating an enabled and the other for displaying a disabled button. The graphic shown on the button gives information regarding the status of the button.

Setting:

In the following the configuration interface of the ButtonST control will be discussed in more details:

Appearance Styles:

- Select Enable/Disable BMP
- BMP Directory (Enable): *.bmp [Set BMP...]
- BMP Directory (Disable): *.bmp [Set BMP...]
- BMP Style: Image->FULL
- BMP Position: Button->LEFT
- Boundary Offset: 2
- Color:**
 - Back Color: [Black]
 - Font Color: [White]

Multi-Language Caption:

Caption: [Language 0.]

0. Caption0	1. Caption1
2. Caption2	3. Caption3
4. Caption4	5. Caption5
6. Caption6	7. Caption7

Register Type and Number Assignment:

Button(Down) --> Mno(On)	00000000
Button(Down/Up) --> Mno(On/Off)	00000000
Button(Up) --> Mno(On)	00000000

Control Status:

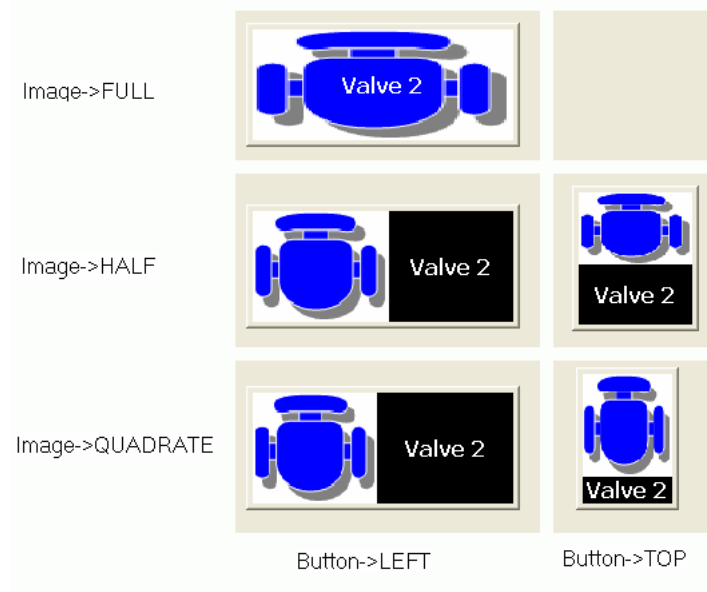
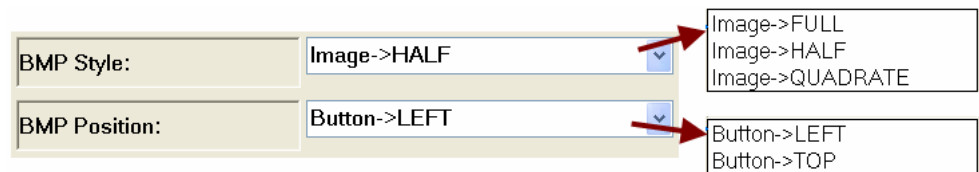
Mno(On) --> Disable ActiveX	00000000
-----------------------------	----------

4.3.5.2 ButtonST Appearances

A ButtonST can have an image, caption and boundary on its surface.

The following display styles are being supported by the ButtonST control:

- FULL: The button surface area is filled with the attached image.
- HALF: Half of the button surface is filled with the image. The position of the image on the surface can be set:
 - LEFT
 - TOP
- QUADRATE: The image will be displayed in a square. The position of the square on the button surface have to selected:
 - LEFT
 - TOP
- BMP Offset: provides a button boundary. Boundary width is measures in pixels.



4.3.5.3 Caption

Each ButtonST support caption text in 8 different languages. During runtime the caption text can be changed to a different language. To display a caption on the ButtonST

- Type for each text box (0. to 7.) the caption text in a different language. Each text box represents a different language.
- Set the caption text font and size by clicking the “*Fonts*” tab
- Set the text and background color by clicking the color button next to “*Back Color*” and “*Font Color*”.
- Selected from the language combo box (Language 0., Language 1., etc) the default language (0 to 7) to be displayed.

Multi-Language Caption:

Caption: Language 0. ▼

0.	<input type="text" value="Valve 2"/>	1.	<input type="text" value="閥門 2"/>
2.	<input type="text" value="閥門 2"/>	3.	<input type="text" value="バルブ 2"/>
4.	<input type="text" value="Ventil 2"/>	5.	<input type="text" value="Válvula 2"/>
6.	<input type="text" value="Valve 2"/>	7.	<input type="text" value="Válvula 2"/>



4.3.5.4 Bitmap Attachment

Each ButtonST control can be enabled and disabled during runtime. For each status a different bitmap can be attached to the button.

Procedure to load a bitmap into a ButtonST:

Appearance Styles:

Select Enable/Disable BMP

BMP Directory (Enable): *.bmp Set BMP...

BMP Directory (Disable): *.bmp Set BMP...

BMP Style: Image->FULL v

BMP Position: Button->LEFT v

Boundary Offset: 2

STEP 1: Create a bitmap picture with a graphic program and save it as a *.bmp* file.

STEP 2: Load the bitmap file to the ButtonST by clicking the “*Set BMP*” button and selecting the desired image. Make sure that the data format of the bitmap is *.bmp*.

STEP 3: Copy the bitmap file to the following directory on the WinCon or MPac:

- for WinCon: “\CompactFlash\EzProg-I\EzHMI\BMP”
- for MPac: “\System_Disk\EzProg-I\EzHMI\BMP”

4.3.5.5 Register assignment

Each button control can be linked to three different M register numbers.

Register Type and Number Assignment:

Button(Down) --> Mno(On)	41
Button(Down/Up) --> Mno(On/Off)	40
Button(Up) --> Mno(On)	42

Control Status:

Mno(On) --> Disable ActiveX	00000000
-----------------------------	----------

1. “*Button(Down) → Mno(on)*”: Clicking the button sets the assigned register to true.
2. “*Button(Up) → Mno(on)*”: Releasing the button sets the linked register to true.
3. “*Button(DownUp) → Mno(on/off)*”: Clicking and keeping the button down sets the linked register to true. As soon as the button is being released the register will be set to false.

For “**Button(Down) → Mno(on)**” and “**Button(Up) → Mno(on)**” register the flag can only be set to true by the control. The register has to be reset either by using the EzHMI SWITCH control or true by either calling the *SET_M()* API.

The following code, which can be implemented in the UserThread, RTSR or ISR, shows how the change of a register status (here button down action is attached to M register 41) can trigger an event and how to reset the status:

```
if( GET_Ma(41) )
{
    //implement here your event code
    //...

    //Reset the the status of register M41
    SET_M(41, false);
}
```

Each ButtonST supports two states during runtime: enabled and disabled. An enabled button can be activated (clicked) while a disable button allows no user input. Assign the button a M register number by entering a value for “**Mno(on) → Disable ActiveX**”. This register number determines the status of the button. A true enables and a false disables the button. The status of the register has to be set either by using a EzHMI control (e.g. SWITCH) or by calling the *SET_M()* API.

4.3.6 EzHMI Image Control

4.3.6.1 Description

The Image control allows the insertion of a bitmap image to the user interface. The image can be a background image representing a plant or control system. Other controls like LED and switches can be placed on top of the image. The Image control can also be configured in such a way that it move across the user interface to a target position during runtime. Furthermore the Image control can handle click events. C code has to be implemented to process the event.

The Image control supports the following properties:

- Adding a bitmap Image to the user interface
- Replacing images during runtime
- Moving the images during runtime
- Clicking events

Setting:

In the following the configuration interface of the Image control will be discussed in more details:

Appearance Styles:

BMP Directory: *.bmp

Register Type and Number Assignment:

Dynamic Change Image:

Dno --> Change BMP (PICxxx.bmp)	00000000
Mno(On) --> Active Change	00000000
Image Position(Left and Top):	
Input X(Dno), Y(Dno+1) --> Move to (X, Y)	00000000
Click Down/Up --> Mno(On/Off)	00000000
Click Down --> Mno(On)	00000000
Click Up --> Mno(On)	00000000

Control Status:

Mno --> Hide ActiveX	00000000
----------------------	----------

4.3.6.2 Bitmap Attachment

4.3.6.2.1 Adding Default Image

A default bitmap image is an image shown on the user interface directly after program start.

The following procedure describes the loading of a default bitmap:

Appearance Styles:

BMP Directory: *.bmp

- STEP 1: Create a bitmap picture with a graphic program and save it as a **.bmp** file.
(Notice: WinCon supports only a color depth of 16 bits)

STEP 2: Load the bitmap file to the Image control by clicking the “**File...**” button and selecting the desired image. Make sure that the data format of the bitmap is **.bmp**.

If you do not want to display at program start, just leave the “**BMP Directory:**” empty.

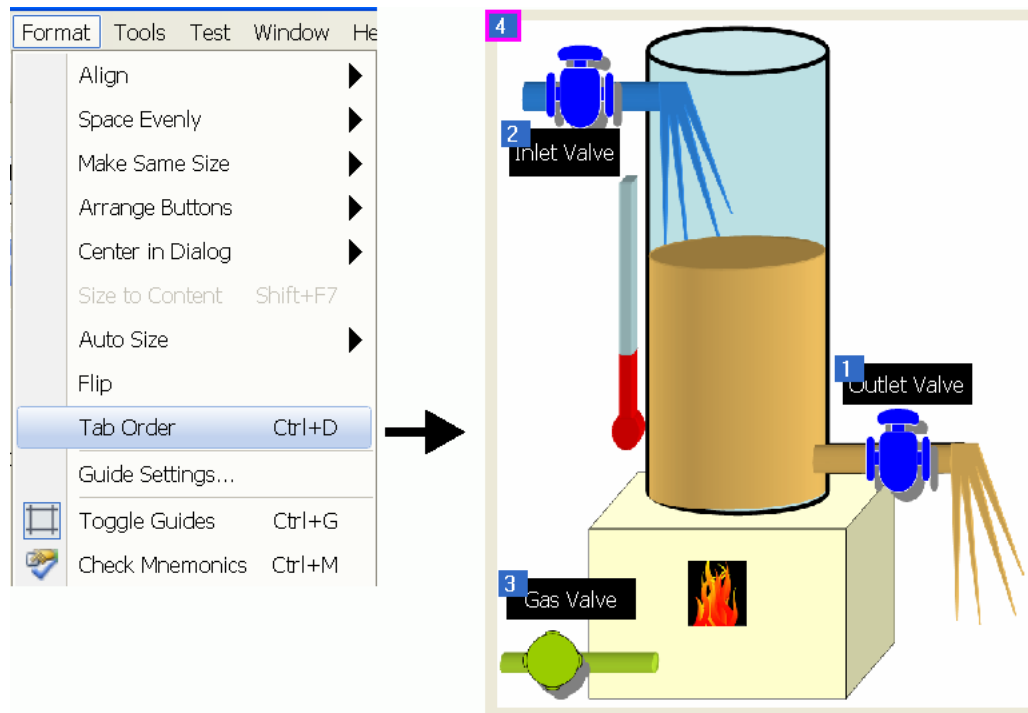
STEP 3: Upload the bitmap file to the following directory on the WinCon or MPac:

- for WinCon: “\CompactFlash\EzProg-I\EzHMI\BMP”
- for MPac: “\System_Disk\EzProg-I\EzHMI\BMP”

Notice:

If you want to add a control (e.g. SWITCH, EzKnob, LED, etc) on top of the image make sure that the tab order number of the added control is lower than the Image control. To view and alter the tab order number in VS2008, click **Format → Tab Order**. The tab number for each control is displayed in a blue box. Change a number of a control by clicking on it. Click the Image control representing the background after all the other controls have been clicked. The Image control will show the highest tab number.

The following figure shows four controls with their respective tab numbers. Three EzHMI Label controls with the numbers 1, 2, and 3 and an Image control with the tab number 4. Image control displays a tank, combustion chamber and inlet and outlet valves. The Image control forms the background and therefore is set to the highest tab number 4.



4.3.6.2.2 Changing Image at Runtime

A default bitmap image can be replaced during runtime by another image.

The following steps demonstrate the procedure:

Dynamic Change Image:

Dno --> Change BMP (PICxxx.bmp)	3000
Mno(On) --> Active Change	1
Image Position(Left and Top):	
Input X(Dno), Y(Dno+1) --> Move to (X, Y)	00000000

STEP 1: Create one or more bitmap pictures and save them as .bmp files by using the following filename convention: **PICxxx.bmp**

The filename should always start with **PIC**

Enter for **xxx** any number in the range from 1 to 2147483647

Examples:

- PIC1.bmp
- PIC10.bmp
- PIC1356.bmp

(Notice: WinCon supports only a color depth of 16 bits)

STEP 2: Download the bitmap files to the following directory on the WinCon or MPac:

for WinCon: “\CompactFlash\EzProg-I\EzHMI\BMP”

for MPac: “\System_Disk\EzProg-I\EzHMI\BMP”

STEP 3: Assign “**Dno → Change BMP(PICxxx.bmp)**” a D register number. This register determines which picture is going to be displayed on the Image control.

Dno --> Change BMP (PICxxx.bmp)	3000
---------------------------------	------

Specification	Register	Register numbers
long integer	D	None Retain: 1 ~ 3599
		Retain: 4096 ~ 7999


The number indicated by **xxx** in STEP 1: have to be written to the D register number. The register value can be changed either programmatically (UserThread, RTSR, ISR) or by using an EzHMI control (e.g. ColorEdit control).

Examples:

In this example it is assumed register number 3000 has been mapped to “*Dno →Change BMP(PICxxx.bmp)*”. The register value is changed by calling the API *SET_D()*:

- PIC**1**.bmp → SET_D(3000, **1**);
- PIC**10**.bmp → SET_D(3000, **10**);
- PIC**1356**.bmp → SET_D(3000, **1356**);

STEP 4: Assign “*Mno(On) →Active Change*” a M register number. If this register changes its value from false to true the control displays the image assigned to the D register number in STEP 3:.. The Image control sets the M flag automatically back to false after updating the image.



Specification	Register	Register numbers	
Flag	M	None Retain	1 ~ 6999
		Retain	8192 ~ 15999

STEP 5: Implement the code for changing the image.

In the following example the “*Dno →Change BMP(PICxxx.bmp)*” has been mapped to D register number 3000 and “*Mno(on) →Active Change*” to M register number 1.

Alternative 1:

Add the following code to the UserThread, RTSR or ISR:

```
//Assign image PIC1356 to D register 3000:
SET_D(3000, 1356);

//Inform the Image control to display PIC1356
// by setting the M register 1 to true
SET_M(1, true);
```

Alternative 2:

Add a SWITCH and a ColorEdit control to the dialog.

- Set the SWITCH control properties as shown in the following figure:

Register Type and Number Assignment:

Y/M Enable

Select Y/M: Register M

Switch(On/Off) --> Y/Mno: 1

Mno(On) --> Lamp: 00000000

Control Status:

Mno(On) --> Disable ActiveX: 00000000

Refresh Interval (Unit 50ms):

Flash Timer 0,1,2.....: 2

- Set the ColorEdit control properties as follows:

Register Type and Number Assignment:

Select Input MSG/AO/D/F: Register D

ColorEdit --> MSG/AO/D/Fno: 3000

Refresh Interval (Unit 50ms):

Flash Timer 0,1,2.....: 2

At runtime the user can enter a valid picture number in the ColorEdit box. After setting the SWITCH to true the Image control will be updated with the selected image.

4.3.6.2.3 Changing the Image position at runtime

During runtime the position of the Image control can be changed. In this case the control can represent a moving object (e.g. assembly line) on the screen.

The following steps are required:

- STEP 1: Add an image to the control as described in the previous two section
- STEP 2: Select two D register for storing the image x and y position. Only the register number for the x position has to be entered. The number following x register number is automatically assigned to the y position. Therefore you have to make sure that this number is not being reserved for a different purpose.

Image Position(Left and Top):

Input X(Dno), Y(Dno+1) --> Move to (X, Y)

Specification	Register	Register numbers	
long integer	D	None Retain:	1 ~ 3599
		Retain:	4096 ~ 7999

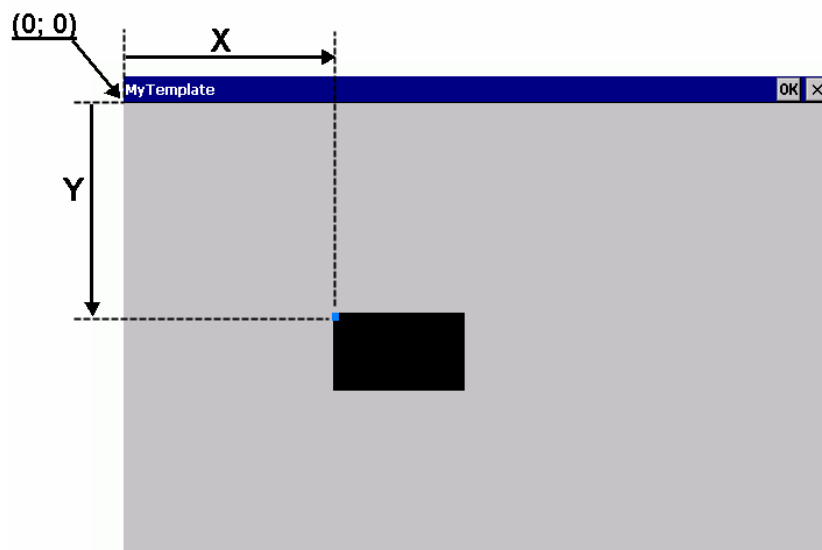
STEP 3: Assign “*Mno(on) → Active Change*” a M register number. If this register changes its value from false to true the control reads the target position and moves to the new position. The Image control sets the M flag automatically back to false after the reading operation has finished.

Mno(On) --> Active Change

Specification	Register	Register numbers	
Flag	M	None Retain:	1 ~ 6999
		Retain:	8192 ~ 15999

STEP 4: Implement the code for changing the image position. In the following example “*InputX(Dno),Y(Dno+1) → Move to(X,Y)*” has been mapped to D register number 4 and “*Mno(on) → Active Change*” to M register number 1.

The control uses a coordinate system similar to the Cartesian's but the origin is located on the top left corner of the dialog box. Using this coordinate system, any point can be located by its distance (unit = pixel) from the top left corner of the dialog box.



Alternative 1:

Add the following code to the UserThread, RTSR or ISR:

```
//New image position:  
// x-axis:  
SET_D(4, 15); // x = 15  
// y-axis:  
SET_D(5, 20); // y = 20  
  
//Inform the Image control that the position has been  
// changed:  
SET_M(1, true);
```

Alternative 2:

Add a SWITCH and two ColorEdit control to the dialog.

- Set the SWITCH control properties as shown in the following figure:

Register Type and Number Assignment:

Y/M Enable

Select Y/M: Register M

Switch(On/Off) --> Y/Mno: 1

Mno(On) --> Lamp: 00000000

Control Status:

Mno(On) --> Disable ActiveX: 00000000

Refresh Interval (Unit 50ms):

Flash Timer 0,1,2.....: 2

- Set the ColorEdit control properties for the **X- axis** as follows:

Register Type and Number Assignment:

Select Input MSG/AO/D/F: Register D

ColorEdit --> MSG/AO/D/Fno: 4

Refresh Interval (Unit 50ms):

Flash Timer 0,1,2.....: 2

- Set the ColorEdit control properties for the **Y- axis** as follows:

Register Type and Number Assignment:

Select Input MSG/AO/D/F: Register D

ColorEdit --> MSG/AO/D/Fno: 5

Refresh Interval (Unit 50ms):

Flash Timer 0,1,2.....: 2

At runtime the user can change the position of the image by entering values for the x- and y-axis in the ColorEdit boxes. Set the SWITCH to true in order to move the Image control to the new position.

4.3.6.3 Events

The Image control generates three types of events. To make use of an event assign it to an M register numbers.

Click Down/Up --> Mno(On/Off)	00000000
Click Down --> Mno(On)	00000000
Click Up --> Mno(On)	00000000

1. “**Click Down → Mno(On)**”: Clicking the image sets the assigned register to true.
2. “**Click Up → Mno(On)**”: Releasing the button after clicking the image sets the linked register to true.
3. “**Click Down/Up → Mno(On/Off)**”: Clicking the image and keeping the button down sets the linked register to true. The register will be set to false as soon as the button is being released.

Click Down/Up --> Mno(On/Off)	00000000
Click Down --> Mno(On)	50
Click Up --> Mno(On)	00000000

Specification	Register	Register numbers
Flag	M	None Retain: 1 ~ 6999
		Retain: 8192 ~ 15999

The flag for the “*Click Down →Mno(On)*” and the “*Click Up →Mno(On)*” registers set to true by the control itself. The register has to be reset to false either by using an EzHMI SWITCH control or by calling the *SET_M()* API.

Example:

The following code, which can be implemented in the UserThread, RTSR or ISR, shows how to use a flag generated by a click event. It is important to reset the flag to false in order to catch the next click event.

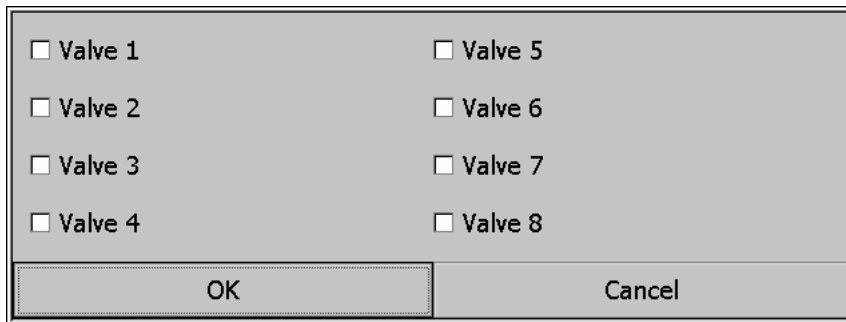
```
if( GET_Ma(50) )
{
    //implement here your event code
    //...

    //Reset the the status of register M41
    SET_M(50, false);
}
```


4.3.7 EzHMI ColorCheck

4.3.7.1 Description

The ColorCheck control is associated to a group of 8 check boxes. The checkbox group appears on a separate window form when the ColorCheck control is being clicked. None, one or more checkboxes may be checked by the user according to the information they wish to send with the form. A checkbox indicates whether a particular item is selected or not. A checkbox can also be used as a switch to present a on/off , open /closed or true/false status to the user.



The image shows a dialog box with a light gray background. It contains eight checkboxes arranged in two columns. The left column has checkboxes for 'Valve 1', 'Valve 2', 'Valve 3', and 'Valve 4'. The right column has checkboxes for 'Valve 5', 'Valve 6', 'Valve 7', and 'Valve 8'. At the bottom of the dialog box, there are two buttons: 'OK' on the left and 'Cancel' on the right.

The ColorCheck control supports the following properties:

- Multi-Language selection
- Selection of one or more options from a set of alternatives
- Selection change notification

Setting:

In the following the configuration interface of the ColorCheck control will be discussed in more details:

Multi-Language Caption:

Language 0. ▾

0. Caption0 1. Caption1 2. Caption2 3. Caption3 4. Caption4 5. Caption5 6. Caption6 7. Caption7

List Items:

0. 0.Item0	1. 0.Item1	2. 0.Item2	3. 0.Item3	4. 0.Item4	5. 0.Item5	6. 0.Item6	7. 0.Item7
0. 1.Item0	1. 1.Item1	2. 1.Item2	3. 1.Item3	4. 1.Item4	5. 1.Item5	6. 1.Item6	7. 1.Item7
0.	1.	2.	3.	4.	5.	6.	7.
0.	1.	2.	3.	4.	5.	6.	7.
0.	1.	2.	3.	4.	5.	6.	7.
0.	1.	2.	3.	4.	5.	6.	7.
0.	1.	2.	3.	4.	5.	6.	7.
0.	1.	2.	3.	4.	5.	6.	7.

Color:

Back Color

Font Color

Register Type and Number Assignment:

Check Chang--> Mno(On)

Check Select --> Bno

Control Status:

Mno(On)--> Disable ActiveX

4.3.7.2 Caption

4.3.7.2.1 ColorCheck Caption

The purpose of the caption is to act as a header for the item list to enable the user to identify the list more easily. As with most EzHMI ActiveX the ColorCheck supports caption text in 8 different languages. In the control property sheet each text box (0. to 7.) represents a different language. During runtime the caption text can be changed to a different language. To display a caption on the ColorCheck:

- Replace the default caption text (Caption0, Caption1, ...). Each textbox represents a different language.
- Set the caption text font and size by clicking the “**Fonts**” tab
- Set the text and background color by clicking the color button next to “**Back Color**” and “**Font Color**”.
- Selected from the language combo box (Language 0., Language 1., etc.) the default language (0 to 7).

Multi-Language Caption:

Language 0. ▾

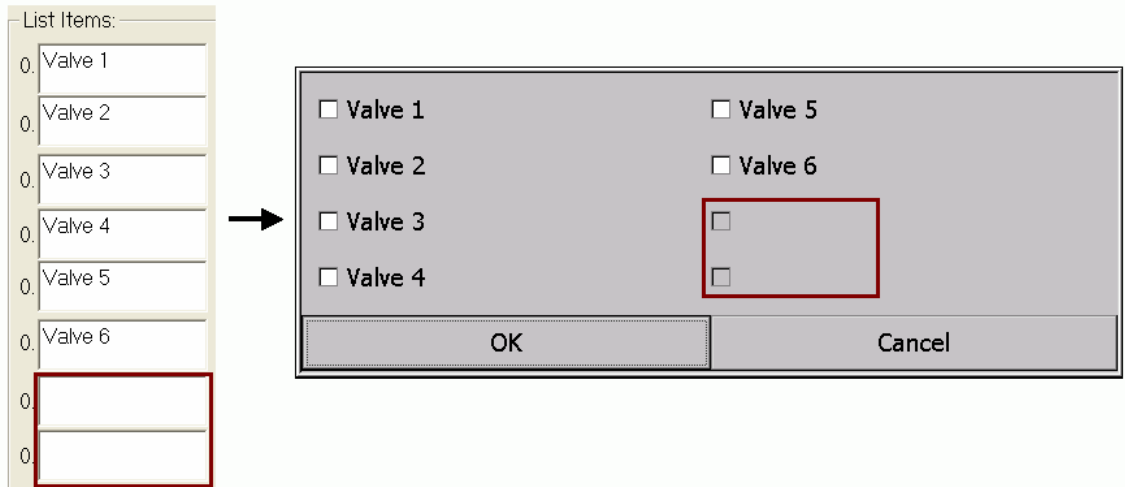
0. Valve 1. 閥門 2. 閥門 3. バルブ 4. Ventil 5. Válvula 6. Valve 7. Válvula

4.3.7.2.2 Checkbox Caption

The ColorCheck dialog window displays 8 checkboxes. Each checkbox has to be provided with a caption. A checkbox with no caption will automatically be disabled and the user can not enable or check it during runtime. Multilanguage support for checkbox caption is provided. Each column in the “List Items” group represents the caption text of the checkboxes in a different language. A column consists of eight textboxes each storing the text for the eight checkbox caption. It is not possible to change the color, font or size of the checkbox caption text.

	Language 0	Language 1	Language 2	Language 3	Language 4	Language 5	Language 6	Language 7
Checkbox 0:	0 Valve 1	1 閥門 1	2 閥門 1	3 バルブ 1	4 Ventil 1	5 Válvula 1	6 Valve 1	7 Válvula 1
Checkbox 1:	0 Valve 2	1 閥門 2	2 閥門 2	3 バルブ 2	4 Ventil 2	5 Válvula 2	6 Valve 2	7 Válvula 2
Checkbox 2:	0 Valve 3	1 閥門 3	2 閥門 3	3 バルブ 3	4 Ventil 3	5 Válvula 3	6 Valve 3	7 Válvula 3
Checkbox 3:	0 Valve 4	1 閥門 4	2 閥門 4	3 バルブ 4	4 Ventil 4	5 Válvula 4	6 Valve 4	7 Válvula 4
Checkbox 4:	0 Valve 5	1 閥門 5	2 閥門 5	3 バルブ 5	4 Ventil 5	5 Válvula 5	6 Valve 5	7 Válvula 5
Checkbox 5:	0 Valve 6	1 閥門 6	2 閥門 6	3 バルブ 6	4 Ventil 6	5 Válvula 6	6 Valve 6	7 Válvula 6
Checkbox 6:	0	1 閥門 7	2 閥門 7	3 バルブ 7	4 Ventil 7	5 Válvula 7	6 Valve 7	7 Válvula 7
Checkbox 7:	0	1 閥門 8	2 閥門 8	3 バルブ 8	4 Ventil 8	5 Válvula 8	6 Valve 8	7 Válvula 8

In the following figure only the first six of the eight checkboxes are assigned a caption text. Therefore on the checkbox group window the checkboxes provided with a name are enabled and can receive use input the others are disabled.



4.3.7.3 Register assignment

The checkboxes within the group are sequentially numbered starting from zero. The statuses of the eight checkboxes are written to the linked BYTE register. The least significant bit represents the status of the first checkbox in the group, bit 1 represents the second checkbox and so forth. Enabling/disabling one or more checkboxes writes the new status to the mapped B register.

Register Type and Number Assignment:

Check Change --> Mno(On)	00000000
Check Select --> Bno	00000000

Control Status:

Mno(On) --> Disable ActiveX	00000000
-----------------------------	----------

The checkbox status is linked to a register by entering for “*Check Select → Bno*” a B register number.

Furthermore it is possible to indicate with a notification flag that changes were made to the ColorCheck by the user. Assign “*Check Change → Mno(On)*” to a M register number. Every time the user changes a checkbox status this register will be set to true after the checkbox group window has been closed.

Register Type and Number Assignment:

Check Change --> Mno(On)	111
Check Select --> Bno	1

Control Status:

Mno(On) --> Disable ActiveX	00000000
-----------------------------	----------

Specification	Register	Register numbers
Flag	M	None Retain: 1 ~ 6999
		Retain: 8192 ~ 15999
BYTE	B	None Retain: 1 ~ 699
		Retain: 1024 ~ 2047

The following code snippet, which can be implemented in the UserThread, RTSR or ISR, shows how to read the ColorCheck status. The notification flag “*Check Change → Mno(On)*” is mapped to M register 111 and the ColorCheck status “*Check Select → Bno*” to B register 1.

```

bool bCheckBox[8];

// STEP 1: Check whether the ColorCheck control status has changed
if (GET_Ma(111)== true)
{
    // STEP 2: Read the status of each checkbox
    for(int i=0; i< 8; i++)
    {
        bCheckBox[i] = GET_B(1) & (1<< i);
    }

    // STEP 3: Reset the notification flag to false
    SET_M(111, false);

    // STEP 4:
    //Implement code to handle the new setting

    //If checkbox 0 is checked
    if(bCheckBox[0]== true)
    {
        // add your code
    }

    //If checkbox 1 is checked
    if(bCheckBox[1]== true)
    {
        // add your code
    }

    ....
}

```

The status of the ColorCheck can not only be changed via the graphic interface but also by directly changing the value of the assigned B register number in the program. Use the following function:

```
SET_B(WORD Bno, BYTE Val);
```

The only way to set the checkbox group to a default selection at program start is by calling the SET_B() function in the MFC OnInitDialog() function.

Example:

The ColorCheck status “*Check Select → Bno*” is linked to B register 1. The following function call unchecks all the checkboxes.

```
SET_B(1, 0);
```

4.3.8 EzHMI ColorRadio

4.3.8.1 Description

The ColorRadio control represents a group of 10 radio buttons which appears on a separate window form when the ColorRadio control has been clicked. Radio buttons work just like checkboxes except that they are mutually exclusive to one another. When the operator selects an option, any previously selected option in the same group becomes deselected. A ColorRadio control therefore allows only one single item out of the group to be selected.

By default one option is selected, even if the user has not made any input.

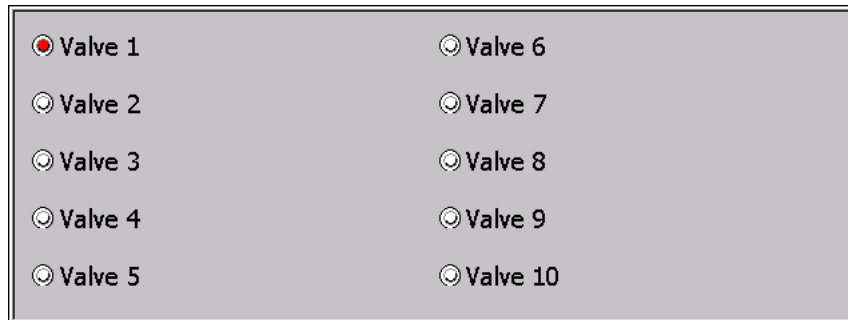


Figure 8: ColorRadio window

The ColorCheck control supports the following properties:

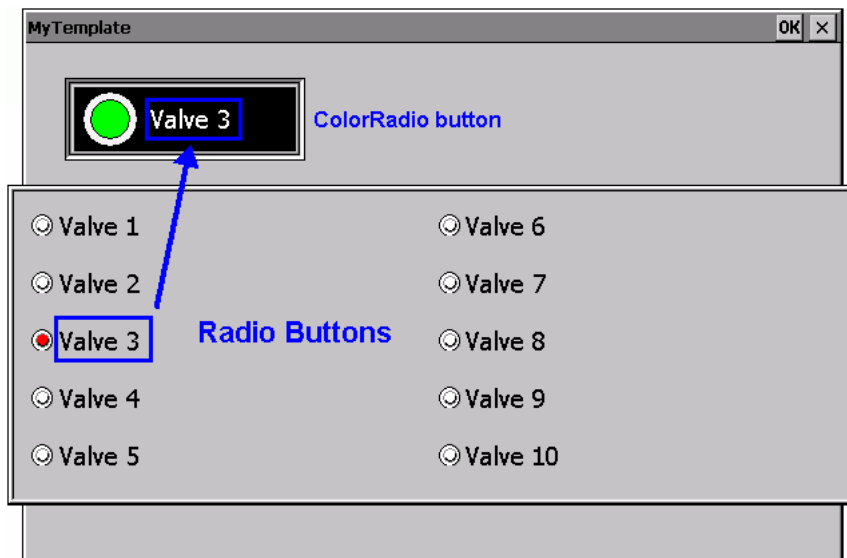
- Multi-Language selection
- Selection of one options from a set of alternatives
- Selection change notification

Setting:

In the following the configuration interface of the ColorRadio control will be discussed in more details:

4.3.8.2 Caption

Each radio button on the radio window needs to be provided with a caption name. A radio button without a name will be shown as a disabled button and is not available for user input. The name of the selected option button will be shown automatically on ColorRadio control.



The Radio button names have to be entered in the items list. Each column in the “List Items” group represents the caption name of the radio button in a different language. A column consists of ten edit boxes each storing the name of the respective radio button.

Option Button Caption	Multilanguage							
	Language 0	Language 1	Language 2	Language 3	Language 4	Language 5	Language 6	Language 7
Radio Button 0: 0	Valve 1	閥門 1	閥門 1	バルブ 1	Ventil 1	Válvula 1	Valve 1	Válvula 1
Radio Button 1: 0	Valve 2	閥門 2	閥門 2	バルブ 2	Ventil 2	Válvula 2	Valve 2	Válvula 2
Radio Button 2: 0	Valve 3	閥門 3	閥門 3	バルブ 3	Ventil 3	Válvula 3	Valve 3	Válvula 3
Radio Button 3: 0	Valve 4	閥門 4	閥門 4	バルブ 4	Ventil 4	Válvula 4	Valve 4	Válvula 4
Radio Button 4: 0	Valve 5	閥門 5	閥門 5	バルブ 5	Ventil 5	Válvula 5	Valve 5	Válvula 5
Radio Button 5: 0	Valve 6	閥門 6	閥門 6	バルブ 6	Ventil 6	Válvula 6	Valve 6	Válvula 6
Radio Button 6: 0	Valve 7	閥門 7	閥門 7	バルブ 7	Ventil 7	Válvula 7	Valve 7	Válvula 7
Radio Button 7: 0	Valve 8	閥門 8	閥門 8	バルブ 8	Ventil 8	Válvula 8	Valve 8	Válvula 8
Radio Button 8: 0	Valve 9	閥門 9	閥門 9	バルブ 9	Ventil 9	Válvula 9	Valve 9	Válvula 9
Radio Button 9: 0	Valve 10	閥門 10	閥門 10	バルブ 10	Ventil 10	Válvula 10	Valve 10	Válvula 10

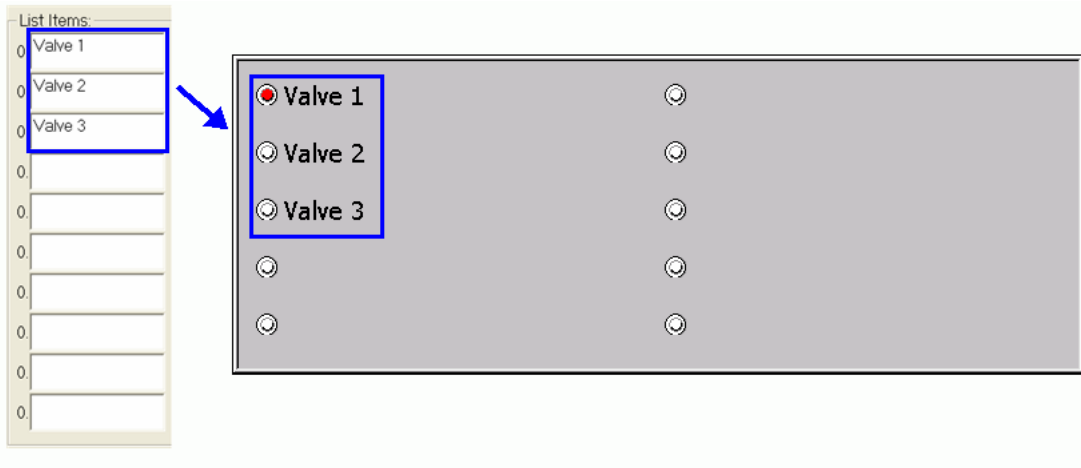
It is possible to set the color, font and text size of the ColorRadio button but this setting can not be done for the buttons on the ColorRadio window.

Altering the appearance of the ColorRadio button:

- Set the caption text font and size by clicking the “**Fonts**” tab
- Set the text and background color by clicking the “**Back Color**” and “**Font Color**” buttons.
- Selected from the language combo box (Language 0., Language 1., etc.) the default language text (0 to 7).

The following figure shows how to disable a radio button by simply not entering a name for the respective radio button. The first three radio buttons have a caption text.

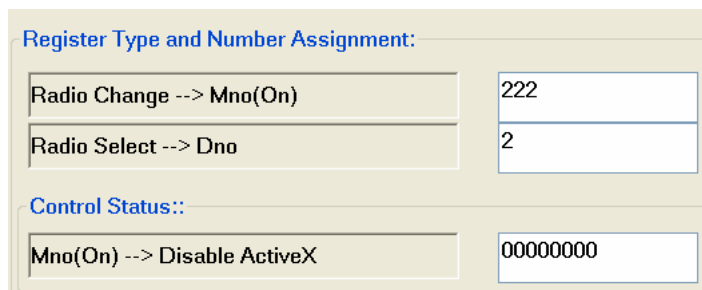
Therefore the ColorRadio window shows only three enabled radio button with names and the others are disabled. The unused buttons can not be removed from the window.



4.3.8.3 Register assignment

The ColorRadio window has to be mapped to a D register. This is done by entering for “**Radio Select → Dno**” a valid D register number. The radio buttons on the ColorRadio window are numbered from zero to nine. After selecting a radio button the number of this button will be written to the assigned register. For example, when button 3 is selected the register will store the number 3.

The default value of a radio button has to be set in the `USER_INITIAL()` function in the `A_Template.ccp` file of the `EzTemplate`.



Example: The “**Radio Select → Dno**” is linked to D Register 2. The radio button no. 7 is chosen to be the default selected button; therefore it is required to write the value 7 to the register.

```
void USER_INITIAL()
```



```

{
// Add your initialisation code:
SET_D( 2, 7);
}

```

Furthermore it is possible to indicate with a notification flag that changes were made to the ColorRadio window by the user. Assign “*Radio Change → Mno(On)*” to a M register number. Every time the user selects a different radio button, this register will be set to true. The register have to be reset to false by calling *SET_M(RegisterNo, false)*.

A ColorRadio is disabled by assigning “*Mno(On) → Disable ActiveX*” a M register number and set this register to false by using the EzHMI SWITCH control or by calling the *SET_M()* API in the program. A disabled ColorRadio does not process any user input.

The following code, which can be implemented in the UserThread, RTSR or ISR, shows how to read the ColorRadio status. The notification flag “*Check Change → Mno(On)*” is mapped to M register 222 and the radio button selection “*Check Select → Dno*” to B register 2.

```

// STEP 1: Check whether a new radio button has been selected
if (GET_Ma(222)== true)
{
// STEP 2: Reset the notification flag to false
SET_M(222, false);

// STEP 3:
//Implement code to handle the new setting

switch (GET_D(2))
{
case 0: //option button no 0 is selected
// add your code
break;

case 1: //option button no 1 is selected
// add your code
break;

case 2: //option button no 2 is selected
// add your code
break;

.
.
}
}

```

```
    case 9: //option button no 9 is selected
            // add your code
            break;
    }
}
```

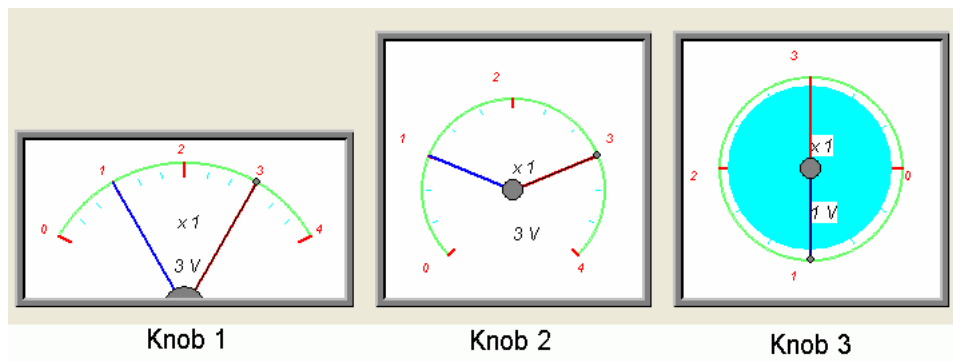
4.3.9 EzHMI EzKnob

4.3.9.1 Description

The EzKnob visualizes data like pressure, temperature, volt, etc. in form of Multi-Needle Gauge. The EzKnob object has two features:

- it displays the value of the source register in an intuitive way,
- manual input by dragging the needle (via mouse or touch screen) to the required position automatically updates the destination register with the value of the EzKnob.

The EzKnob is available in different graphic looks with a range of configurable parameters in order to be customized according designer's needs. Text regions on the dial face can show tick value, units and scale multiplier. The color of each part of the dial is configurable. If the EzKnob source register is connected to an AIO channel, the dial will automatically move its needle to match the value of the channel with no need for any application code.



Setting:

In the following the configuration interface of the EzKnob control will be discussed in more details:

4.3.9.2 Appearances

1. Styles:

- Knob 1
- Knob 2
- Knob 3

2. Frame:

Select between EzKnob with or without a frame

3. Tick configuration:

The configuration allows two different sized ticks (major and minor ticks) to be placed at regular intervals along the Scale. The control allows the following ticks settings:

- Number of ticks
- Tick color
- Tick minimum and maximum value

4. EzKnob color:

Each component of the dial may have a different color:

- Indicator or needle
- Track
- Scale text
- dial background
- Tick label
- Tick unit label
- Scale factor label

5. Label text font and position:

The text font and position of the following labels can be set:

- Tick label
- Tick unit label
- Scale factor label

4.3.9.3 Register assignment

The EzKnob object can be set to one of the following mode at a time:

- Only read and display register data.
- Write data to a register by letting the user change the needle or indicator position and reading data from the register.

4.3.9.3.1 Display register data

The EzKnob control is able to display data from an AI, AO, D or F register type.

STEP 1: Select “**Output**” as the control type to put the control into read mode.



STEP 2: Select a register type from the “**Select Input AI/AO/D/Fno**” combo box

Link the control to the selected register type by entering a number for “AI/AO/D/Fno →Knob Value”.

Specification	Register	Register numbers
Analog Output	AO	Local AO: 0 ~ 511
Analog Input	AI	Local AI: 0 ~ 511
long integer	D	None Retain: 1 ~ 3599
		Retain: 4096 ~ 7999
Float	F	None Retain: 1 ~ 1899
		Retain: 2048 ~ 3999

STEP 3: Enter a refresh time interval value for the “Flash Timer 0, 1, 2,..”. This enables the EzKnob to read the assigned register and update the display to the current reading at the set time interval.

STEP 4: Flashing alarm: Alarm occurs when the range limits of the control has been exceeded. In an event of an alarm the background of the control starts to flash. The following settings are required:

- i. Specify the upper and lower limit
- ii. Define the flashing frequency (“Alarm Timer 0,1,2...”). The value must be greater than zero otherwise the alarm is disabled.

Input Limited or Alarm Level of Output

Upper Limit: 3

Lower Limit : 1

Refresh Interval (Unit 50ms):

Alarm Timer 0,1,2...: 2

Flash Timer 0,1,2...: 2

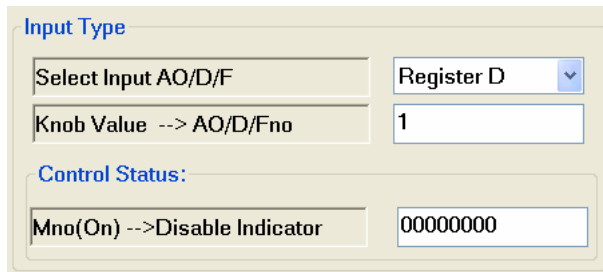
4.3.9.3.2 Enter and display register data

The EzKnob control is able to write data to and read data from an AO, D or F register type. By changing the position of the gauge needle the user can write data to the register. In addition new register data are displayed by the gauge.

STEP 1: Select “**Input**” as the control type to put the control into read/write mode.



STEP 2: Select a register type from the “**Select Input AO/D/Fno**” combo box. Link the control to the selected register type by entering a number for “Knob Value → AO/D/Fno”.



STEP 3: Enter a refresh time interval value for the “**Flash Timer 0, 1, 2,..**”. This enables the EzKnob to read/write the assigned register and update the display to the current reading at the set time interval.

STEP 4: Flashing alarm: To enable flashing when a range has been exceeded the following settings are required:

- i. Specify the upper and lower limit
- ii. Define the flashing frequency (“Alarm Timer 0,1,2...”). The value must be greater than zero otherwise the alarm is disabled.

STEP 5: The EzKnob can be disabled for user input during runtime. Enter a M register number for “**Mno(On) →Disable Indicator**”. Setting this flag to true during runtime will disable the control for user input.

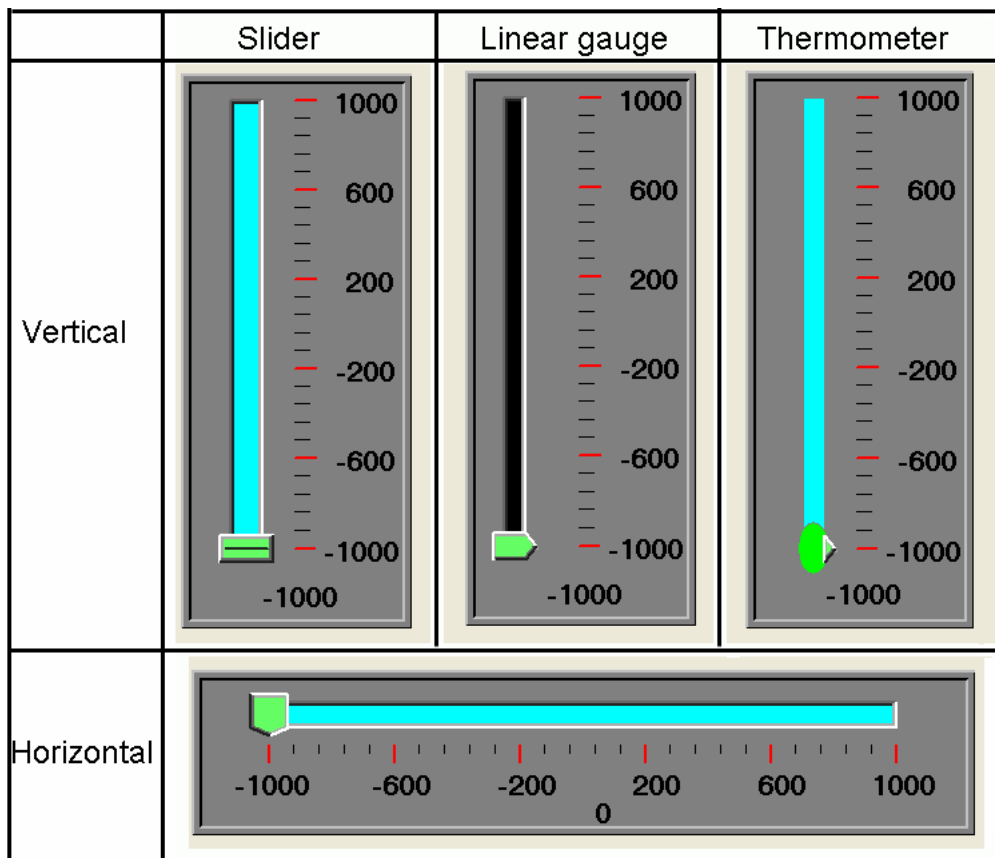
4.3.10 EzHMI EzSlider

4.3.10.1 Description

The EzSlider object offers two modes:

- Displays register values. The slider element will move up or down the scale to show the value.
- Manual input capability by slider dragging (via mouse or touch screen). Each change of the slider position immediately updates the destination register with the slider value.

The EzSlider is available in different graphic looks with a range of configurable parameters in order to be customized according designer's needs. Orientation can be horizontal or vertical, with slider value increasing from bottom to top or from left to right. This multi-purpose control has three different style options: Slider, linear gauge and thermometer.



Setting:

In the following the configuration interface of the EzSlider control will be discussed in more details:

4.3.10.2 Appearances

1. Styles:

- Slider
- Linear Gauge
- Thermometer

2. Frame:

Select between a control with or without a frame.

3. Orientation

- VERTICAL
- HORIZONTAL

4. Tick configuration:

Two different tick types (major and minor ticks) are provided to be placed at regular intervals along the Scale. The control allows the following ticks settings:

- Number of ticks
- Tick color
- Tick minimum and maximum value

5. EzSlider color:

Each component of the control can be assigned a color:

- Indicator or pointer
- Scale track
- Slider background
- Tick label

6. Label text font:

The text font of the tick label labels can be set.

4.3.10.3 Register assignment

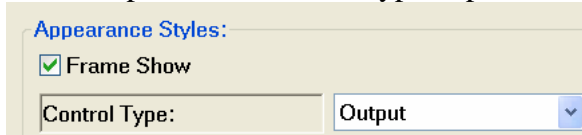
The slider supports two modes:

- If configured to read a source register, the slider will move up or down the scale to the position representing the value.
- If a destination register is selected, that register will be updated with the value of the slider. If the register value has been changed by another control or program code the slider position will be updated to the new value.

4.3.10.3.1 Display source register data

The EzSlider control is able to display data from an AI, AO, D or F register type.

STEP 1: Select “Output” as the control type to put the control into read mode.



STEP 2: Select a register type from the “*Output AI/AO/D/Fno*” combo box
Link the control to the selected register type by entering a number for “*AI/AO/D/Fno --> Slider Value*”.

Specification	Register	Register numbers	
Analog Output	AO	Local AO:	0 ~ 511
Analog Input	AI	Local AI:	0 ~ 511
long integer	D	None Retain:	1 ~ 3599
		Retain:	4096 ~ 7999
Float	F	None Retain:	1 ~ 1899
		Retain:	2048 ~ 3999

STEP 3: Enter a control refresh time interval value for “*Flash Timer 0, 1, 2,..*”. This enables the EzSlider to read the assigned register and update the display to the current reading.

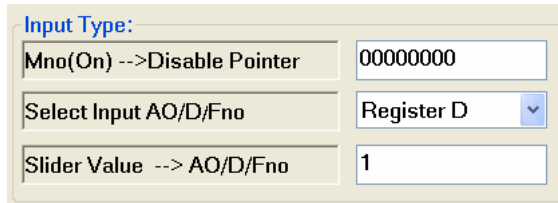
4.3.10.3.2 Change and display destination register data

If the EzSlider control is set to a read/write mode data can be written to or read from AO, D or F register types. By changing with the mouse the slider position the associated register will immediately be updated to the new value. If the register data is being changed by the program or a different control the EzSlider object is automatically updated with the new data.

STEP 1: Select “*Input*” as the control type to put the control into read/write mode.



STEP 2: Select a register type from the “*Select Input AO/D/Fno*” combo box
 Link the control to the selected register type by entering a number for “*Slider Value →AO/D/Fno*”.



STEP 3: Enter a refresh time interval value for the “*Flash Timer 0, 1, 2,..*”. This enables the EzSlider to read the assigned register and update the display to the current reading.

STEP 4: The slide bar can be disabled for user input during runtime. Enter a M register number for “*Mno(On) →Disable Pointer*”. Setting this flag to true during runtime disables the control for user input.

4.3.11 EzHMI EzList

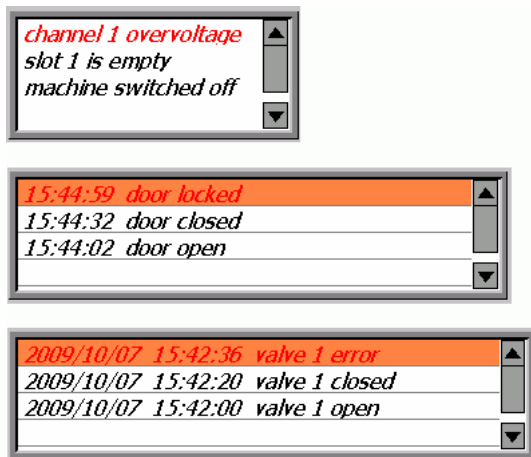
4.3.11.1 Description

The EzList records messages and informs the user during operation. For example information, warning, error and alarm messages are displayed by the list. The list consists of three columns displaying the message together with the date and time of arrival. A list can store up to 256 messages. The newest message is always added on top of the existing list. Older messages will be overwritten when the number of messages exceeds the available list rows. Each list is provided with a vertical scroll bars to scroll through the messages.

The EzList has the following runtime functions:

- Output of messages (maximum 30 Unicode characters)
- Output of the date and time
- Delete EzList contents

The EzList is a passive control which only reads messages from the MSG register and does not allow the user to do any direct editing. The MSG register itself however can be changed for example through the ColorEdit or by using the SET_MSG() function.



Setting:

In the following the configuration interface of the EzList control will be discussed in more details:

4.3.11.2 Appearances

1. Styles:

- Row with/without lines
- Row color
- Text color and size
- Background color
- Number of rows

2. Frame:

Select between a control with or without a frame

3. Time label:

- List with/without date label
- List with/without time label

4.3.11.3 Register assignment

Specification	Register	Register numbers
Flag	M	None Retain: 1 ~ 6999 Retain: 8192 ~ 15999
Message	MSG	Retain: 1 ~ 249

- STEP 1: Assign a source MSG register number for “*MSGno → EzList*”. This register contains the message to be displayed on the list
- STEP 2: A flag is required to trigger an update of the list. Assign “*Mno → Update*” a M register number. As soon as this register changes its status from false to true the text in the MSG register will be added to the top of the list. After the list has been updated the EzList automatically resets the M register status to false.
- STEP 3: Assign “*Mno → Clear All*” an M register number. Setting this register to true will remove all text messages from the list. The EzList automatically resets the M register status back to false.
- STEP 4: Set the display refresh time (Flash Timer 0,1,2...).

Example:

In the following code the message “Hello World” is written to the list. “*Mno → Update*” is assigned to M register 1 and “*MSGno → EzList*” to MSG register 1.

```
//Step 1: Assign the MSG register number 1 the string "Hello World":
SET_MSG(1, L"Hello World");
```

```
//Step 2: Add the string to the top of the list:  
SET_M( 1, true);
```

4.3.12 EzHMI Position

4.3.12.1 Description

The Position control is used for motion control applications. It can be set to display one of the following motion parameters:

- Logic Position (LP)
- Encoder Position (EP)
- Velocity (CV)
- Acceleration (CA)

4.3.12.2 Configuration

The screenshot shows a configuration window with three main sections:

- Motion Card , Axis and Status:** This section contains five rows of controls:
 - CardNo (SlotNo): A dropdown menu set to "Card 1".
 - Axis X/Y/Z/U: A dropdown menu set to "Axis X".
 - Select LP/EP/CV/CA: A dropdown menu set to "LP".
 - Engineering: A text input field containing "000000000001".
 - Decimal Point: A text input field containing "0".
- Register Type and Number Assignment:** This section contains one row:
 - Control Status: A sub-section with a label "Mno --> Hide ActiveX" and a text input field containing "00000000".
- Refresh Interval (Unit 50ms):** This section contains one row:
 - Flash Timer 0,1...: A text input field containing "00000001".

STEP 1: Select the slot number of the motion control card.

STEP 2: Select an axis.

STEP 3: Determine which parameter to read:

- i. Logic Position (LP)
- ii. Encoder Position (EP)
- iii. Velocity (CV)
- iv. Acceleration (CA)

STEP 4: Enter the update rate.

5 EzConfig Utility

5.1 Introduction

EzConfig is a utility to read, set and test the system configuration. The main task of the utility is to detect the IO modules in the PAC slots and map each input/output channel to a register number of the corresponding register type. The programmer can access the IO channels by directly reading or writing to their mapped registers. Four different IO register types (X, Y, AO, AI) are provided: two for storing digital IO values and two for storing analog IO values.

Specification	Register	Register numbers		Data type	Size	Range
Digital Input	X	Local DI:	0 ~ 777	bit	1 bit	true / false
		FRNet DI:	1000 ~ 7777			
Digital Output	Y	Local DO:	0 ~ 777	bit	1 bit	on / off
		FRNet DO:	1000 ~ 7777			
Analog Output	AO	Local AO:	0 ~ 511	float	4 bytes	3.4E +/- 38
Analog Input	AI	Local AI:	0 ~ 511	float	4 bytes	3.4E +/- 38

Table 5: IO register types

The EzCore provides c functions for accessing IO registers:

Specification	Register	Read from Register	Write to Register
Digital Input	X	<code>IN_Xa(X_RegisterNo);</code>	-
Digital Output	Y	<code>GET_Ya(Y_RegisterNo);</code>	<code>OUT_Y(Y_RegisterNo, Flag);</code>
Analog Output	AO	<code>GET_AO(AO_RegisterNo);</code>	<code>OUT_AO(AO_RegisterNo, Value);</code>
Analog Input	AI	<code>IN_AI(AI_RegisterNo);</code>	-

Table 6: APIs for accessing IO registers

Note: It is very important to map all IO channels to IO registers with the EzConfig utility before any other EzProg-I program (or any other program using the EzCore library) can be started. Changing the PAC module setup (e.g. adding new modules to the PAC, plugging existing modules into different slots) requires a new IO mapping by means of the utility.

5.2 Main properties

The utility has got the following main tasks:

- Slot configuration
- Module configuration
- Register mapping
- Reading input values and writing output values

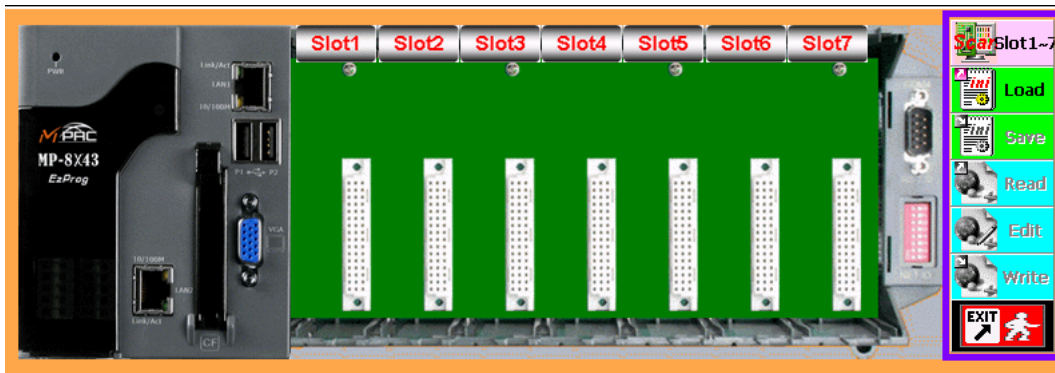


Figure 9: PAC before scanning its slots

Button

Property



Scans every slot of the PAC for IO modules. Once a module has been detected and identified its image will be shown on the corresponding slot on the screen and its channels will be automatically mapped to a register number of the respective IO register.



Reads the previously saved PAC slot setting, module configurations and IO register mapping. The slot settings are displayed by showing the image of the PAC device itself with all the plug-in modules at the correct position.

All the *.NOT* and *.INI* binary files in the EzConfig directory are read.



Saves the PAC slot settings (module names and their slot position), module configurations and IO register mappings to the *.NOT* and *.INI* binary files :

- Device.ini
- noteAI.not
- noteAO.not
- noteB.not
- noteC.not
- noteD.not
- noteDI.not
- noteDO.not
- noteDW.not
- noteF.not
- noteM.not
- noteT.not
- noteW.not

The EzCore engine uses these settings to update the IO registers with the correct input and output values.




Reads the value (“Val”) column and description (“NOTE”) column of the *IO_Table.XML* file. The initial value of each register and the description of the register are loaded.



Opens the user interface for entering the default or initial value and description for each register.

Save the initial value settings and descriptions of all the registers to the *IO_Table.XML* file.



Note: The EzCore engine is responsible for initializing and updating the registers. The EzCore only accesses the configuration set in the *.NOT* and *.INI* binary files and does NOT read the *IO_Table.XML* file. Therefore it is necessary to save the initialization to the *.NOT* and *.INI* binary files by clicking the  button.



Exit the EzConfig utility.

Table 7: EzConfig button description

5.3 Slot scan and IO register mapping

STEP 1: Start the the EzConfig utility on the Windows CE platform:

Start →Program →ICPDAS →EzProg-I →EzConfig

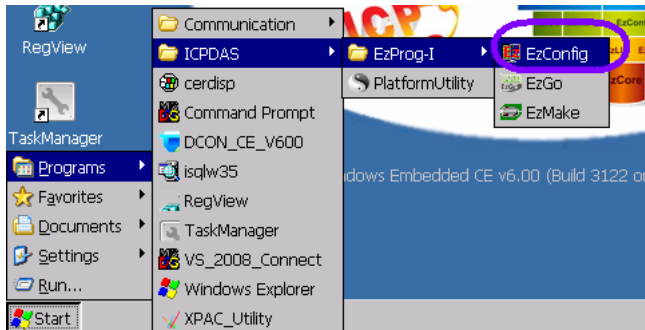


Figure 10: Start EzConfig utility

STEP 2: Click the “Scan Slot 1~7” button and confirm your decision by clicking “Yes” in the message box.

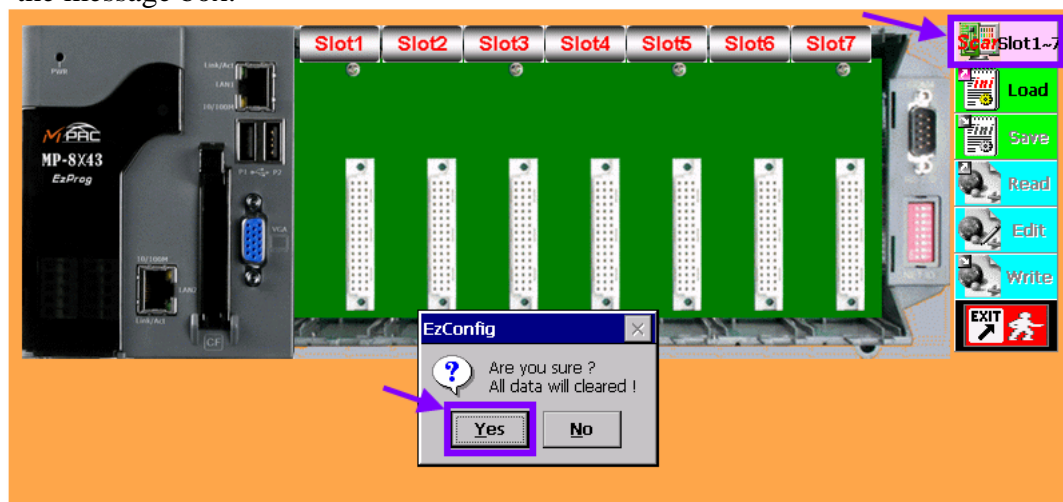


Figure 11: Start scan process

Every slot of the PAC will be scanned. Once a module has been detected and identified it is displayed in the corresponding slot on the EzConfig drawing. At the same time all the IO channels of the modules are mapped to IO register numbers.

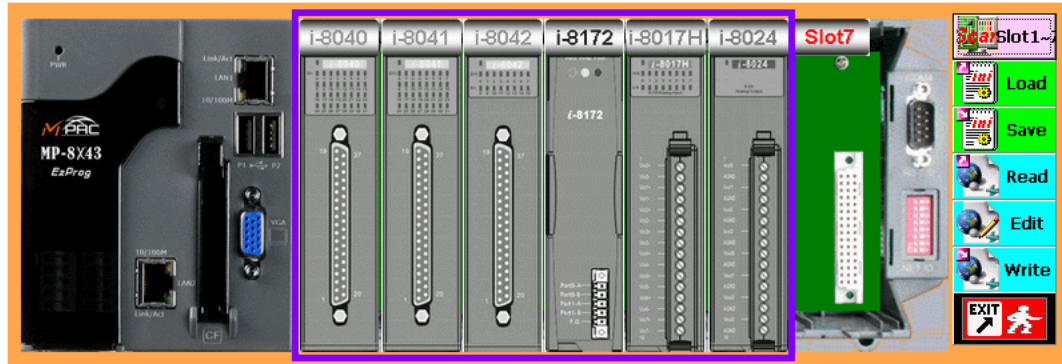


Figure 12: Modules detected during the scan process

STEP 3: The IO register mapping can be determined by clicking on a module. In addition each module can be configured individually. For example the physical unit of the analog input data (e.g. ampere, volt, and temperature) and the data range (e.g. -50 to 50mV, 4 to 20mA) of an analog input module have to be configured.

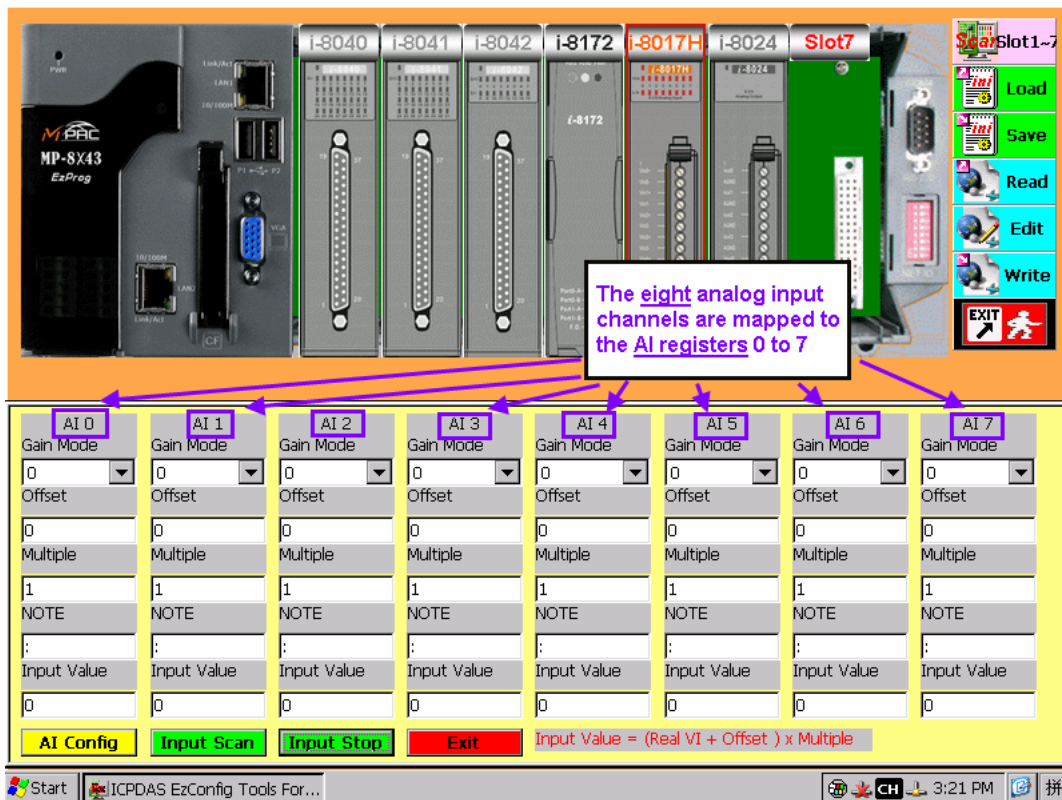


Figure 13: Configuration interface of the i-8017H analog input module

STEP 4: Press “Save” to save the IO register mapping.

STEP 5: Press “Exit” to close the program.



Note: It is necessary to close the EzConfig utility in order for other EzProg-I programs to run smoothly.

5.4 Module and channel configuration

The configuration page has the following purpose:

- i. Channel configuration
- ii. Channel description
- iii. Direct channel access: writing output, reading input

Only Modules which are plugged into the slots can be configured.

- i. Use the Scan  button to detect and display all plug-in modules. After the slots have been scanned it is not allowed to change the slot setting. That means it is not permitted to remove modules from a slot, add new modules or change the modules slot position. This will cause an error. Every change of the slot setting requires a new slot scan.
- ii. Click on the image of a module to open its configuration interface. Make the necessary configuration and confirm the configuration setting by clicking the “OK” button. Now click on the image of the next module which needs to be configured.
- iii. After all the modules have been set save the configuration by clicking the  button.

5.4.1 Digital input configuration

The configuration page of a digital input module displays for each input channel the current status (ON/OFF), the assigned register type and register number. Digital input channels can not and do not need to be configured, but it is possible to add a 30 character comment next to each channel to describe its function in the control system.

Channel status:

- green color represent OFF
- red color represent ON

Example:

Click on the image of the DI module i-8040 in the first slot (see Figure 12). The i-8040 module has got 32 DI channels. The current status of each channel is displayed together with the mapped register type (X) and register number.

Note: The register numbering for the X register type is based on the octal numeral system (base-8 number system).

Octal:	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20	21
--------	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----

Decimal: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

In this example some DI channels LED are provided with a comment to specify their purpose in the control system.

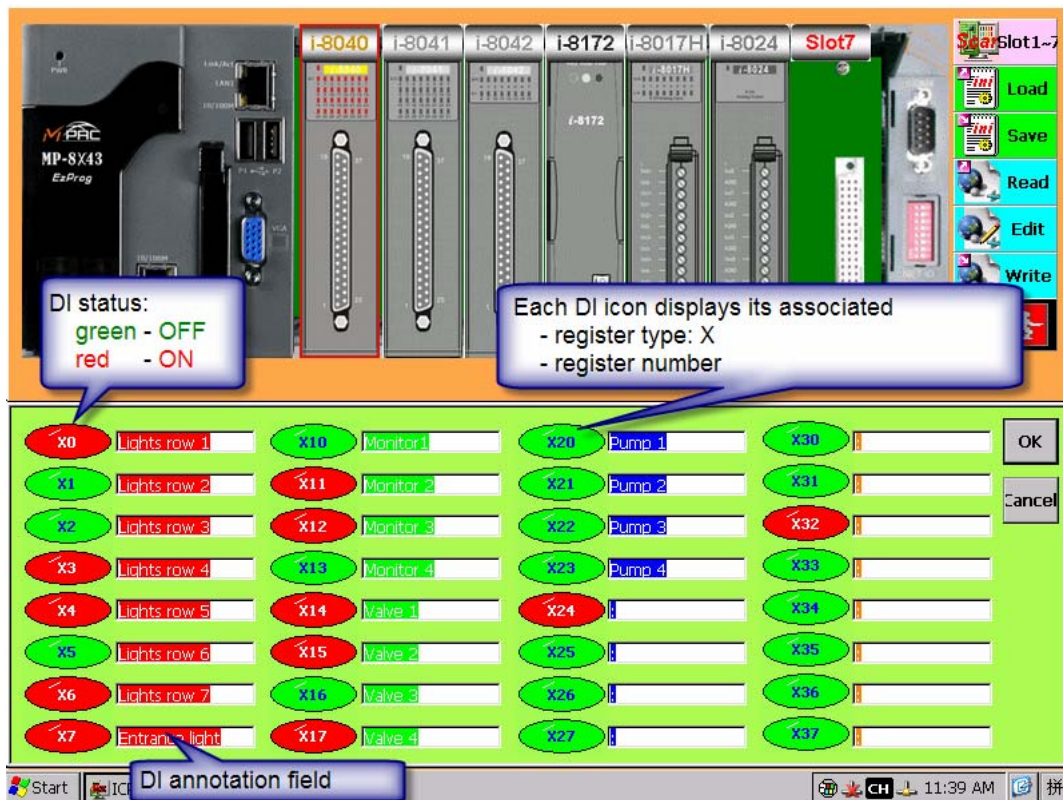


Figure 14: Configuration interface of the i-8040 digital input module

5.4.2 Digital output configuration

The configuration page of a digital output module displays for each output channel the current status (ON/OFF), the assigned register type (Y) and register number. Each channel is represented by a switch which indicates the channel state. The state of a channel can be directly changed by clicking the respective switch. A 30 character long comment can be added to the text box next to each channel to describe its task in the control system.

Channel status:

- green color represent OFF
- red color represent ON

Example:

Click on the image of the DO module i-8041 in the second slot (see Figure 12). The i-8041 module has got 32 DO channels. The current state of each channel is displayed together with the mapped register type (Y) and register number.

Note: The register numbering for the Y register is based on the octal numeral system (base-8 number system).

Octal:	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20	21
Decimal:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

In the following picture some DO channels have a comment to describe their task in the control system.

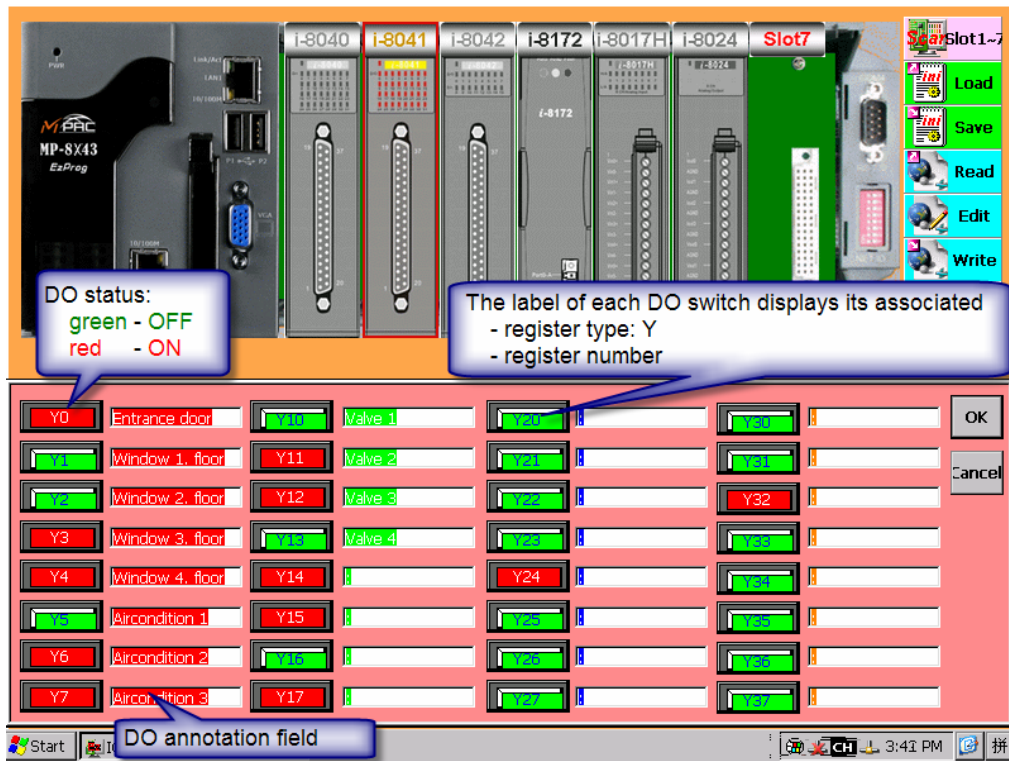


Figure 15: Configuration interface of the i-8041 digital output module

5.4.3 Digital IO configuration

This user interface is a combination of the digital in- and output configuration page. The current status (ON/OFF), the assigned register type (X/Y) and register number of each channel are displayed. Input channels are represented by an oval-shape icon and output channels by a rectangular switch. The state of an output channel can be directly changed by clicking the respective switch. A 30 character long comment can be edited to the text box next to each channel to describe its role in the control system.

Channel status:

- green color represent OFF
- red color represent ON

Example:

Click on the image of the DIO module i-8042 in the third slot (see Figure 12). The i-8042 module has got 16 DI and 16 DO channels. The current state of each channel together with the mapped register type (X or Y) and register number are displayed.

Note: The register numbering for the X and Y registers are based on the octal numeral system (base-8 number system).

Octal:	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20	21
Decimal:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

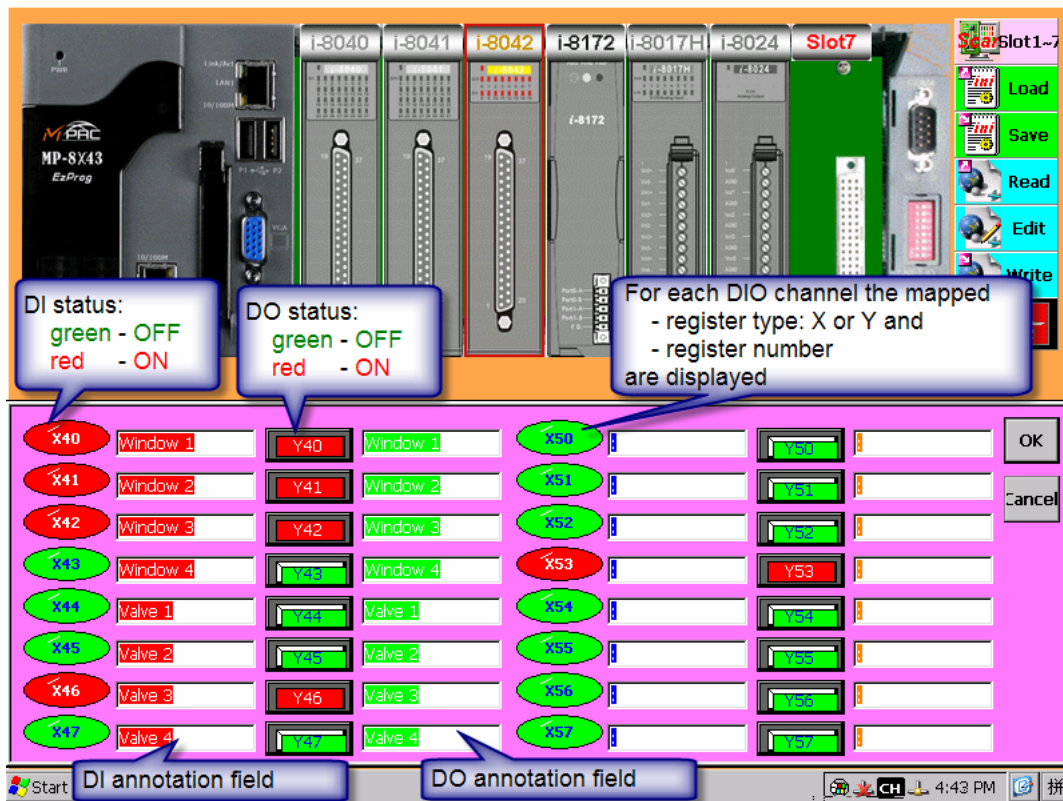


Figure 16: Configuration interface of the i-8042 digital IO module

5.4.4 FRnet configuration

The FRnet master module i-8172 has two ports (port 1 and port 2) and is preconfigured by ICPDAS as follows:

The master can control via one port up to 128 digital output and 128 digital input channels. For each master port the first 8 slave addresses (00 to 07) are reserved for digital output modules and the remaining 8 addresses (08 to 15) are reserved for digital input modules. ICPDAS provides two types of FRnet slaves: digital input devices with 16 channels and digital output devices with 16 channels. Therefore up to 16 ICPDAS FRnet modules can be connected to each master port: a maximum of 8 DI and 8 DO FRnet slaves.

The user interface for the FRnet master module displays the FRnet slaves with their FRnet addresses and their master port. For each FRnet slave a separate user interface can be opened in which the current status (ON/OFF), the assigned register type (X/Y) and register number of each channel are displayed. Input channels are represented by an oval-shape icon and output channels by a rectangular switch. The state of an output channel can be changed by clicking the respective switch. A 30 character long comment can be edited to the text box next to each channel to describe its role in the control system.

Note: The register numbering for the X and Y registers are based on the octal numeral system (base-8 number system). FRnet X and Y register numbers start from 1000 (octal number).

Octal:	1000	1001	1002	1003	1004	1005	1006	1007	1010	1011	1012
Decimal:	512	513	514	515	516	517	518	519	520	521	522

Channel status:

- green color represent OFF
- red color represent ON

Example:

Click on the image of the FRnet module i-8172 in the fourth slot (see Figure 12). The first eight rows represent the FRnet addresses (00 to 07) for the digital output slaves for the FRnet master port 0 and 1. The remaining addresses (08 to 15) are reserved for digital input FRnet modules.

In this example one digital output module (FR-2057T) with the address 00 and one digital input module (FR-2053T) with the address 08 are connected to port 0 of the i-8172 FRnet master module and one digital output module (FR-2057T) with the address 00 to port 1.

STEP 1: Enable the “**Port 0: DO(00) Group**” check box to integrate the digital output module (FR-2057T) with the address 00 into the system.

STEP 2: Enable the “**Port 0: DI(08) Group**” check box to integrate the digital input module (FR-2053T) with the address 08 into the system.

STEP 3: Enable the “**Port 1: DO(00) Group**” check box to integrate the digital output module (FR-2057T) with the address 00 into the system.

STEP 4: Click the “FRnet Config” button to add all three modules to the masters polling list. The FRnet master will now access these modules.

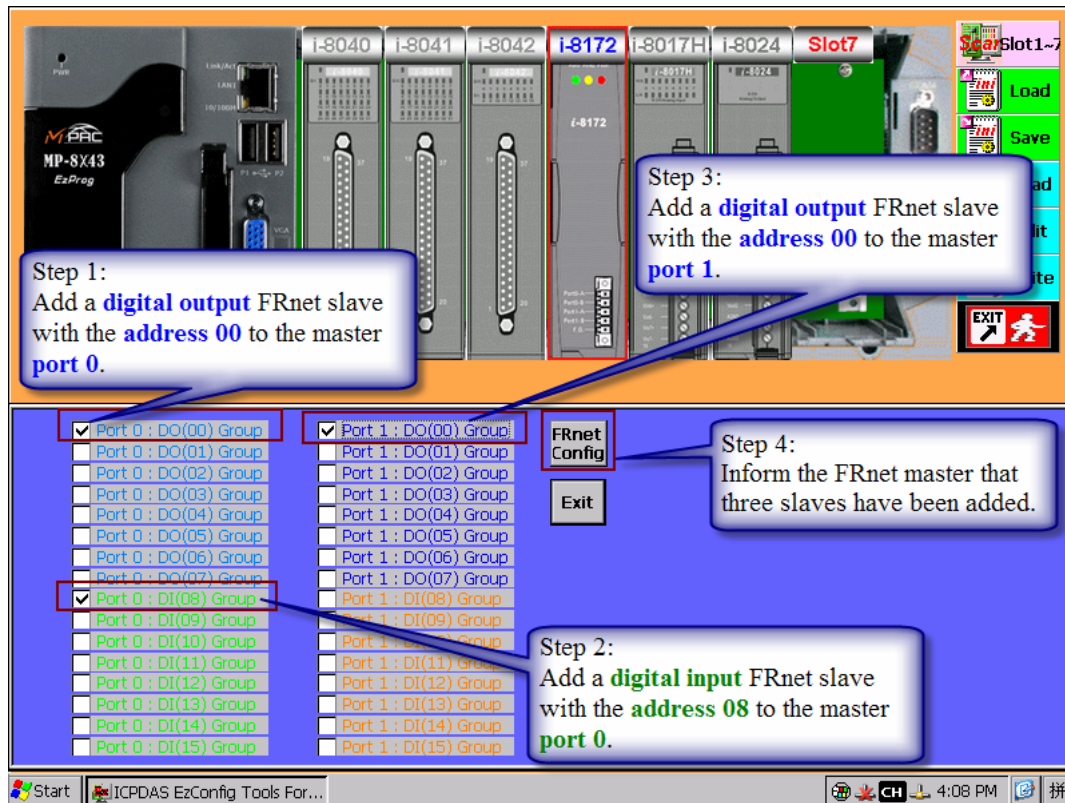


Figure 17: Configuration interface of the i-8172 FRnet master module

STEP 5: To open the digital IO user interface for a slave FRnet module click the red button next to the slave.

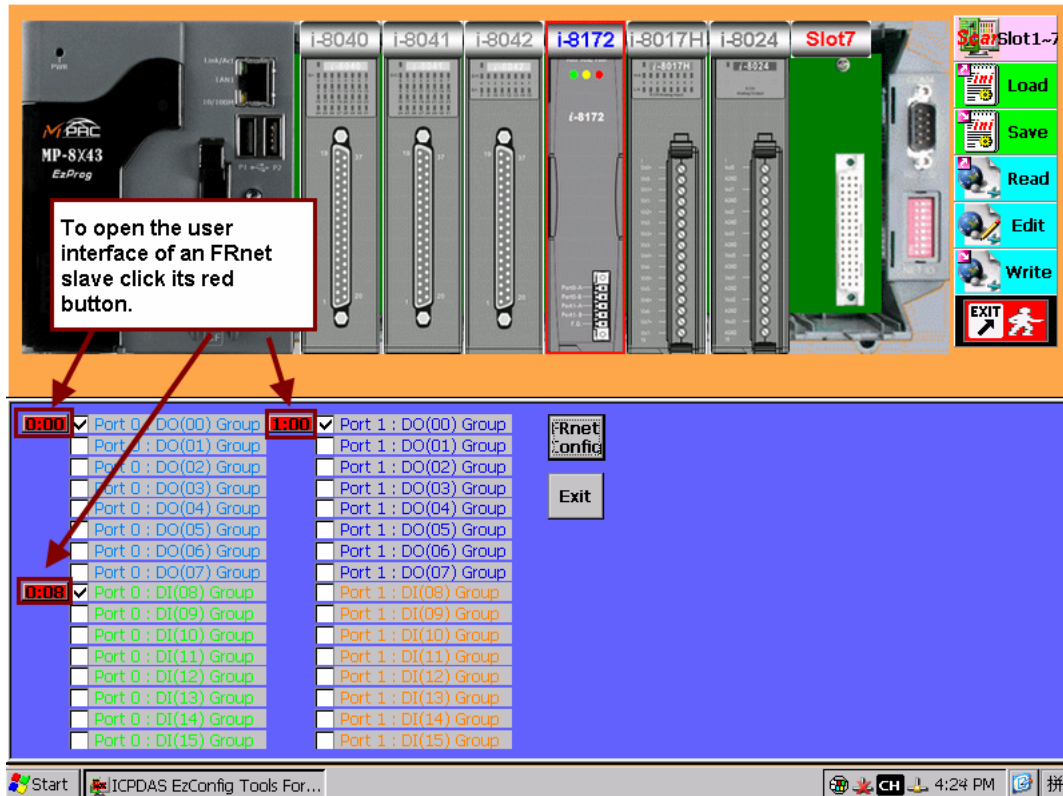


Figure 18: FRnet master configuration interface

STEP 6: Click the “0:00” button to open the user interface of the digital output slave (address 00) of port 0. The state and the Y register number of each channel are displayed.

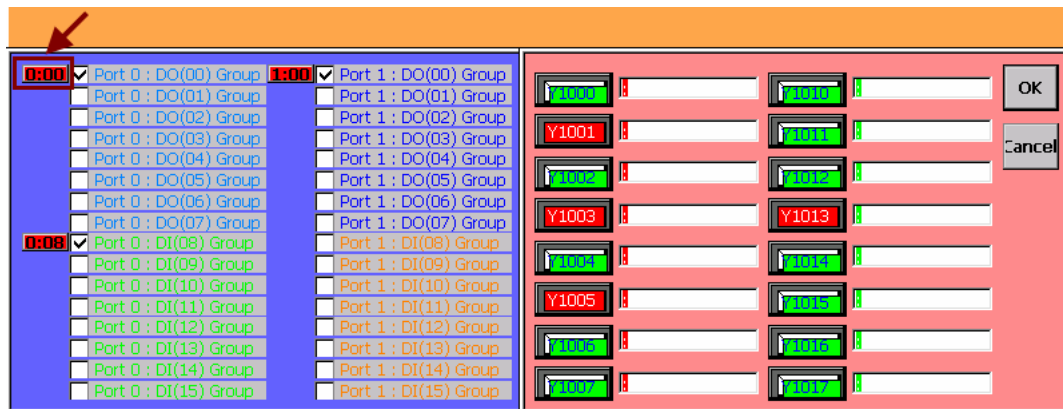


Figure 19: FRnet slave interface

Note:

The register number mapping may change if new FRnet slaves are added to the control system.

5.4.5 Analog input configuration

For each channel the analog input type (engineering unit and range), offset and gain can be set. The configuration page displays the mapped register type (AI) and register numbers. The polling of the input channels starts as soon as the scanning process has been activated (“Input Scan”). A 30 character long comment can be edited to the text box next to each channel to describe its purpose.

The screenshot shows the configuration page for AI 4. The fields and their values are: AI 4, Gain Mode, 0: +/-10V, Offset: -0.5, Multiple: 0.5, NOTE: test, and Input Value: 1.50049. Red arrows point from callout boxes to each of these fields.

- Each analog input channel is automatically mapped to a register number of the register type AI. The number can not be changed by the user.
- Select an engineering unit (ampere, voltage) and range for the input value.
- Analog input **offset**.
- Analog input **gain**.
- Annotation field
- Actual input value = (measured value + Offset) x Multiple

5.4.5.1 Offset and Gain

Offset and gain commands are used for calibration. By setting offset and gain, you make sure that values read from a field device are more accurate.

Offset

Offset is the difference between the minimum analog input value read and the actual minimum analog value.

$$\text{Actual Value} = \text{Reading} + \text{Offset}$$

Example:

The module is set to measure a range 4–20 mA values. If the actual input signal is 4 mA and the module reads a value of 4.006 mA then the offset is 0.006 mA. The offset represents the difference between the two minimum values.

Gain

Gain is the ratio of the full-scale reading to the maximum input.

$$\text{Actual Value} = \text{Measured Value} * \text{Gain}$$

Offset must be calculated first; then gain is calculated.

Example:

The module is set to measure a range 4–20 mA values.

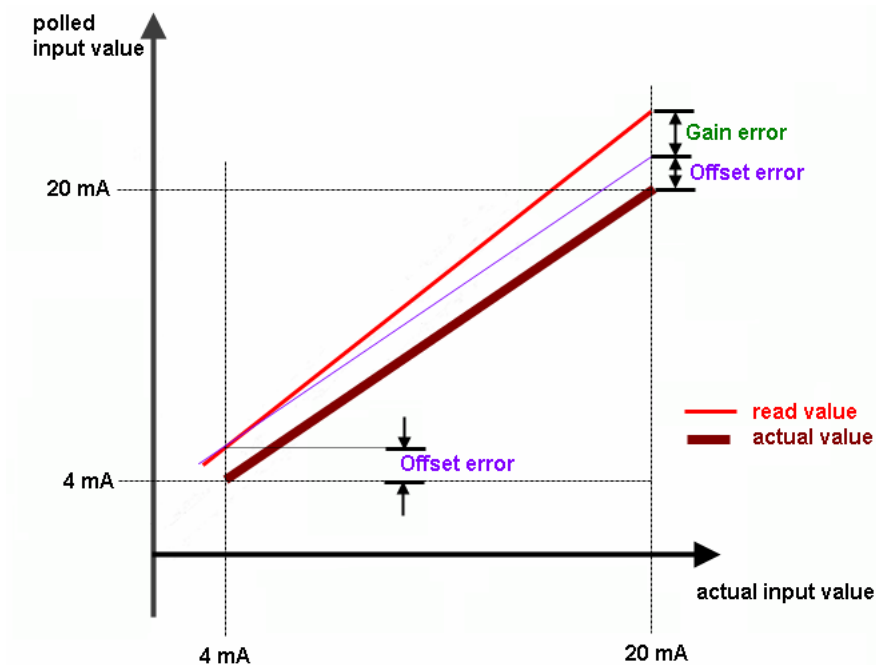
If maximum input = 20.00 mA

and measured value = 20.40 mA

then **gain** = actual value/ measured value = 20/20.40 = 0.980392

The formula which takes the offset and gain into account:

$$\text{Actual Value} = (\text{Reading} + \text{Offset}) * \text{Gain}$$



Example:

Click on the image of the analog input module i-8017 in the fifth slot (see Figure 12). The i-8017 module has got eight analog input channels. The configuration page shows the mapped register type and register numbers (AI 0 to AI 7).

STEP 1: Select the required gain mode (engineering unit and range) of the input signal.

STEP 2: Calibrate all channels by entering the correct offset and gain (multiple) values.

STEP 3: Save the calibration values by clicking the “AI Config” button.

STEP 4: Click “Input Scan” to poll the channels.

AI 0	AI 1	AI 2	AI 3	AI 4	AI 5	AI 6	AI 7
Gain Mode	Gain Mode	Gain Mode	Gain Mode	Gain Mode	Gain Mode	Gain Mode	Gain Mode
4: +/-20mA	4: +/-20mA	4: +/-20mA	2: +/-2.5V	2: +/-2.5V	2: +/-2.5V	0: +/-10V	0: +/-10V
Offset	Offset	Offset	Offset	Offset	Offset	Offset	Offset
0.2	0.8	0.7	0	0	0	0	0
Multiple	Multiple	Multiple	Multiple	Multiple	Multiple	Multiple	Multiple
1.6	0.7	1.01	1	1	1	1	1
NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE
flow rate	weight	temperature	temperature	:	:	:	:
Input Value	Input Value	Input Value	Input Value	Input Value	Input Value	Input Value	Input Value
0.322441	0.560214	0.703609	0.00396729	0.00915527	0.00152588	0.000305176	-0.00488281
AI Config	Input Scan	Input Stop	Exit	Input Value = (Real VI + Offset) x Multiple			

Figure 20: Configuration interface of the i-8017H analog input module

5.4.6 Analog output configuration

For each analog output channel the output type (physical unit and range), offset and gain can be set. The register type (AO) and register numbers of each channel are displayed on the configuration interface. Further more the user can directly output analog values by using the configuration interface. A 30 character long description can be added to every channel to specify its purpose in the control system.

1. Each analog output channel is automatically mapped to a register number of the register type AO. The register number allocation is done by the EzConfig utility and can not be changed by the user.

AO 0	Offset	Multiple	Output
0:+/-10V	0.5	2	1.5
test			

2. Select a physical unit (ampere, voltage) and range for the output value.

AO 0	Offset	Multiple	Output
0:+/-10V	0.5	2	1.5
test			

3. Offset value for analog output signal

AO 0	Offset	Multiple	Output
0:+/-10V	0.5	2	1.5
test			

4. Gain value for analog output signal

AO 0	Offset	Multiple	Output
0:+/-10V	0.5	2	1.5
test			

5. Default analog output value.

AO 0	Offset	Multiple	Output
0:+/-10V	0.5	2	1.5
test			

6. Annotation field.

AO 0	Offset	Multiple	Output
0:+/-10V	0.5	2	1.5
test			

5.4.6.1 Offset and Gain for analog output

Offset and gain commands are used for calibration. By setting offset and gain, you make sure that values sent to a field device are more accurate.

Offset

Offset is the difference between the *minimum* analog output value sent by the controlling software and the actual analog value.

$$\text{Actual Value} = \text{Sending} + \text{Offset}$$

Example:

The module is set to a 0–20 mA range. If the controlling program sends 0 mA and the channel actual output value is 0.003 mA then the offset is 0.003 mA. The offset represents the difference between the two minimum values.

Gain

Gain is the ratio of the actual output and the *maximum* value of the range sent.

$$\text{Actual Value} = \text{Value Sent} * \text{Gain}$$

Offset must be calculated first; then gain is calculated.

Example:

The module is set to a output range of 4–20 mA.

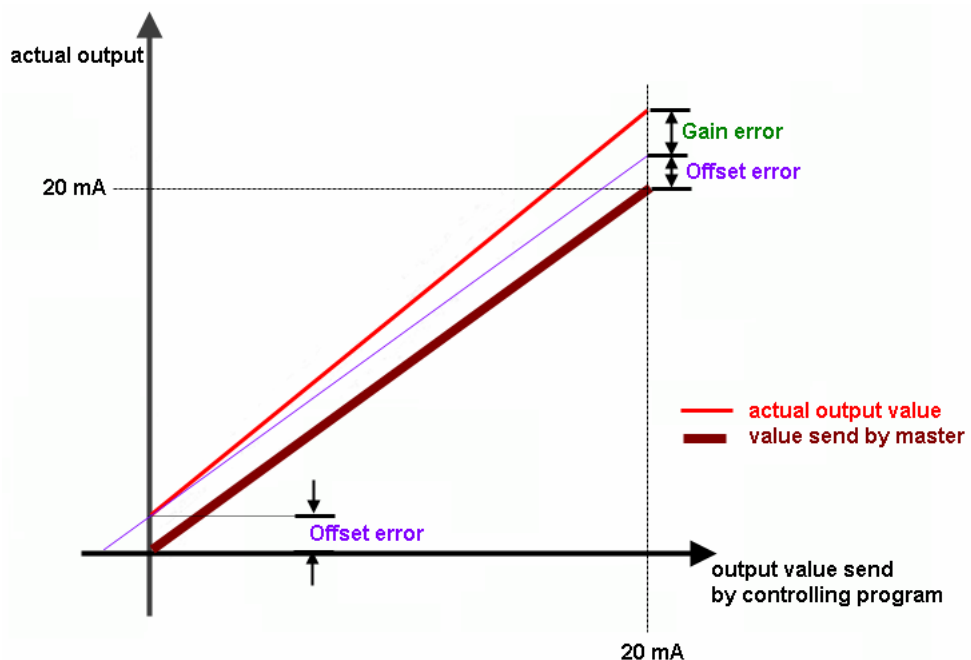
If maximum output sent = 20.00 mA

and actual value = 20.70 mA

then **gain** = actual value / value sent = 1.035

The formula which takes the offset and gain into account:

$$\text{Actual Value} = \text{Value sent} * \text{Gain} + \text{Offset}$$



Example:

Click on the image of the analog output module i-8024 in the sixth slot (see Figure 12). The i-8024 module has got four analog input channels. The configuration page shows the mapped register type and register numbers (AO 0 to AO 3).

STEP 1: Select a suitable gain mode (engineering unit and range) for the analog output channels.

STEP 2: Calibrate all channels by entering the correct offset and gain (multiple) values.

STEP 3: Save the calibration values by clicking the “AO Config” button.

STEP 4: Test the configuration: Enter output values in the output field and click the “Output” button to directly send these values to the corresponding output channels of the i-8024 module.

For each analog output channel the mapped
- register type (AO) and
- register number
are displayed.

AO	Offset	Multiple	Output
AO 0	0	1.004	6.5
boiler temperature			
AO 1	-0.005	1.0007	8.0
outlet flow rate			
AO 2	0.0001	1.001	9.5
inlet flow rate			
AO 3	0	1	0

Real VO = (Output x Multiple) + Offset

AO Config (Yellow button): Enter the channel configurations.

Output (Green button): Send the output values directly to the channels.

Exit (Red button): Exit the configuration page.

Figure 21: Configuration interface of the i-8024 analog input module

5.5 Default startup settings

Non-IO related registers, such as the M, D, F, W, C, T, B, DW and MSG register, can be set to a default value at program start. Furthermore a 30 character long comment can be added to each register to clarify its purpose in the controlling program.

STEP 1: Click the “Edit” button to open the non-IO related register configuration page.

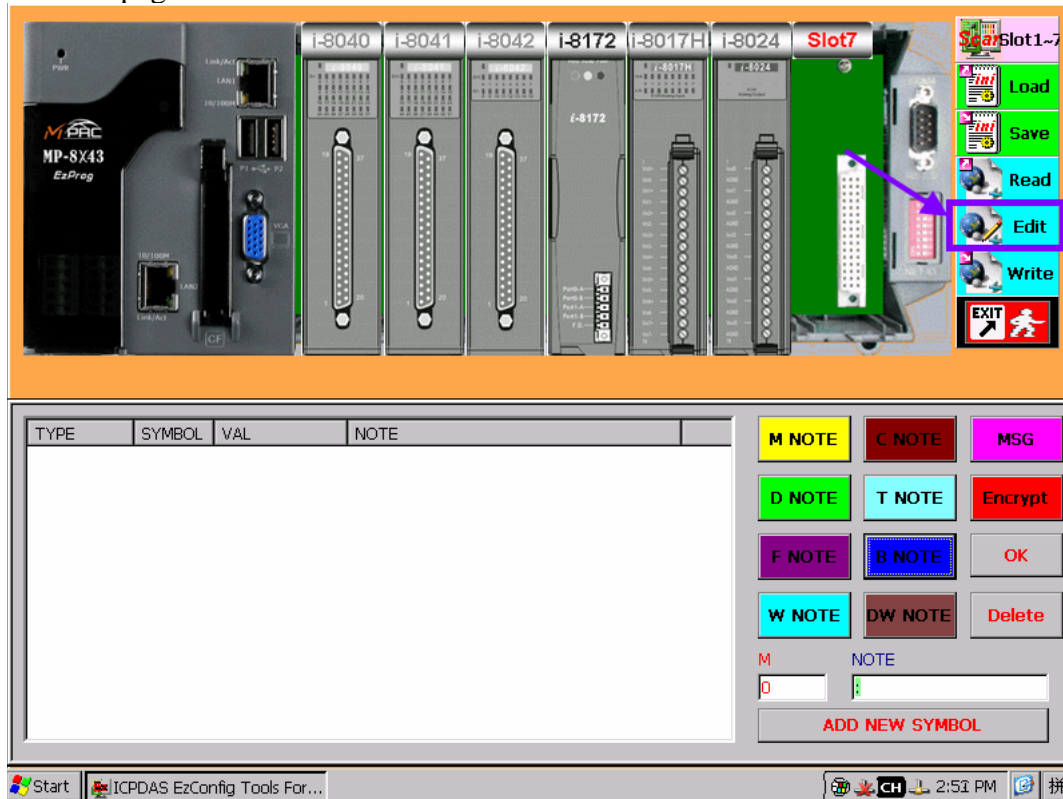


Figure 22: Non-IO register settings

STEP 2: Select a register type which register numbers you would like to initialize or comment.

In the following the procedure of assigning a default value is demonstrated by using an M register type. The same procedure applies to the other register types.

STEP 3: Click the “M NOTE” button to open the M register list. The “M” symbol at the bottom of the dialog window indicates that the list on the left hand sight represents the M register list.

STEP 4: Enter a M register number which has to be initialized with a default value at program startup. Enter a comment in the NOTE field (for example: “My first annotation”).

STEP 5: Add the register to the M list by clicking the “ADD NEW SYMBOL” button.

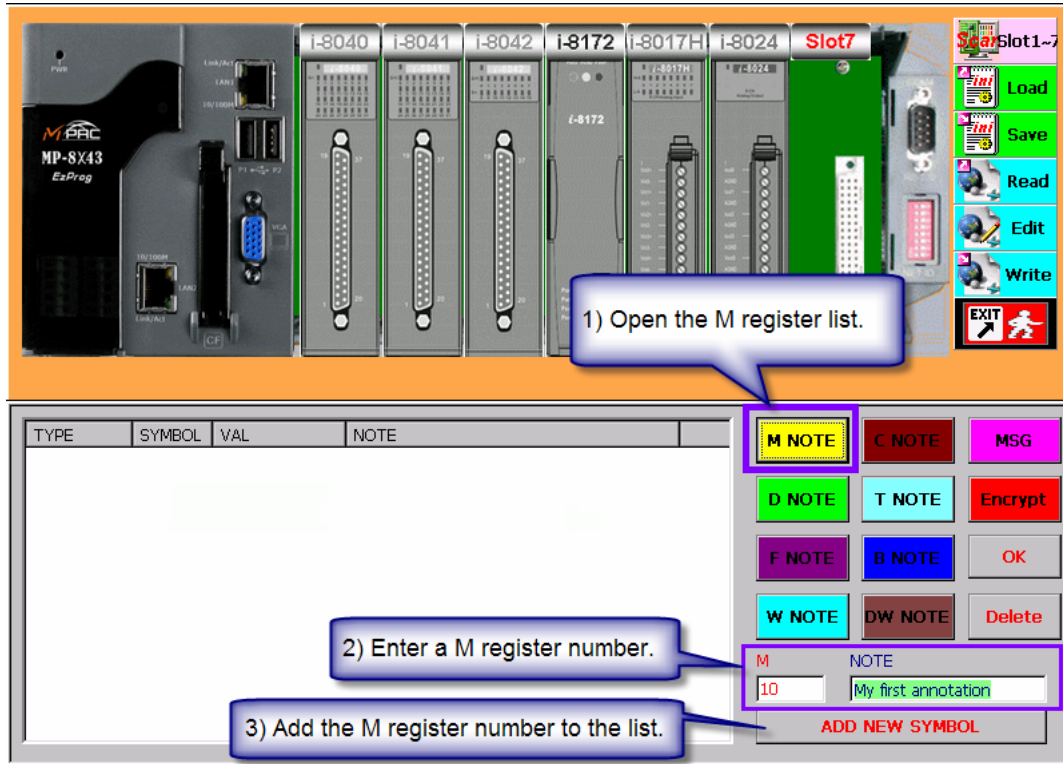


Figure 23: Add a register to the register list

STEP 6: Double click the value field in the VAL column to set the initial value.

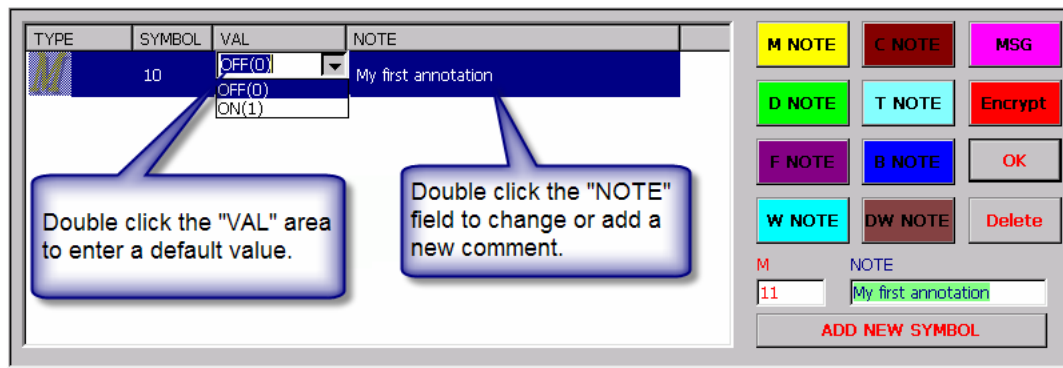


Figure 24: Enter a default value and comment to a register

STEP 7: Add additional registers to the list by repeating STEP 4 to 6. To remove a row from the list, click the “Delete” button.

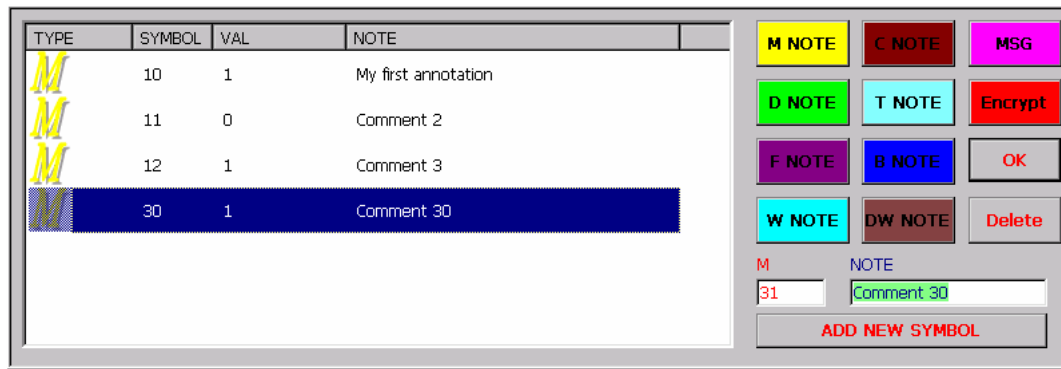


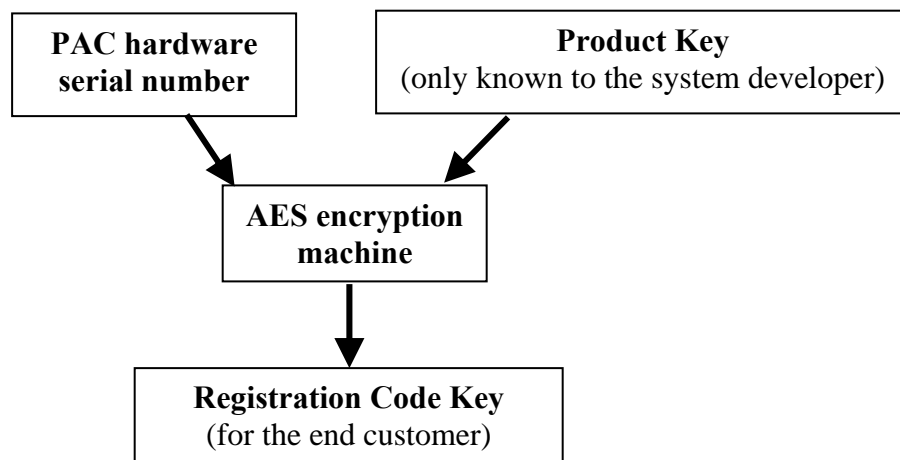
Figure 25: M register default value list

STEP 8: Close the register editor by clicking the “OK” button. Click “Save” to save all settings.

5.6 Registry Key Editor

The registry key editor allows the creation, deletion and updating of device registry settings. This tool is useful for protecting 3rd party applications running on the EzCore engine.

To generate a registration key for the end user the program developer has to specify a random 16 character product key. The encryption engine generates a registration key by using the hardware serial number of the PAC and the product key. The product key is only known to the system developer and should **not** be disclosed.



The **CHECK_KEY** API requires the programmer to enter a product key and then it generates from the registration code key entered by user and the hardware serial number a product key and compares it to the real product key. The API returns a true when they match.

STEP 1: Click the “Edit” button to open the non-IO related register configuration page.

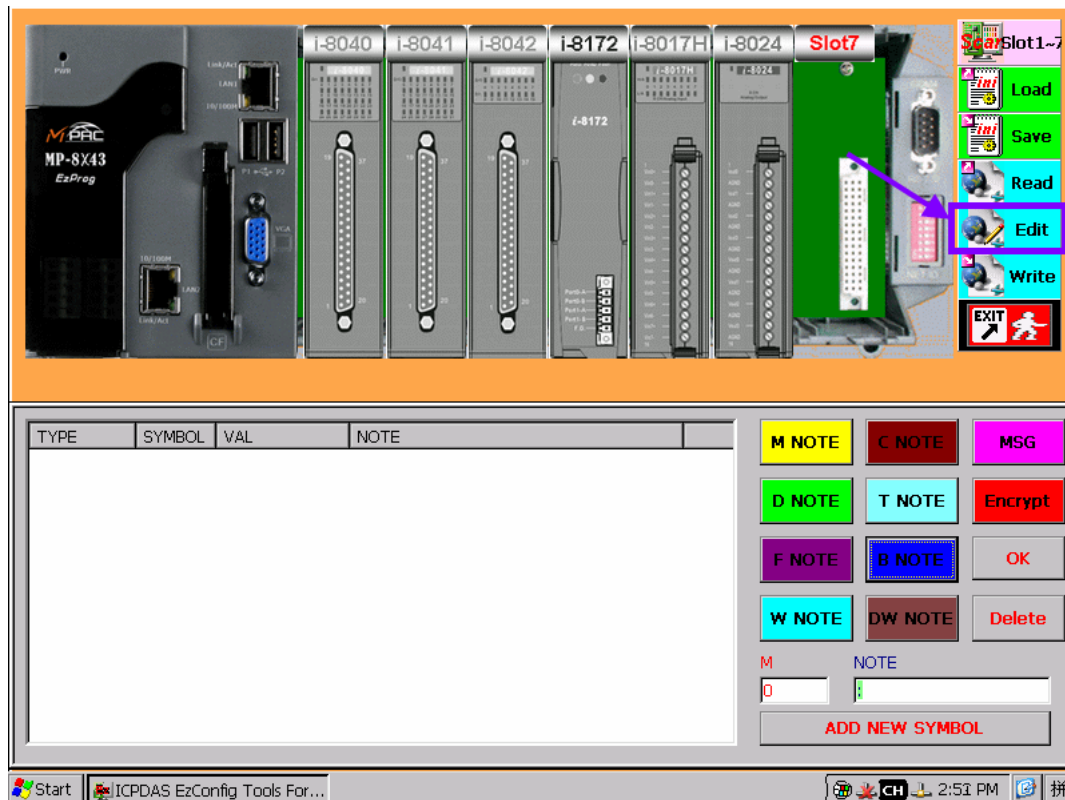


Figure 26: Non-IO related register configuration interface

STEP 2: Click the “Encrypt” button.

STEP 3: Enter your product key. This key must be 16 characters long. You can choose any character sequence.

STEP 4: Click the “Get SN” button. The PAC hardware serial number will be displayed in the “Serial Number” text box.

STEP 5: Generate the registration code key by clicking “OK”. This key will be displayed in the “Registry-code Generator”.

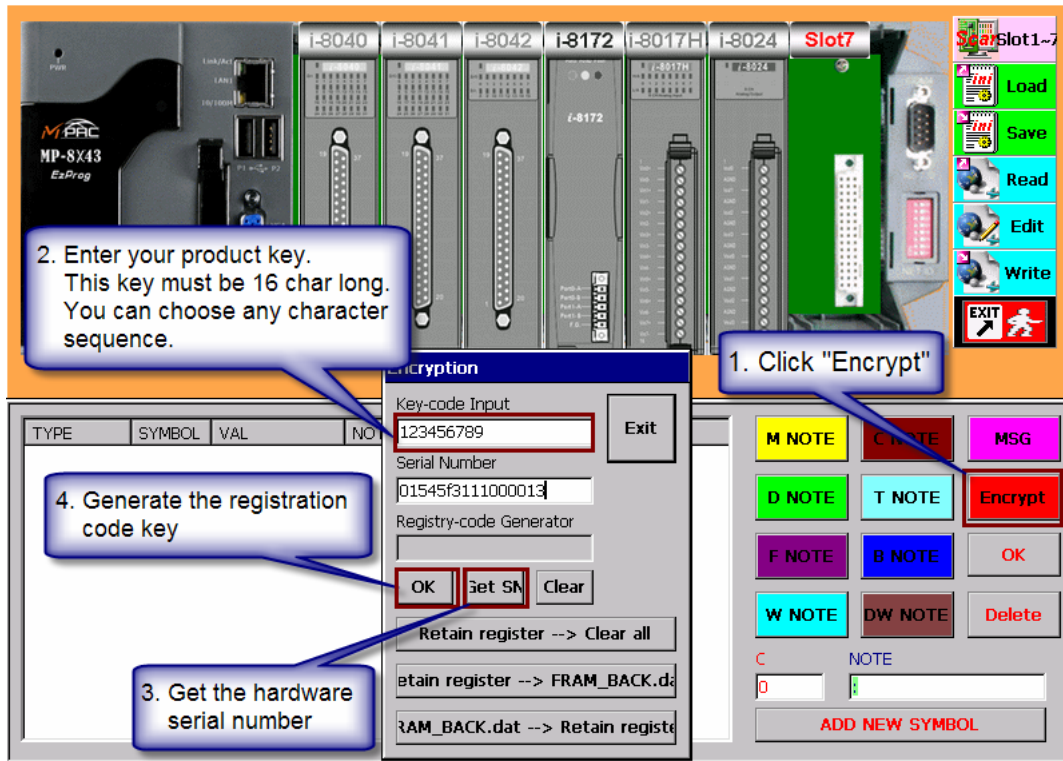


Figure 27: Encryption dialog

STEP 6: Exit the Encryption editor by clicking the "Exit" button. Click "Save" to save all settings.

6 EzGo

6.1 Introduction

The EzGo is a test utility program written for the Windows CE platform which allows the user to test the basic operation of the motion control card and detect any malfunction in the motion control system. EzGo assists the system developer in identifying any wiring problem in the motion control system and helps the user to configure the system before starting to write the actual control program. Parameter settings can be saved to a configuration file.

At the present the EzGo utility supports the following motion control modules:

- i8092
- i8092F
- i8094
- i8094F

The i8094A and i8094AH modules can only be tested but the configuration can not be saved to a file.

6.2 Using EzGo

The EzGo utility consists of four main windows:

1. Main page or selection page
The main page appears after launching EzGo. It allows the user either to open the “Configuration”, “Basic_Operation” or “Advance_Features” window. At program start the user is restricted to open only the “Configuration” window to first do the necessary parameter settings.
2. Configuration page
This page enables the user to do the necessary hardware setting of the motion control card in order for the card to be able to communicate with the servo motor. The settings can be saved to a configuration file or previously saved configuration can be loaded.
3. Basic operation page

This page is for executing independent axis motion commands. A motor representing an axis can be selected to execute a sequence of motion commands regardless of other axes.

4. Advanced features page

This window assists you in testing different interpolation modes like 2/3 axes linear interpolation and any 2 axes circular interpolation.

6.2.1 Selection window



Figure 28: Selection window

The selection window appears directly after program start. During program launch all the slots of the PAC are scanned for any motion cards. In order for the scan process to be successful make sure that no other program is running on the PAC which accesses modules in the slots. The window provides the user with three options:

- Configuration: hardware configuration of the motion control card
- Basic Operation: Single axis testing and point to point motion
- Advanced Features: Multi-axis interpolation testing, 2/3 axes linear interpolation and any 2 axes circular interpolation.

From the selection page the user has to choose the required type of operation. Right after program start only the configuration page can be selected. The motion control card first needs to be initialized and configured before the other operation are supported. Once the motion control card has been successfully initialized you can advance to select the windows for single axis or multi axis motion.

6.2.2 Initialization and configuration window

The configuration page is for setting the hardware signal configuration. It is mainly for setting the

- Pulse output type
- Pulse input type
- Hardware limit signals
- In-position input signal, alarm input signal
- Logical level of the input signal

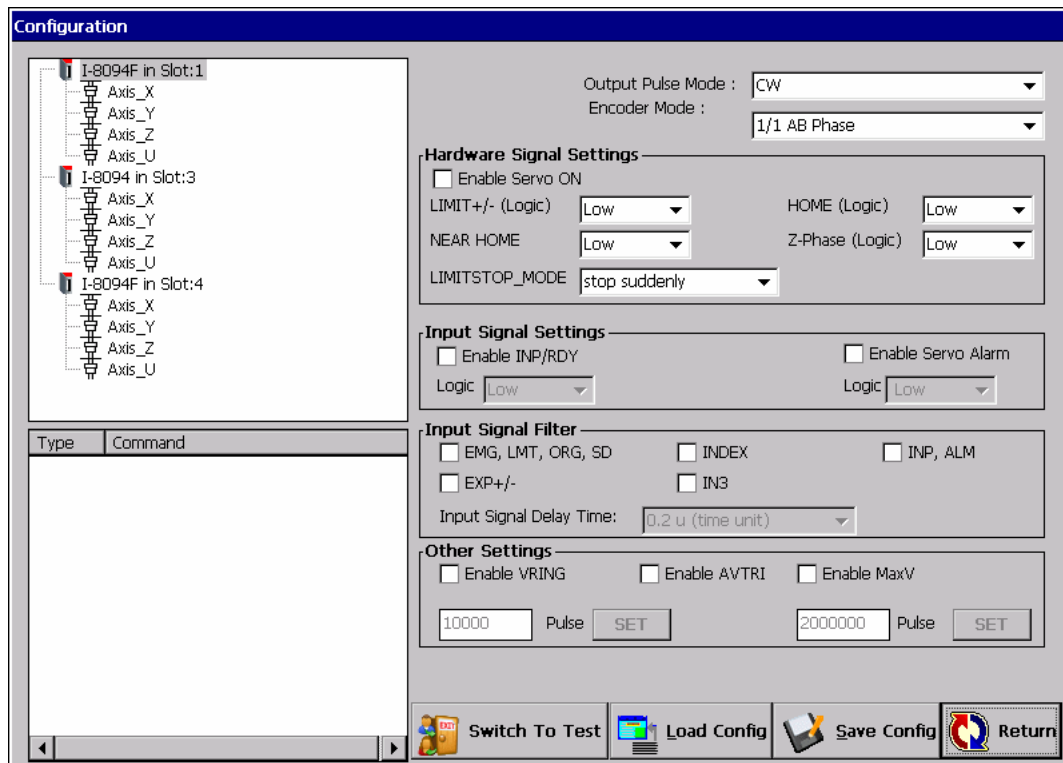


Figure 29: Configuration window

6.2.2.1 Parameter settings

The tree view lists every motion card detected in the slots with its respective name, slot number and number of axis. The utility provides two methods to configure a motion card:

- Configuring all the axis of a motion module at once. In the tree view click the name of the module. All the settings done will now apply to all axis of the selected motion module.
- Configuring each axis separately. To set the axis of a motion module card individually just double click the name of the module to extend the tree view

and select from the sub items an axis to open its configuration page. The current configuration page always belongs to the axis which name is marked by a blue background color in the tree view.

Therefore, if the module name is selected and highlighted by a blue background color the setting done in the configuration page applies to all axis of the module. If the name of an axis is highlighted the settings is only valid for this axis.

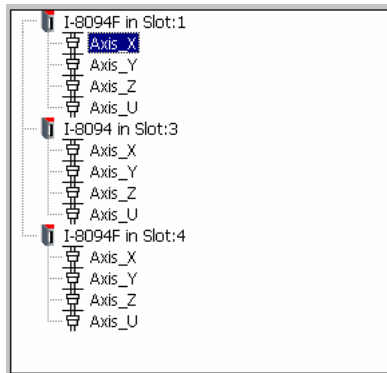


Figure 30: Detected motion cards

The setting of the parameters displayed in Figure 31 is essential.

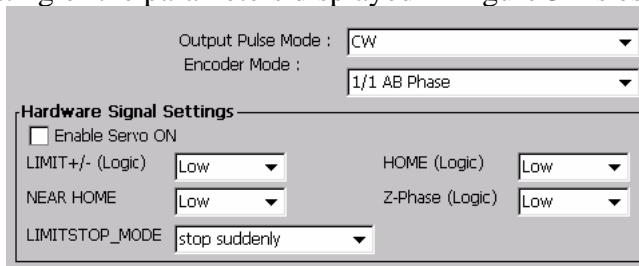


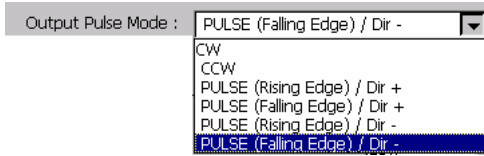
Figure 31: Required configuration

Select from the combo boxes the necessary options. The remaining settings of the configuration page are optional.

You can also load a previous configuration from the configuration file.

6.2.2.1.1 Pulse output / input type selection

Description



Pulse output mode setting:

This function sets the pulse output mode as either CW/CCW or PULSE/DIR for the assigned axes and their direction definition.

The related function is

```
SET_PULSE_MODE(BYTE cardNo, WORD axis, BYTE nMode);
```

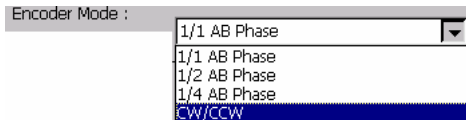
Encoder related parameters

This function sets the encoder input related parameters.

A/B quadrature pulse input mode

When A/B quadrature is selected, the position counter will count up if phase A leads Phase B; the position counter will count down if phase B leads phase A:

- **1/1 AB phase:**
Only the rising edge of A phase is counted.
- **1/2 AB phase:**
The rising and falling edges of A phase are counted.
- **1/4 AB phase:**
Both the rising and falling edges of A phase and B phase are counted.



Up/down pulse input mode

- **Up/ down input**

The counter counts at the rising edge of the positive pulse. The A phase represents the count up input and B phase the count down input.

The related function is

```
SET_ENCODER(BYTE cardNo, WORD axis, BYTE nMode, BYTE nDivision, BYTE nZEdge);
```

6.2.2.1.2 Hardware signal setting

Description

Enable Servo ON

Setting the Servo Driver (ON/OFF)

This function outputs a DO signal (ENABLE) to enable the motor driver.

Active level of the hardware limit switches

This function sets the active logic level of the inputs of the hardware limit switches.

– **Low:**

set the active level for the forward and reverse limit switch to low.

– **High:**

set the trigger signal for the forward and reverse limit switch to high level.

LIMIT+/- (Logic)

The related function is

```
SET_HLMT(BYTE cardNo, WORD axis, BYTE nFLEdge, BYTE nRLEdge);
```

Trigger level of the NHOME sensor

This function sets the trigger level of the near home sensor (NHOME).

Active level setting for the near home sensor:

– **Low = low active (0);**

– **High = high active (1);**

NEAR HOME

The related function is

```
SET_NHOME(BYTE cardNo, WORD axis, BYTE nNHEdge);
```

Motion stop method

This function sets the motion stop mode of the axes when the corresponding limit switches are detected.

– **Stop suddenly:** Axis stops immediately when any limit switch on the axis is triggered;

– **Stop after deceleration:** The axis decelerates to

LIMITSTOP_MODE

standstill when any limit switch on the axis is triggered.

The related function is

```
LIMITSTOP_MODE (BYTE cardNo, WORD axis, BYTE nMode);
```

Trigger level of the home sensor

This function sets the trigger level of the home sensor (HOME).

Active level setting for the home sensor:

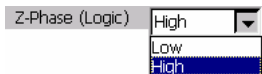
- Low = low active (0);
- High = high active (1);



Z phase trigger level

Sets the trigger level for the Z phase

- Low = low active (0);
- High = high active (1);



The related function is

```
SET_ENCODER(BYTE cardNo, WORD axis, BYTE nMode, BYTE nDivision, BYTE nZEdge);
```

6.2.2.1.3 Input signal settings

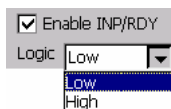


Description

In-position signals

This function sets the active level of the in-position input signals (INPOS input signal).

- Enable INPOS input signal
 - 0 = disable INPOS input;
 - 1 = enable INPOS input



- Set the trigger level
 - Low = low active (0);
 - High = high active (1);

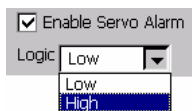
The related function is

```
SET_INPOS(BYTE cardNo, WORD axis, BYTE nMode, BYTE nIEdge);
```

Servo alarm

This function sets the parameters of the ALARM input signal.

- Enable INP/RDY
 - 0 = disable ALARM function;
 - 1 = enable ALARM function



- Sets the trigger level:
 - Low = low active (0);
 - High = high active (1);

The related function is

```
SET_ALARM(BYTE cardNo, WORD axis, BYTE nMode, BYTE nAEdge);
```

6.2.2.1.4 Input signal filter

Input Signal Filter

EMG, LMT, ORG, SD
 INDEX
 INP, ALM

EXP+/-
 IN3

Input Signal Delay Time:

These checkboxes enables you to select the input signals and set the delay time of the filter.

Select input signal:

	FEn Code	Input signals	Description
<input type="checkbox"/>	1	EMG	Emergency
<input type="checkbox"/>		LMT	Hardware limit switch
			– positiv direction limit signal (nLMTTP)
			– negative direction limit signal (nLMTM)
<input type="checkbox"/>		ORG	Home signal (nIN1)
		SD	Slow Down signal Near Home signal (nIN0)
<input type="checkbox"/>	2	INDEX	index of Encoder input (Z phase) (nIN2)
	4	INP	servo in-position input signal (nINPOS)
<input type="checkbox"/>		ALM	Alarm input signal (nALARM)
<input type="checkbox"/>	8	EXP+/-	External pulse (nEXPP, nEXPM, EXPLSN)
<input type="checkbox"/>	16	IN3	nIN3

The sum of the **FEn** code numbers (0~31) are used to set the time constant for filtering the input signals.

Example:

Input Signal Filter

EMG, LMT, ORG, SD
 INDEX
 INP, ALM
 EXP+/-
 IN3

Input Signal Delay Time: 0.2 u (time unit) ▼

Selected options are:

Input signals	FEn Code
EMG, LMT, ORG, SD	1
INDEX	2
INP, ALM	4
Σ	FEn = 7

SET_FILTER(BYTE cardNo, WORD axis, 7, WORD FLn)

Select delay time:

Input signal delay time

This function sets the time constant for digital filters of the input signals:

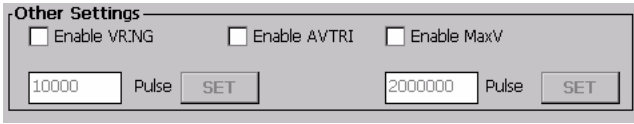
Input signal delay time	FLn code
2μ SEC	0
256μ SEC	1
512μ SEC	2
1.024mSEC	3
2.048mSEC	4
4.096mSEC	5
8.192mSEC	6
16.384mSEC	7

Input Signal Delay Time: 0.2 u (time unit) ▼

The related function is

SET_FILTER(BYTE cardNo, WORD axis, WORD FEn, WORD FLn)

6.2.2.1.5 Other Settings



Description

Position Counter Variable Ring

This function sets the maximum number of pulses needed in order to rotate one full cycle. This function is for circular motion and not for linear motion and is useful for managing the rotation position.

The related functions are

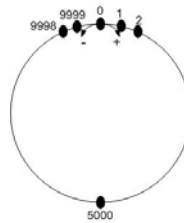
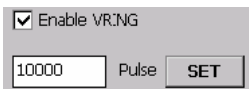
`VRING_ENABLE(BYTE cardNo, WORD axis, DWORD nVRing)`

`VRING_DISABLE(BYTE cardNo, WORD axis)`

Example:

The ring counter is set to 9999.

The encoder range will be from 0 to 9999. The counter will reset to 0 when its current value is 9999 and it is incremented by one pulse. A pulse count down by one will result in a counter value of 9999 when the counter value is currently 0.



The count operation will be as follows:

Increment in the + direction: ...
 $\rightarrow 9998 \rightarrow 9999 \rightarrow 0 \rightarrow 1 \rightarrow \dots$

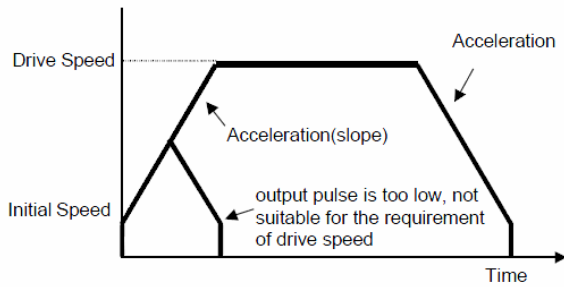
Increment in the - direction: ...
 $\rightarrow 1 \rightarrow 0 \rightarrow 9999 \rightarrow 9998 \rightarrow \dots$

Triangle prevention of fixed pulse driving

This function prevents a triangle form in linear

Enable AVTRI

acceleration (T-curve) fixed pulse driving even if the number of output pulses is low.



The related functions are

```
AVTRI_ENABLE (BYTE cardNo, WORD axis);
AVTRI_DISABLE (BYTE cardNo, WORD axis);
```

Setting the Maximum Speed

This function sets the maximum rate for the output pulses (speed). A larger value will cause a rougher resolution.

For example,

- when the maximum speed is set to 8000 PPS, one speed unit is equal to 1 PPS;
- when the maximum speed is set to 16000 PPS, one speed unit is equal to 2 PPS;
- when the maximum speed is set to 80000 PPS, one speed unit is equal to 10 PPS, etc.

The maximum value is 4,000,000 PPS, which means the resolution of speed will be 500 PPS.

 Enable MaxV
 Pulse

This function changes the resolution of speed to reach the desired maximum speed. Since the scale in hardware is changed, other parameters will be influenced too, such as the starting speed, the acceleration, and the jerk. It is recommended to set the maximum speed value as an integral multiplier of 8000.

The related function is

```
SET_MAX_V(BYTE cardNo, WORD axis, DWORD data);
```

6.2.3 Basic Operation: Independent axis motion

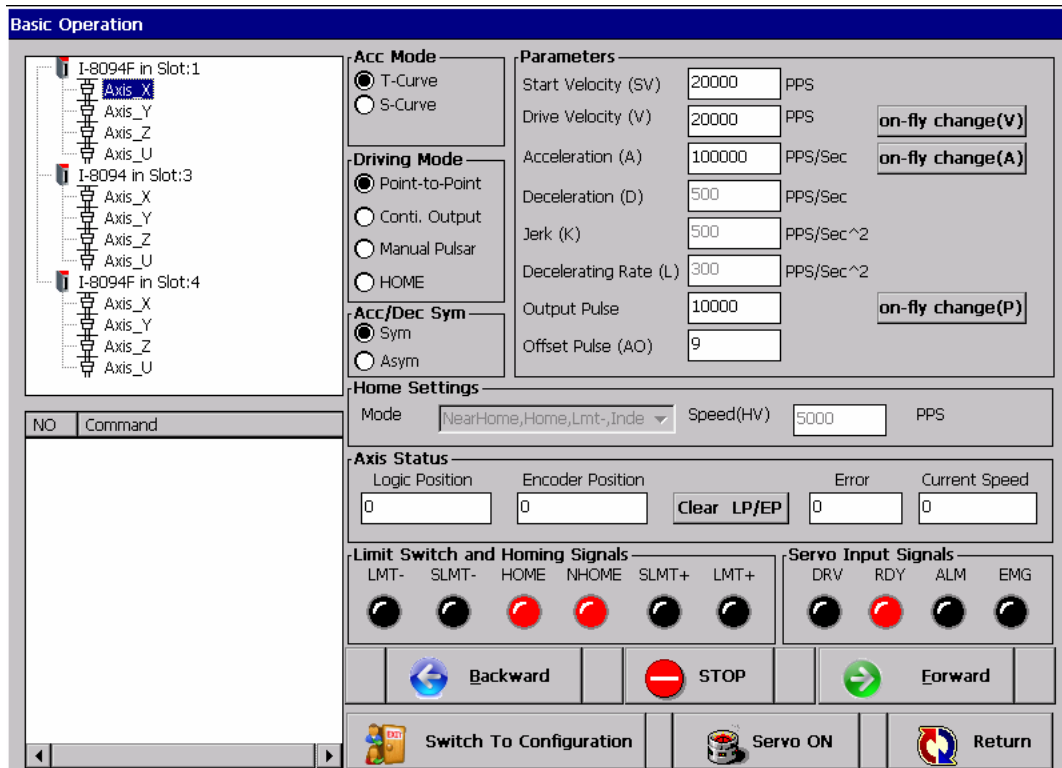
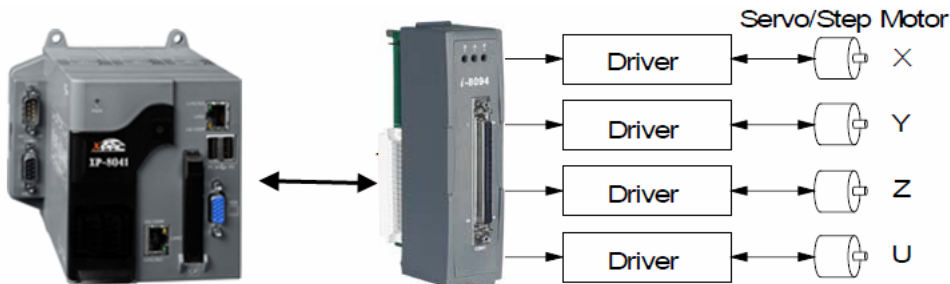


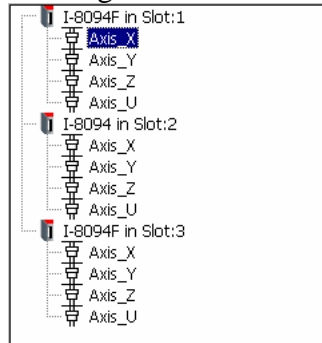
Figure 32: Single axis testing window

Each axis of a motion card executes motion commands independent from the other axis of the same module. Each axis move exactly as it has been programmed either from point to point or in a continuous motion. Each axis will execute its motion commands regardless of other axes.



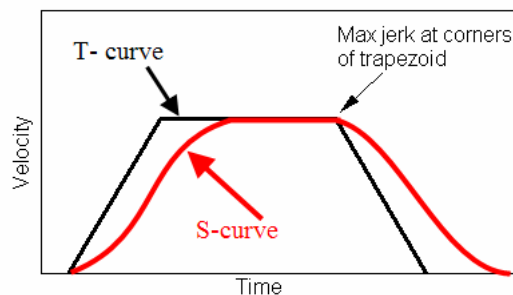
6.2.3.1 Configuration procedure

Step 1: First select an axis of a specific motion card. A selected axis is indicated by a blue background.



Step 2: Select the acceleration mode. T-curve and S-curve refers to the shape of the velocity profile.

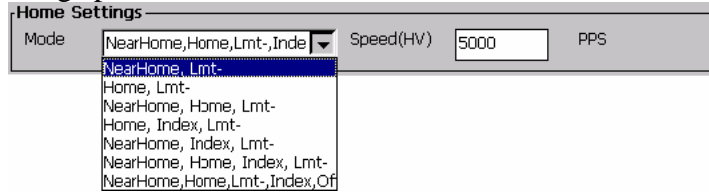
- T-curve (Trapezoidal-curve)
In a T-curve profile the motor tries to go from 0 to the specified acceleration instantaneously. When the motor is decelerating, it once again goes from 0 acceleration to a negative acceleration as fast as it can until it reaches 0 velocity and then abruptly stops. These abrupt starts and stops reduce the life of mechanical components.
- S-curve
The s-curve is used to slowly reach a certain acceleration or deceleration value. Acceleration and deceleration changes occur smoothly and thereby reduce the stress in the mechanical components.



Step 3: Select the driving mode

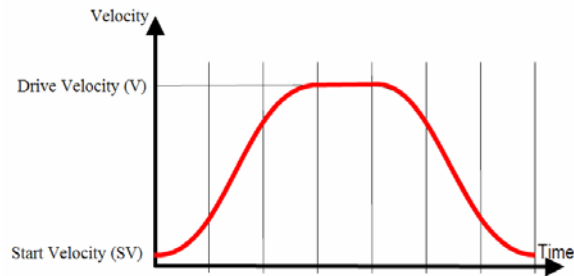
- Point to point motion
- Continuous motion (Conti. Output)
Continuous pulse driving output:
The motion card will continuously output pulses at a specified speed until is interrupted by a stop command or an external stop signal.
- Manual pulsar
Output pulses are generated from a hand wheel.
- Home

If home has been selected enter the required home setting mode and homing speed.

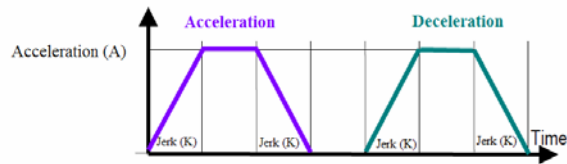


Step 4: Decide between a symmetrical and asymmetrical velocity profile.

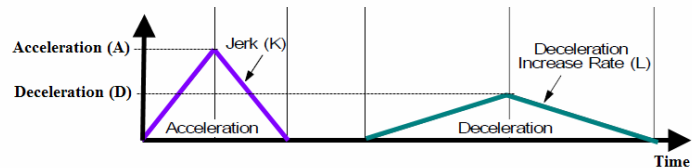
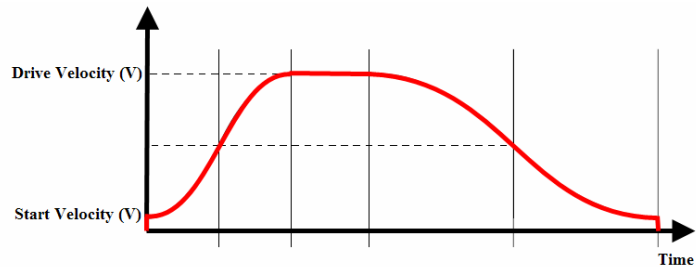
- **Symmetrical profile:**
For the symmetrical profile the absolute acceleration and deceleration values are equal.



Acceleration/ Deceleration



- **Asymmetrical profile:**
The asymmetrical velocity profile allows the individual setting of the acceleration and deceleration values.



Step 5: Enter values for the motion parameters:

Parameters		
Start Velocity (SV)	<input type="text" value="20000"/>	PPS
Drive Velocity (V)	<input type="text" value="20000"/>	PPS <input type="button" value="on-fly change(V)"/>
Acceleration (A)	<input type="text" value="100000"/>	PPS/Sec <input type="button" value="on-fly change(A)"/>
Deceleration (D)	<input type="text" value="500"/>	PPS/Sec
Jerk (K)	<input type="text" value="500"/>	PPS/Sec^2
Decelerating Rate (L)	<input type="text" value="300"/>	PPS/Sec^2
Output Pulse	<input type="text" value="10000"/>	<input type="button" value="on-fly change(P)"/>
Offset Pulse (AO)	<input type="text" value="9"/>	




Step 6: Set the logic and encoder position to zero by clicking the “Clear LP/EP” button.

Axis Status				
Logic Position	Encoder Position	<input type="button" value="Clear LP/EP"/>	Error	Current Speed
<input type="text" value="0"/>	<input type="text" value="0"/>		<input type="text" value="0"/>	<input type="text" value="0"/>

Step 7: Enable the motor driver by clicking the “Servo ON” button.

Step 8: Now you can send the motion controller the motion command for the selected operating mode.

The servo motor can either be moved in the positive or negative direction:

- Positive direction: click  Forward
- Negative direction: click  Backward
- To stop the servo motor click  STOP. This button will cause the motor to stop immediately.

If the driving mode is set to “Manual Pulsar” the motor follows the outputting pulses of a manual pulse generator.

The command window on the bottom left list all the commands send to the control cards with their specified parameter values.

Command
i8094MF_SET_V(1,2,20000)
i8094MF_SET_AO(1,2,9)
i8094MF_SET_SV(1,2,20001)
i8094MF_NORMAL_SPEED(1,2,0)
i8094MF_SET_A(1,2,100000)
i8094MF_FIXED_MOVE(1,2,10000)
i8094MF_SET_V(1,2,20000)
i8094MF_SET_AO(1,2,9)
i8094MF_SET_SV(1,2,20000)
i8094MF_NORMAL_SPEED(1,2,0)
i8094MF_SET_A(1,2,100000)
i8094MF_FIXED_MOVE(1,2,10000)

6.2.3.2 Status Indicators

The status indicators reflect the current state of the hardware limit, near home and home switches. In addition the current state of the servo motor is displayed that means whether it is in driving mode or has successfully completed the motion command. Alarm or emergency stop events are also indicated.



Input signals	Description
LMT-	Negativ hardware limit switch
SLMT-	Negative software limit
HOME	Home hardware signal
NHOME	Near home signal
SLMT+	Positive software limit
LMT+	Positive hardware limit switch
DRV	Servo motor is busy driving
RDY	Ready signal
ALM	Alarm input signal
EMG	Emergency

6.2.4 Advance motion features: Multi-axis interpolation

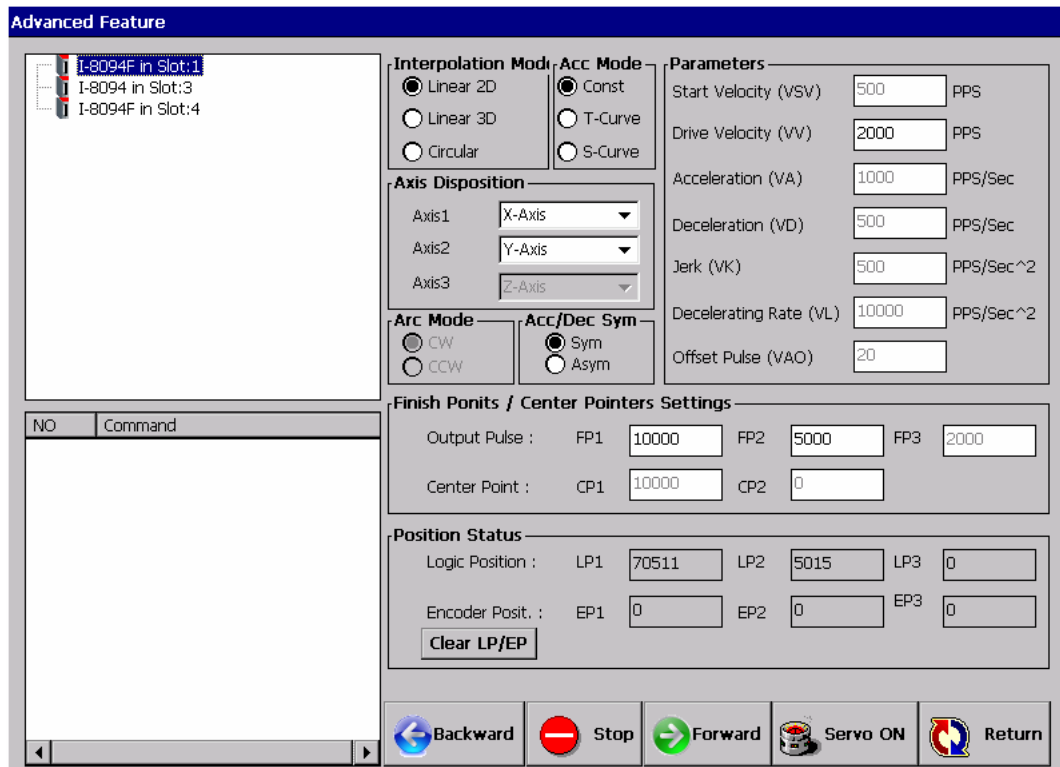


Figure 33: Multi-axis interpolation window

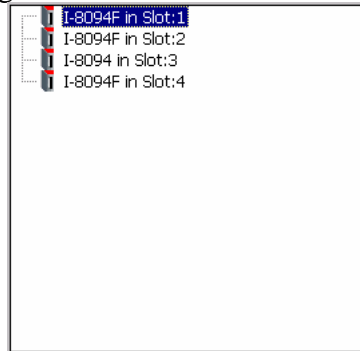
This window assists you in testing different interpolation modes. The 4-axis motion control cards offered by ICPDAS perform any 2/3 axes linear interpolation and any 2 axes circular interpolation. Any 2 or 3 axes can be selected to perform linear interpolation.

6.2.4.1 Configuration procedure

6.2.4.1.1 Linear interpolation

The following steps describe the procedure of executing a 2/3 axes linear interpolation:

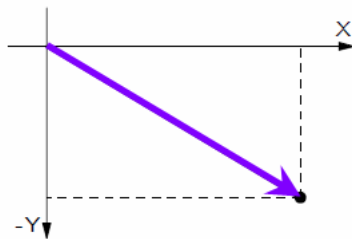
Step 1: Select a motion module from the tree view on the left top screen by clicking on one of the listed modules.



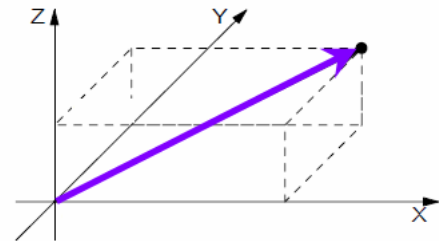
Step 2: Select the interpolation mode:

- Two axes linear interpolation
- Three axis linear interpolation

2-axis linear interpolation



3-axis linear interpolation



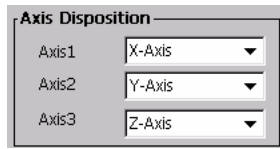
Step 3: Select an acceleration mode.

- Const – no acceleration
- T-curve
- S-curve

Step 4: Decide between a symmetrical and asymmetrical velocity profile.

- Symmetrical profile
- Asymmetrical profile

Step 5: Select the interpolation axes. Any 2 or 3 axis of a motion module can be selected to perform linear interpolation. The axis ports on the daughter board are labeled as X-Axis, Y-Axis, Z-Axis and U-Axis. **Axis Port:** This is the port number on the axis board which the motor is connected to.



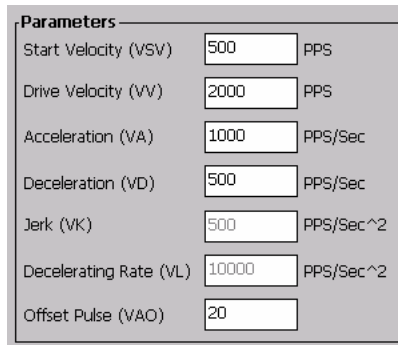
Axis Disposition

Axis1: X-Axis

Axis2: Y-Axis

Axis3: Z-Axis

Step 6: Enter values for the motion parameters:



Parameters

Start Velocity (VSV): 500 PPS

Drive Velocity (VV): 2000 PPS

Acceleration (VA): 1000 PPS/Sec

Deceleration (VD): 500 PPS/Sec

Jerk (VK): 500 PPS/Sec²

Decelerating Rate (VL): 10000 PPS/Sec²

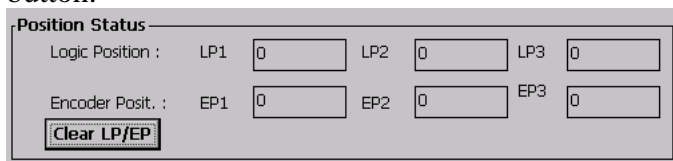
Offset Pulse (VAO): 20

Step 7: Enter for each axis the number of pulse to move from the current position. Set the coordinates for the new position relative to the current position.



Output Pulse : FP1: 10000 FP2: 5000 FP3: 2000

Step 8: Set the logic and encoder position to zero by clicking the “Clear LP/EP” button.



Position Status




Logic Position : LP1: 0 LP2: 0 LP3: 0

Encoder Posit. : EP1: 0 EP2: 0 EP3: 0

Clear LP/EP

Step 9: Enable the motor driver by clicking the “Servo ON” button.

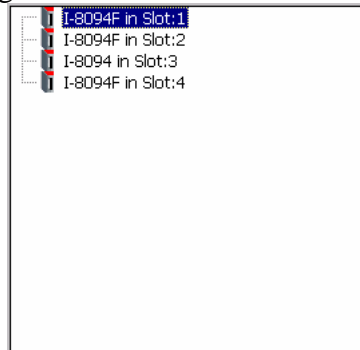
Step 10: Output the specified number of pulses to the servo motor. You can either move in the positive or negative direction.

- Positive direction: click 
- Negative direction: click 
- To stop the servo motor click 

6.2.4.1.2 Circular interpolation

The following steps describe the procedure of executing a circular interpolation:

Step 1: Select a motion module from the tree view on the left top screen by clicking on one of the listed modules.



Step 2: Select the interpolation mode:

- Circular interpolation

Step 3: Select the direction of the circular interpolation.

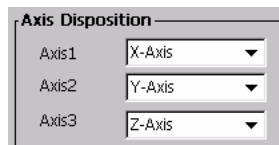
- Clockwise (CW)
- Counterclockwise (CCW)

Step 4: Decide between a symmetrical and asymmetrical velocity profile.

- Symmetrical profile
- Asymmetrical profile

Circular interpolation only supports the T-curve acceleration mode.

Step 5: Select the axis for the interpolation. Any 2 of the 4 axis can be selected for circular interpolation. The axis ports on the daughter board are labeled as X-Axis, Y-Axis, Z-Axis and U-Axis.



Step 6: Enter values for the motion parameters:

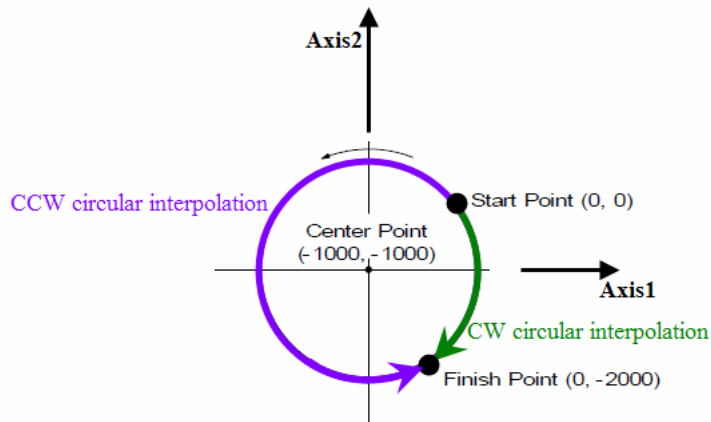
Parameters		
Start Velocity (VSV)	<input type="text" value="500"/>	PPS
Drive Velocity (VV)	<input type="text" value="2000"/>	PPS
Acceleration (VA)	<input type="text" value="1000"/>	PPS/Sec
Deceleration (VD)	<input type="text" value="500"/>	PPS/Sec
Jerk (VK)	<input type="text" value="500"/>	PPS/Sec ²
Decelerating Rate (VL)	<input type="text" value="10000"/>	PPS/Sec ²
Offset Pulse (VAO)	<input type="text" value="20"/>	

Step 7: Enter the center point and for each axis the number of pulse to move from the current position.

The circular interpolation starts from the current position. Therefore it is assumed that the current position (start point) is (0,0). After setting the center point, finish point and the direction (CW or CCW), the user can start the circular interpolation.

Note: The coordinates are relative to the start point.

Finish Points / Center Pointers Settings						
Output Pulse :	FP1	<input type="text" value="0"/>	FP2	<input type="text" value="-2000"/>	FP3	<input type="text" value="2000"/>
Center Point :	CP1	<input type="text" value="-1000"/>	CP2	<input type="text" value="-1000"/>		






Step 8: Set the logic and encoder position to zero by clicking the “Clear LP/EP” button.

Position Status						
Logic Position :	LP1	<input type="text" value="0"/>	LP2	<input type="text" value="0"/>	LP3	<input type="text" value="0"/>
Encoder Posit. :	EP1	<input type="text" value="0"/>	EP2	<input type="text" value="0"/>	EP3	<input type="text" value="0"/>
<input type="button" value="Clear LP/EP"/>						

Step 9: Enable the motor driver by clicking the “Servo ON” button.

Step 10: Output the specified number of pulses to the servo motor. You can either move in the positive or negative direction.

- Clockwise direction: click 
- Counterclockwise direction: click 
- To stop the servo motor click 

7 EzMake

7.1 Introduction

EzMake is a utility which assist the user in programming, debugging and testing simple motion control macros. It can only be used together with the i-8094H motion control module. The macros have to be called by the motion control program written by the user in C. The advantage macros is that for minor changes instead of modifying and recompiling the main program only the macros have to be changed and downloaded to the motion module. The controller can therefore be adapted very quickly to the new requirements.

The EzMake utility provides the operator with a list of valid commands and prevents the input of invalid commands or parameters. With EzMake you can create three different types of macros:

- Initialization macro: Responsible for the basic configuration of the motion card.
- Motion control macro: This macro contains the actual motion control command sequence.
- Interrupt service routine macros: Is being called when an interrupt of the motion card occurs.

These macro files have to be downloaded to the i-8094H module. Macros are stored in a nonvolatile memory.

7.1.1 *Main user interface*

The tree view on the left lists the project files and the three macro file types created by the user: Initial table, Macro program and Interrupt Service Routine. The middle section of the main window forms part of the macro editing interface. Here the user has to edit or modify motion commands and their parameter for the different macro programs. The section on the right lists all the motion and macro commands supported by the i-8094H module. The commands can be selected from the command list by a double click. For some files types certain commands can not be used and are therefore disabled. Disabled commands show their icons in gray color tone. The window at the bottom informs the user about programming restrictions and different axis and motion statuses for online debugging.

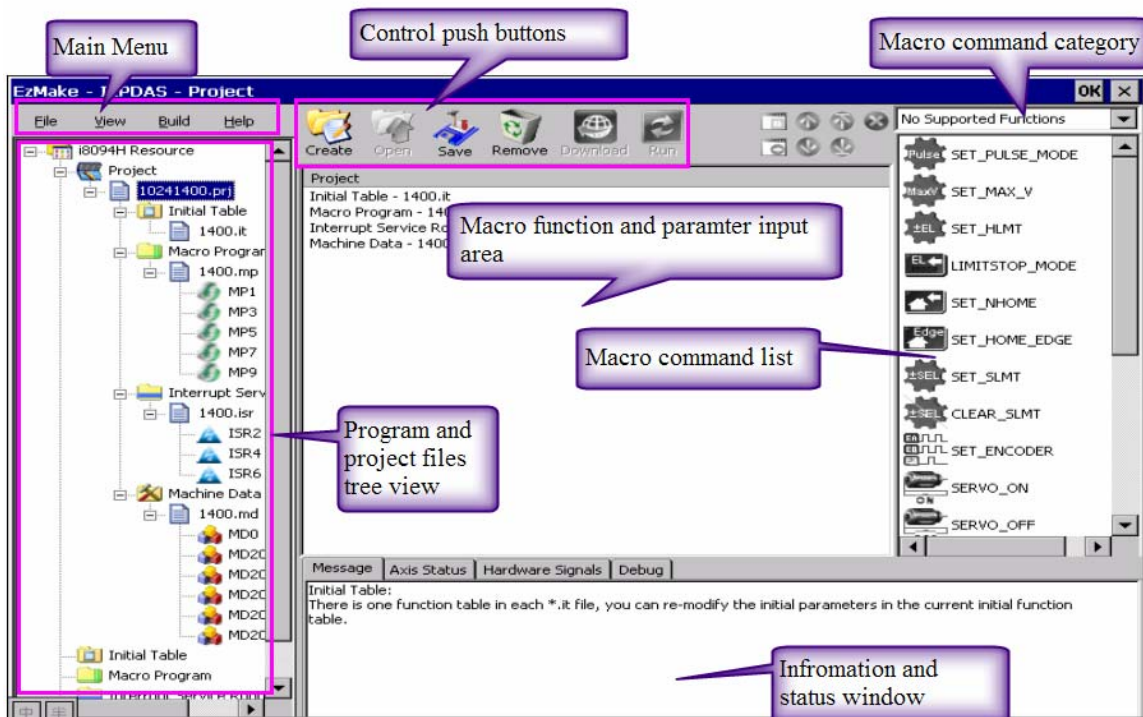


Figure 34: EzMake user interface

7.2 Initial Table

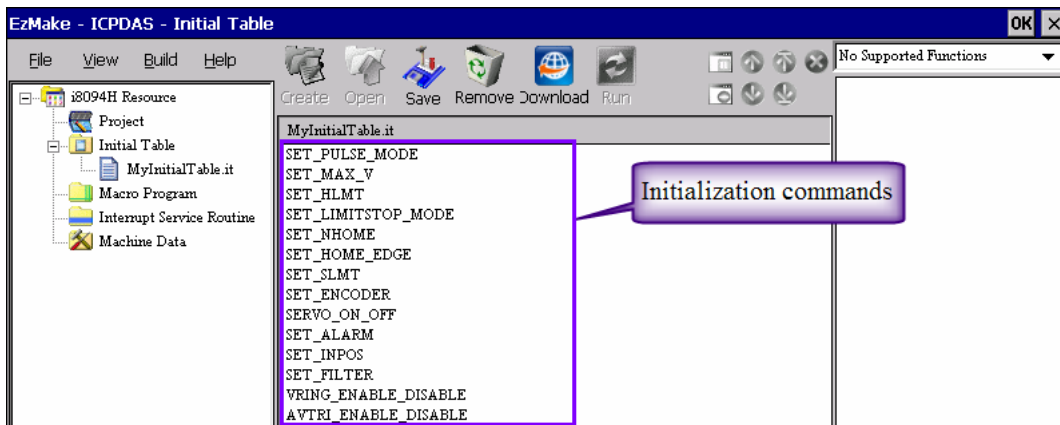
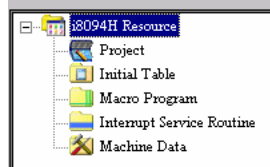


Figure 35: Initialization commands

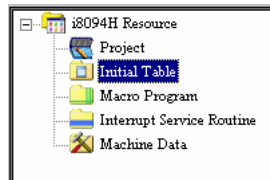
The initial table lists all the commands necessary to initialize the i-8094H module. The sequence of the commands is fixed and commands can not be removed from the list. In a newly created initial table all commands have default parameter values. Double click a command to change its default value(s).

7.2.1 Create a new initialization table

Step 1: Open the “i8094H Resource” tree in the tree view on the left hand sight to display the following five folders:

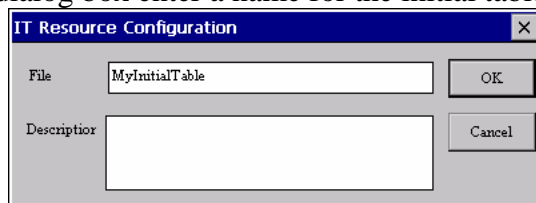


Step 2: Click on the “Initial Table” folder.



Step 3: Click *File* → *Create* to create a new initial table. You can also directly click the “Create” button in the toolbar.

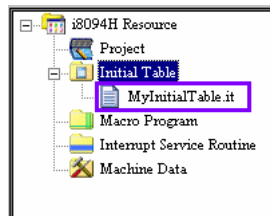
Step 4: In the dialog box enter a name for the initial table and click “OK”.



Step 5: The new table will be added to the “Initial Table” folder.

Note:

- i. The new table is visible in the “Initial Table” folder but the file in fact has not been created yet. It will only be created once you save the table.
- ii. You can add at most 5 tables to the Initial Table folder.



Step 6: Click on the newly created table. The command sequence for initializing the motion control module appears in the middle section of the main window. The commands and command sequence can not be change but the parameters of the commands have to be modified to meet your motion

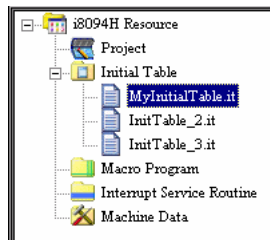
control requirements.

```
MyInitialTable.it
SET_PULSE_MODE
SET_MAX_V
SET_HLMT
SET_LIMITSTOP_MODE
SET_NHOME
SET_HOME_EDGE
SET_SLMT
SET_ENCODER
SERVO_ON_OFF
SET_ALARM
SET_INPOS
SET_FILTER
WRING_ENABLE_DISABLE
AVTRI_ENABLE_DISABLE
```

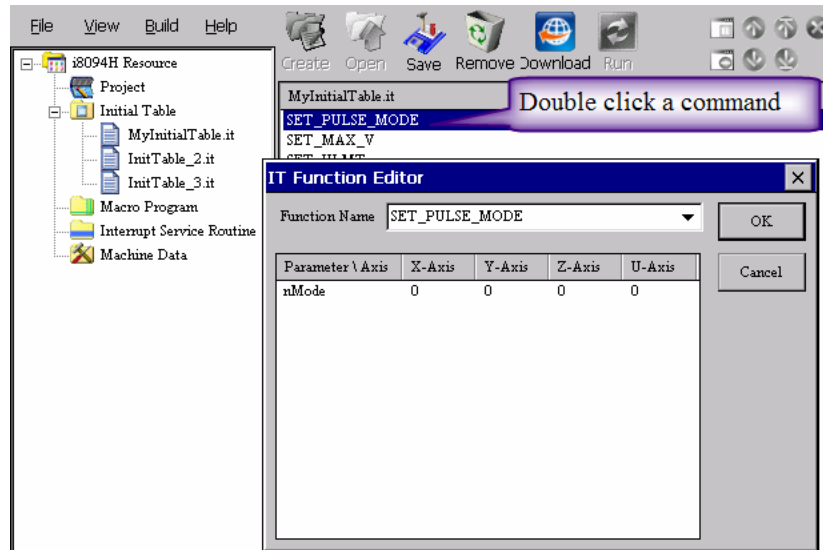
Step 7: Save the initialization table: Click on the table folder and then the “Save” button of the toolbar. The table filename extension is .it.
The file is saved to the following directory:
“\System_Disk\EzProg-I\EzMake”

7.2.2 *Modifying a initialization table*

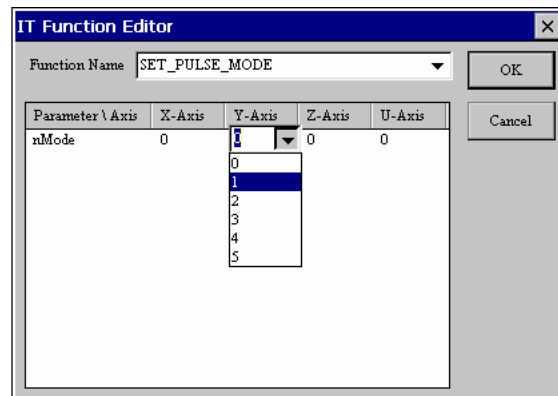
Step 1: Click on one of the initialization table in the “Initial Table” folder to display its initialization commands. Initialization tables in the tree view are identified by names ending with .it . Remember that it is neither possible to remove a command nor change the command sequence in the table.



Step 2: Double click a command to open the window for editing the function parameter.



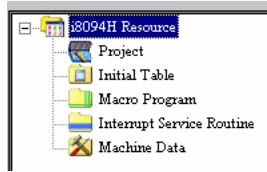
Step 3: Click on one of the parameter to enter or select the parameter value(s) according your system requirement. Confirm the new setting by clicking “OK”.



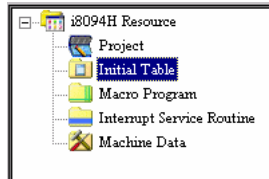
Step 4: After all the parameters of the initializing commands have been set save the file (toolbar: “Save”).

7.2.3 Open an initialization table file

Step 1: Open the “i8094H Resource” tree in the tree view on the left hand sight.

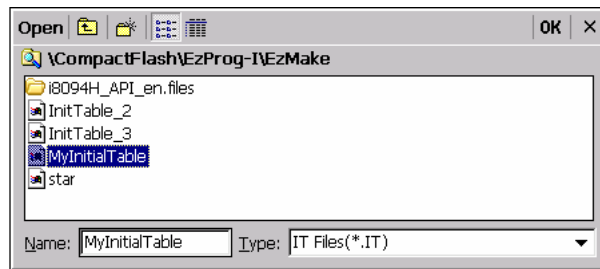


Step 2: Click on the “Initial Table” folder.

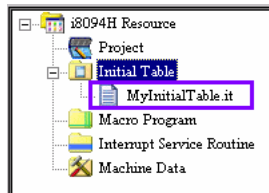


Step 3: Click *File* → *Open* to select an existing file with the extension .it. You can also directly click the “Open” button in the toolbar. Make sure that the file you select is in the following directory:

“\System_Disk\EzProg-I\EzMake”



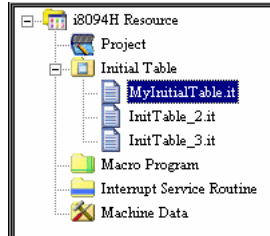
The table will be added to the “Initial Table” folder:



7.2.4 Remove an initialization table from the tree view

Initialization tables can be removed from the tree view without actually deleting the file. You can always add the file again to the “Initial Table” folder.

Step 1: Select the initialization table you like to remove from the “Initial Table” folder.

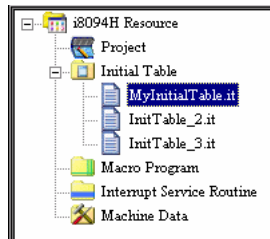


Step 2: Click the “Remove” button in the tool bar.

7.2.5 Downloading of an initialization file

After the initialization table has been set it can be directly downloaded to the i-8094H motion module.

Step 1: Select the initialization table you like to download to the i-8094H module.



Step 2: Click the “Download” button in the tool bar. The file will be downloaded to a non volatile memory of the i-8094H module and will immediately initialize the module to the new setting.

7.3 Macro Program Files (MP Files)

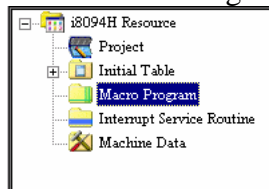
The main task of the EzMake utility is to assist the user to write motion control macros for the i-8094H module. The utility always provides the user with a list of available motion commands for the respective macro and thereby ensures that not an incorrect command is being used. In addition the utility supports debugging your macro.

A macro file contains one or more macro forms called MP. The i-8094H supports up to 157 macro forms (MP1~MP157) with different sizes (stacks). The size indicates how many command lines a macro form supports. Therefore a macro form has to be selected according to the number of motion commands to be used in a macro. The macro forms are divided according to their sizes into five categories (8/16/32/64/128/512 stacks). A macro form with a stack size of 32 has space for 32 instructions. The user can add up to 157 macro forms to one macro file.

7.3.1 Create a new macro file

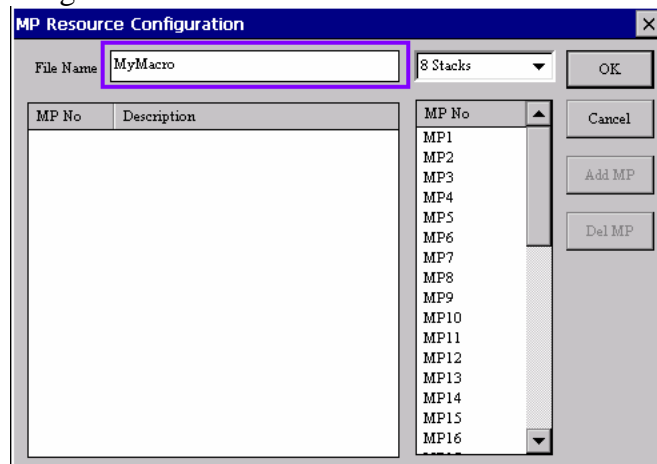
Follow the steps to create a new macro file:

Step 1: Click on the “Macro Program” folder.



Step 2: Click *File* → *Create* to create a new macro file. You can also directly click the “Create” button in the toolbar.

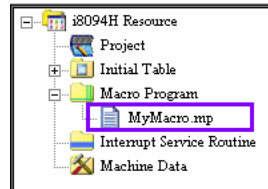
Step 3: In the dialog box enter a name for the macro file and click “OK”



Step 4: The name of the new macro file will be added to the “*Macro Program*” folder.

Note:

- i. The actual macro file has not been created yet. It will only be created once you save the macro file (see next step).
- ii. You can add at most 5 macro files to the “*Macro Program*” folder.



Step 5: Save the macro file: Click on the name of the macro file in the tree view and then the “Save” button of the toolbar. The macro filename extension is .mp.

The file is saved to the following directory:

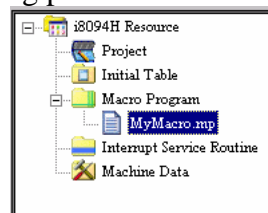
“\System_Disk\EzProg-I\EzMake”

7.3.2 Adding a macro form to a macro file

Each macro file can save up to 157 macro forms. The macro forms are numbered from MP1 to MP157. It is only possible to add one macro form of the same number to the macro file.

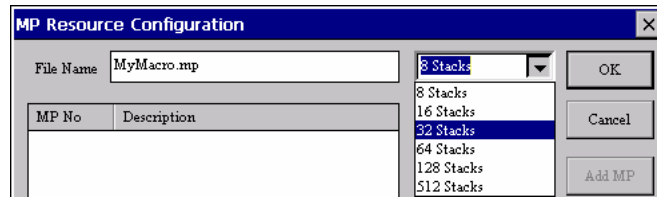
The next steps describe the procedure to add a macro form to a macro file:

Step 1: Click on the macro file folder you want to add a macro form. In the following picture the macro file is *MyMacro.mp*.

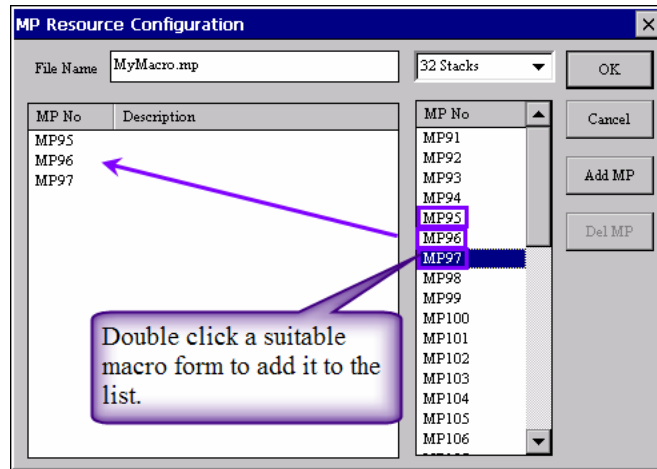


Step 2: Click the “Create” button in the toolbar.

Step 3: Determine the number of command lines the macro form has to support. Six different form sizes are available. Always make sure that a form is selected which provides more command lines and not less than is actually needed for the macro. For example if the macro has to hold 20 commands then a macro form with 32 stacks has to be selected.

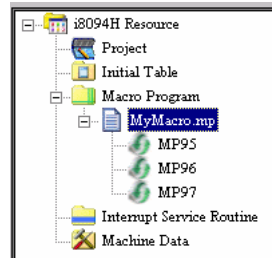


Step 4: Double click a macro name to add it to the macro file list. More than one macro form can be selected.



Step 5: Click “OK”. The name of the new macro forms are shown in the macro file folder.

Note: The macro forms have not been saved yet.



Step 6: Save the macro file: Click on the name of the macro file in the tree view and then the “Save” button of the toolbar.

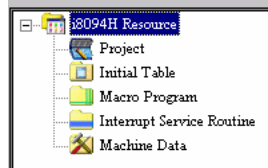
Step 7: Now start adding motion control commands to the macro forms.

7.3.3 Open a macro file

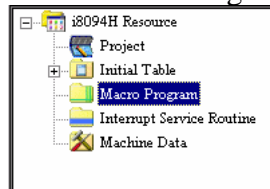
A macro file is a file which contains one or more macro programs. When a macro file is opened all the macro forms in the file will be loaded to the utility.

The following steps describe the loading procedure:

Step 1: Open the “i8094H Resource” tree in the tree view on the left hand sight.

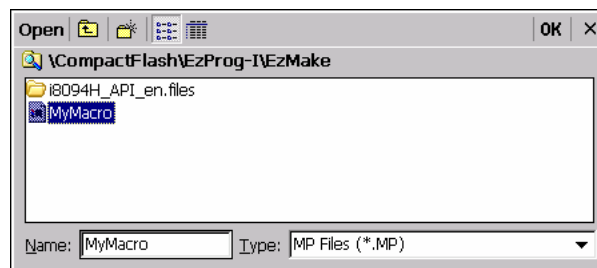


Step 2: Click on the “Macro Program” folder.

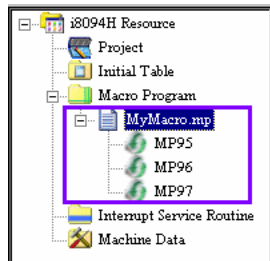


Step 3: Click the “Open” button in the toolbar. Make sure that the file you select is in the following directory:

“\System_Disk\EzProg-I\EzMake”



The table will be added to the “Macro Program” folder:

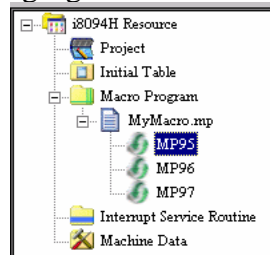


7.3.4 Writing macros (motion commands)

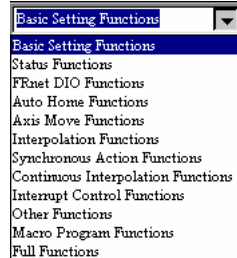
The motion commands can not be edited via keyboard to the macro form but has to be selected from the command list. Consult the i-8094H user manual for a detailed description of the individual commands. This section describes how to use the EzMake utility to write macro programs but does not give any assistance in writing an actual motion control macro program.

7.3.4.1 Adding motion commands to a macro form

Step 1: Click on a macro form of a macro file. For example “MP95” in the following figure:

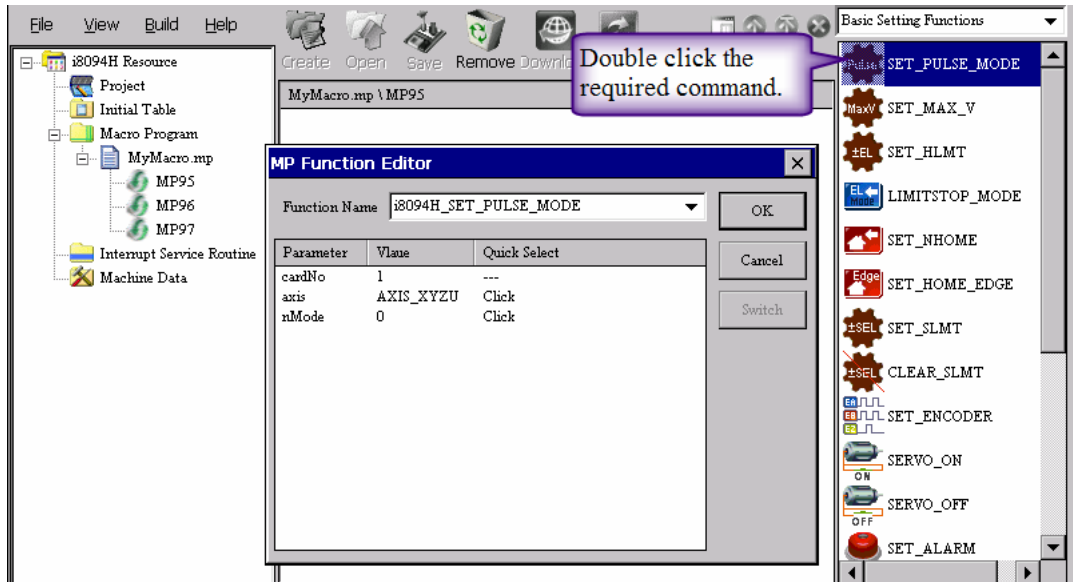


Step 2: The motion commands are arranged into 11 categories to facilitate the selection. Select a command category.

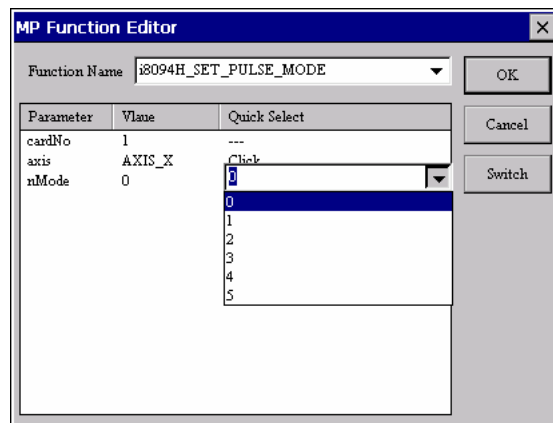


After selecting a category a list of motion commands appears on the right side of the utility window.

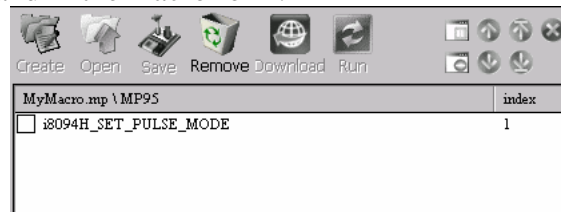
Step 3: Double click a command to add it to the macro form.



Step 4: Set all parameters for the command. Click with the mouse on the row of a parameter in the “Quick Select” column to open a combo box with all possible parameters values.








Step 5: After all the parameters have been set close the dialog window by clicking “OK”. The new command will be added one line below the last macro command in the macro form.



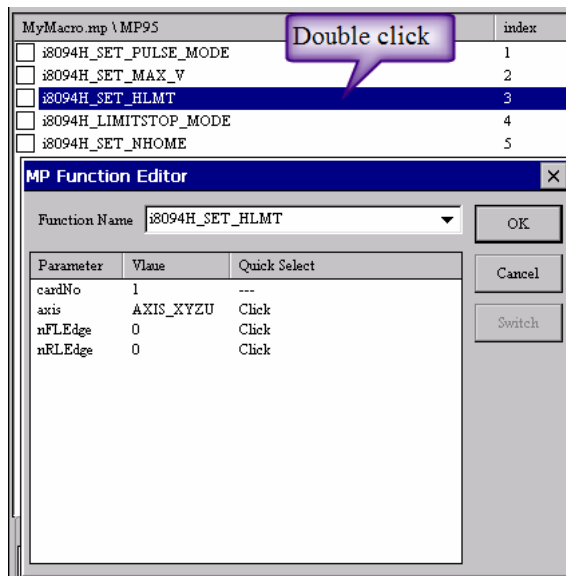
Step 6: Save the macro file after you have finished editing commands to the macro form: Click on the name of the macro file in the tree view and then the “Save” button of the toolbar.

7.3.4.2 Modifying a macro program

A command in the macro form can be manipulated in the following way:

Icon	Description
	Move command up one line
	Move command down one line
	Move command to the first line
	Move command to the bottom line
	Delete command

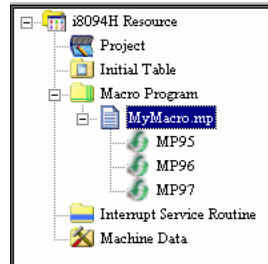
Parameter values are change by double clicking the command in the macro form. The parameter window popping up allows you to modify the existing values of the selected command.



7.3.5 Downloading and executing a macro file

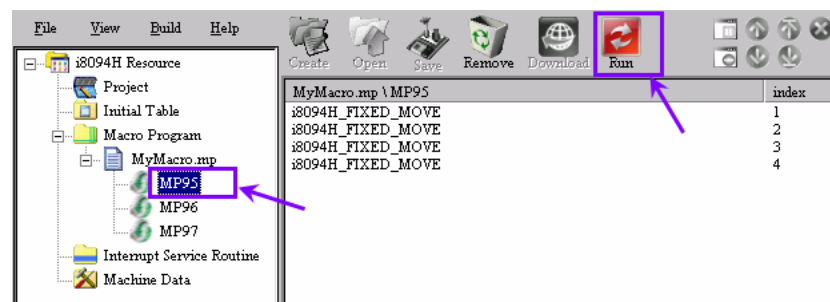
After the macro programming has been completed the macro files has to be downloaded to the i-8094H motion module.

Step 1: Select the macro file in the “Macro Program” folder you like to download to the i-8094H module.







Step 2: Click the “Download” button in the tool bar. The file will be downloaded to a non volatile memory of the i-8094H module.

Step 3: Now you can directly call and execute the macro on the i-8094H module and monitor the command executions. Select a macro from the macro file and click the “Run” button on the toolbar.



The utility highlights the command currently being executed by the motion module. Axis statuses such as the logic position, encoder position, velocity and acceleration are displayed in the “Axis Status” tab window at the bottom of the utility. The debug tab window shows which commands have been successfully carried out. The execution of the macro commands can be stopped or aborted at any time.

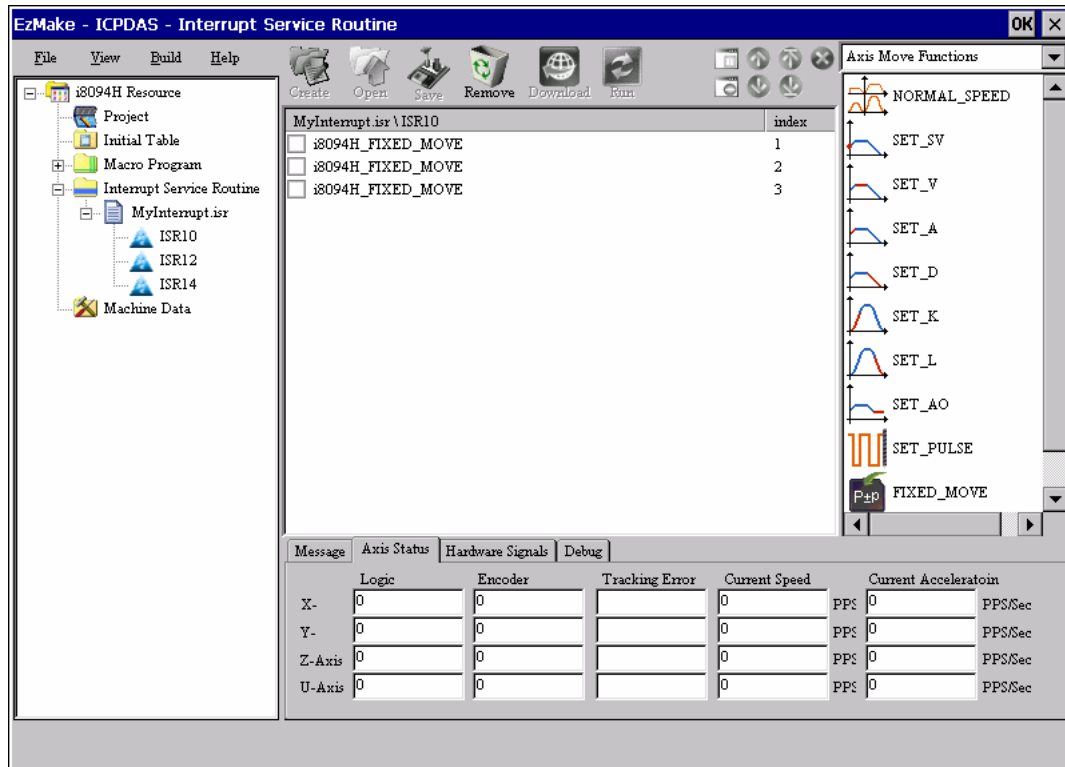
Icon	Description
	Halt macro execution
	Continue macro execution
	Cancel macro execution

Step 4: After the last command has been executed click the  button in the toolbar to end the macro execution mode.

7.4 Interrupt Service Routine (ISR) macro

A macro called by the interrupt service routine is created in the same way as a normal motion control macro described in the previous chapter. The 8094H manual (chapter 6.3) describes in detail the procedure to enable the ISR and link the ISR to a macro. The 8094H module is designed to store up to 20 macros (ISR1 ~ ISR20). The macros are categorized according to the number of commands they can hold (8/16/32/64 commands).

The procedure for adding, deleting, saving and loading of macro forms is very similar to the procedure of a normal motion control macro. The only difference is that the interrupt macro forms are created in the “Interrupt Service Routine” folder and that they are called (ISR1, ISR2, ISR3...). The editing, downloading and testing of an ISR macro is identical to the motion control macro described in the previous chapter. Therefore consult the chapter 7.3 for implementing an ISR macro.



Note:

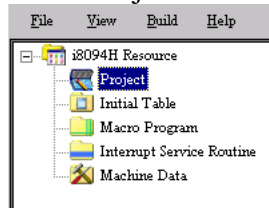
In an ISR macro no “for” loop is allowed and it is not possible to call inside an ISR macro a MP macro.

7.5 Project Files

So far files for the three different macro types had to be created and saved individually. By creating a project file all these macro files can be saved, opened and downloaded at once.

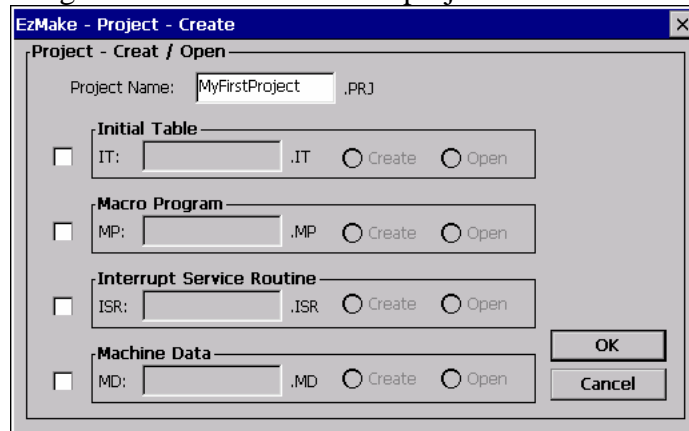
7.5.1 Create a new project file

Step 1: Click on the “Project” folder.

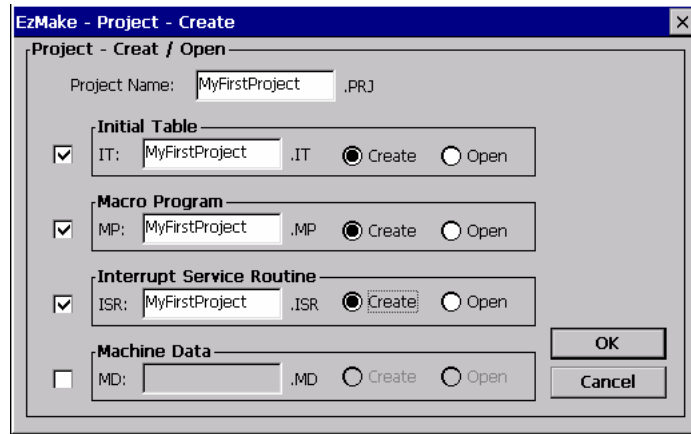


Step 2: Click the “Create” button in the toolbar.

Step 3: In the dialog box enter a name for the project file.

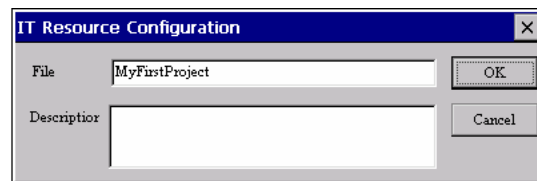


Step 4: Determine which macro types the project file should include. In the following the “Initial Table”, “Macro Program” and “Interrupt Service Routine” are selected. Furthermore the “Create” options have to be selected to indicate that these macros types are newly created. All the newly created macro types automatically adopt the name of the project file. The names can be changed.

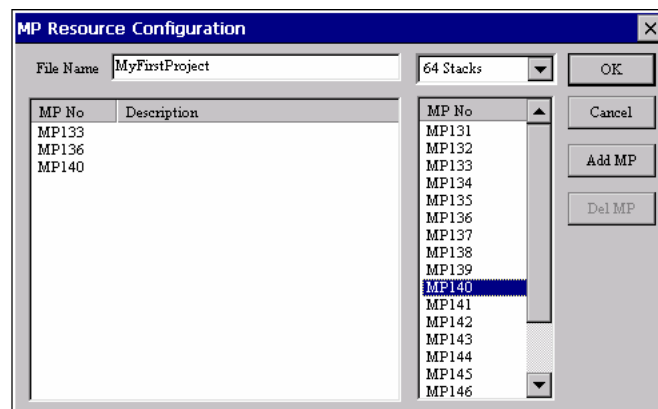


Step 5: Confirm the selection by clicking “OK”.

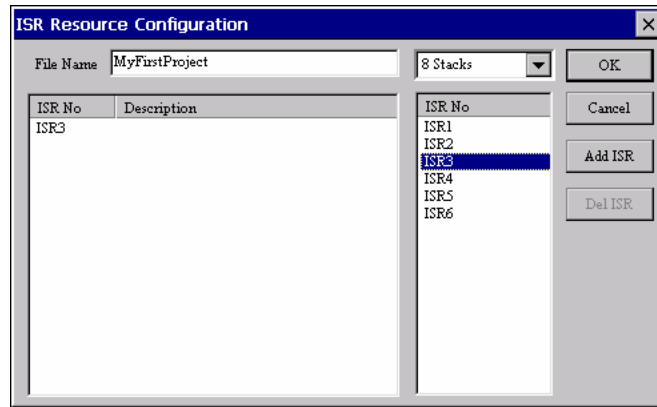
Step 6: Confirm or change the name of the initial table. In addition you can add a comment to the file to describe its purpose. Click “OK”.



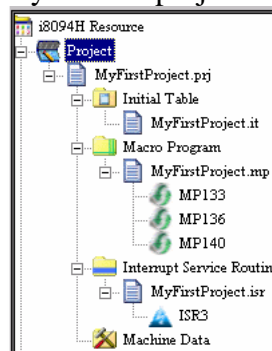
Step 7: Confirm or change the name of the macro program file. Add one or more macro forms to the macro program file by double clicking a macro form name. Click “OK”.



Step 8: Confirm or change the name of the interrupt service routine file. Select one or more macro forms to the ISR file. Click “OK”.



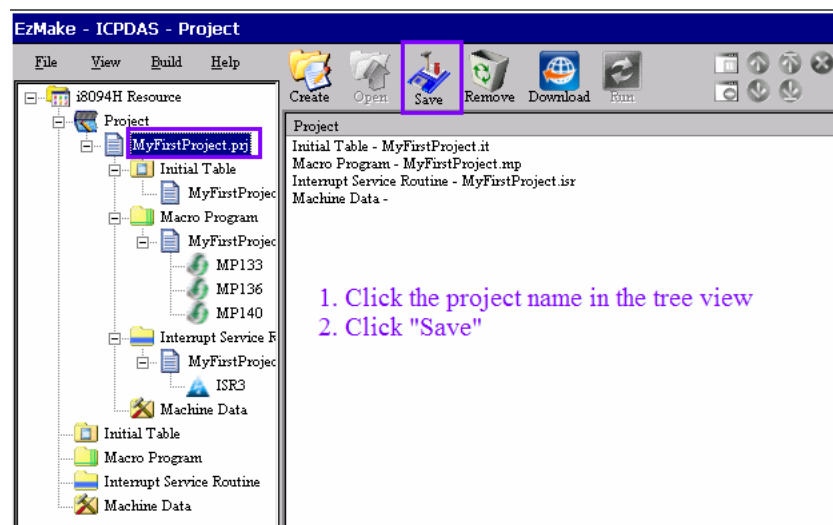
Step 9: The newly created project is now added to the “Project” folder.



Note:

The new project file is visible in the “Project” folder but the file in fact has not been created yet. It will only be created once you save the project.

Step 10: Save the project file: Click on the newly created project name in the tree view and then the “Save” button in the toolbar.



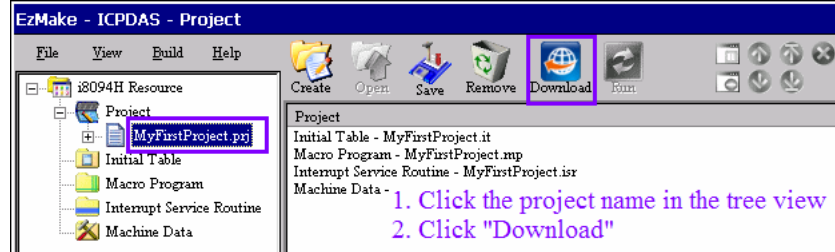
The project filename extension is .prj. The file is saved to the following directory:
“\System_Disk\EzProg-I\EzMake”

Step 11: Now you can start editing the macros.

7.5.2 Downloading a project file

After finishing the macro programming the complete project files can be downloaded to the i-8094H motion module at once or the different macro file types can be downloaded separately.

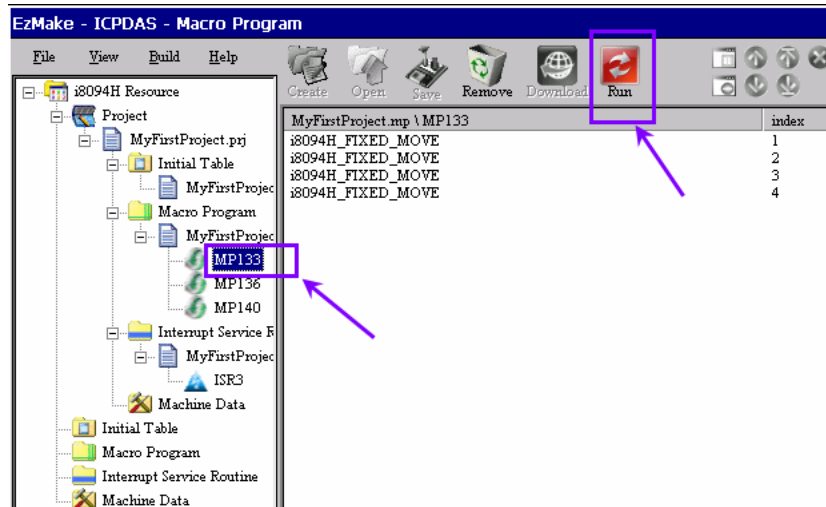
Step 1: Select the project file in the “Project” folder you like to download to the i-8094H module.



If you want to download the different macro types separately, just click on the macro file type (in this example: *MyFirstProject.it*, *MyFirstProject.mp* or *MyFirstProject.isr*) and then the “Download” button on the toolbar.

Step 2: Click the “Download” button in the tool bar. The complete file with all the macros will be downloaded to the non volatile memory of the i-8094H module.

Step 3: Now you can directly call and execute the macro on the i-8094H module and monitor the command executions. Select a macro from the macro file and click the “Run” button on the toolbar.














7.6 Macro motion commands

The following table introduces the macros supported by the EzMake utility. The 8094H manual describes these commands and their parameter setting in more detail. The EzMake support three macro types:








- Macros for module initialization (IT)
- Macros for motion control (MP)
- Macros for interrupt service routine (ISR)

Some macro commands are only valid for a specific macro type. The last three columns of the command table indicate whether the respective macro type supports the command.



7.6.1 Basic Setting Functions

Icon	Function Name	Statement	IT	MP	ISR
	i8094H_SET_PULSE_MODE	This function sets the pulse output mode as either CW/CCW or PULSE/DIR for the assigned axes and their direction definition.	<input type="radio"/>	<input type="radio"/>	
	i8094H_SET_MAX_V	This function sets the maximum rate for the output pulses (speed). A larger value will cause a rougher resolution.	<input type="radio"/>	<input type="radio"/>	
	i8094H_SET_HLMT	This function sets the active logic level of the hardware limit switch inputs.	<input type="radio"/>	<input type="radio"/>	
	i8094H_LIMITSTOP_MODE	This function sets the motion stop mode of the axes when the corresponding limit switches are detected.	<input type="radio"/>	<input type="radio"/>	
	i8094H_SET_NHOME	This function sets the trigger level of the near home sensor (NHOME).	<input type="radio"/>	<input type="radio"/>	
	i8094H_SET_HOME_EDGE	This function sets the trigger level of the home sensor (HOME).	<input type="radio"/>	<input type="radio"/>	
	i8094H_SET_SLMT	This function sets the software limits.	<input type="radio"/>	<input type="radio"/>	
	i8094H_CLEAR_SLMT	This function clears the software limits.	<input type="radio"/>	<input type="radio"/>	
	i8094H_SET_ENCODER	This function sets the encoder input related parameters.	<input type="radio"/>	<input type="radio"/>	
	i8094H_SERVO_ON	This function outputs a DO signal (ENABLE) to enable the motor driver.	<input type="radio"/>	<input type="radio"/>	
	i8094H_SERVO_OFF	This function outputs a DO signal (ENABLE) to disable the motor driver.	<input type="radio"/>	<input type="radio"/>	
	i8094H_SET_ALARM	This function sets the ALARM input signal related parameters.	<input type="radio"/>	<input type="radio"/>	
	i8094H_SET_INPOS	This function sets the INPOS input signal related parameters.	<input type="radio"/>	<input type="radio"/>	
	i8094H_SET_FILTER	This function selects the axes and sets the time constant for digital filters of the input signals.	<input type="radio"/>	<input type="radio"/>	
	i8094H_VRING_ENABLE	This function enables the linear counter of the assigned axes as variable ring counters.	<input type="radio"/>	<input type="radio"/>	
	i8094H_VRING_DISABLE	This function disables the variable ring counter function.	<input type="radio"/>	<input type="radio"/>	
	i8094H_AVTRI_ENABLE	This function prevents a triangle form in linear acceleration (T-curve) fixed pulse driving even if the number of output pulses is low.	<input type="radio"/>	<input type="radio"/>	
	i8094H_AVTRI_DISABLE	This function disables the function that prevents a triangle form in linear acceleration fixed pulse driving.	<input type="radio"/>	<input type="radio"/>	





7.6.2 Status Functions

Icon	Function Name	Statement	IT	MP	ISR
	i8094H_SET_LP	This function sets the command position counter value (logical position counter, LP).		<input type="radio"/>	<input type="radio"/>
	i8094H_GET_LP	This function reads the command position counter value (logical position counter, LP).		<input type="radio"/>	<input type="radio"/>
	i8094H_SET_EP	This function sets the encoder position counter value (real position counter, or EP).		<input type="radio"/>	<input type="radio"/>
	i8094H_GET_EP	This function reads the encoder position counter value (EP).		<input type="radio"/>	<input type="radio"/>
	i8094H_GET_DI	This function reads the digital input (DI) status.		<input type="radio"/>	<input type="radio"/>
	i8094H_GET_ERROR	This function checks whether an error occurs or not.		<input type="radio"/>	<input type="radio"/>
	i8094H_GET_ERROR_CODE	This function reads the ERROR status.		<input type="radio"/>	<input type="radio"/>












7.6.3 FRnet DIO Functions

Icon	Function Name	Statement	IT	MP	ISR
	i8094H_FRNET_IN	This function reads the FRnet digital input signals. One group comprises 16 bits data. Therefore, total 128 DI can be defined for one FRnet interface.		<input type="radio"/>	<input type="radio"/>
	i8094H_FRNET_OUT	This function writes data to the FRnet digital output. One group comprises 16 bits data. Therefore, total 128 DO can be defined for one FRnet interface.		<input type="radio"/>	<input type="radio"/>



7.6.4 Auto Home Functions














Icon	Function Name	Statement	IT	MP	ISR
	i8094MF_SET_HV	This function sets the homing speed.		<input type="radio"/>	
	i8094MF_HOME_LIMIT	This function sets the Limit Switch to be used as the HOME sensor.		<input type="radio"/>	
	i8094MF_SET_HOME_MODE	This function sets the homing method and other related parameters.		<input type="radio"/>	
	i8094MF_HOME_START	This function starts the home search of assigned axes.		<input type="radio"/>	

7.6.5 Axis Move Functions






Icon	Function Name	Statement	IT	MP	ISR
	i8094H_NORMAL_SPEED	The function sets the speed mode.		<input type="radio"/>	<input type="radio"/>
	i8094H_SET_SV	This function sets the start speed for the assigned axes.		<input type="radio"/>	<input type="radio"/>
	i8094H_SET_V	This function sets the desired speed for the assigned axes.		<input type="radio"/>	<input type="radio"/>
	i8094H_SET_A	This function sets the acceleration value for the assigned axes.		<input type="radio"/>	<input type="radio"/>
	i8094H_SET_D	This function sets the deceleration value for the assigned axes.		<input type="radio"/>	<input type="radio"/>
	i8094H_SET_K	The function sets the acceleration rate (i.e., Jerk) value for the assigned axes.		<input type="radio"/>	<input type="radio"/>
	i8094H_SET_L	This function sets the deceleration rate (i.e., Jerk) value for the assigned axes.		<input type="radio"/>	<input type="radio"/>
	i8094H_SET_AO	This function sets the number of remaining offset pulses for the assigned axes. Please refer to the figure below for a definition of the remaining offset pulse value.		<input type="radio"/>	<input type="radio"/>
	i8094H_FIXED_MOVE	Command a point-to-point motion for several independent axes.		<input type="radio"/>	<input type="radio"/>
	i8094H_SET_PULSE	This function sets the pulse number for fixed pulse driving.		<input type="radio"/>	<input type="radio"/>
	i8094H_CONTINUE_MOVE	This function issues a continuous motion command for several independent axes.		<input type="radio"/>	<input type="radio"/>

7.6.6 Interpolation Functions










Icon	Function Name	Statement	IT	MP	ISR
	i8094H_AXIS_ASSIGN	This function assigns the axes to be used for interpolation. Either two or three axes can be assigned using this function. Interpolation commands will refer to the assigned axes to construct a working coordinate system. The X axis does not necessarily have to be the first axis. However, it is easier to use the X axis as the first axis, the Y axis as the second axis, and the Z axis as the third axis.		<input type="radio"/>	<input type="radio"/>
	i8094H_VECTOR_SPEED	This function assigns the mode of interpolation. Either two or three axes will join this interpolation. Each interpolation mode will refer to some assigned axes that construct a working coordinate system. The assigned axes are defined by i8094H_AXIS_ASSIGN() function. The X axis does not necessarily have to be the first axis. However, it is easier to let the X axis as the first axis, the Y axis as the second axis, and the Z axis as the third axis in applications. Different modes need different settings.		<input type="radio"/>	<input type="radio"/>


		Please refer to the mode definitions.			
	i8094H_SET_VSV	This function sets the starting speed of the principle axis (axis 1) for the interpolation motion.		⊙	⊙
	i8094H_SET_VV	This function sets the vector speed of the interpolation motion. Users do not need to assign any axes on this function. The speed setting will take effect on the current working coordinate system which is defined by the i8094H_AXIS_ASSIGN() function.		⊙	⊙
	i8094H_SET_VA	This function sets the vector acceleration for interpolation motion. Users do not have to assign any axes on this function. This speed setting will take effect on the current working coordinate system which is defined by the i8094H_AXIS_ASSIGN() function.		⊙	⊙
	i8094H_SET_VD	This function sets the deceleration value for the interpolation motion.		⊙	⊙
	i8094H_SET_VK	Set the acceleration rate (jerk) value for interpolation motion.		⊙	⊙
	i8094H_SET_VL	This function sets the deceleration rate of the interpolation motion.		⊙	⊙
	i8094H_SET_VAO	Set this value will cause the motion control chip to start deceleration earlier. The remaining offset pulses will be completed at low speed to allow the controller to stop immediately when the offset pulse value has been reached. Please refer to the figure below for more information.		⊙	⊙
	i8094H_LINE_2D	This function executes a 2-axis linear interpolation motion.		⊙	⊙
	i8094H_LINE_3D	This function executes a 3-axis linear interpolation motion.		⊙	⊙
	i8094H_ARC_CW	This function executes a 2-axis circular interpolation motion in a clockwise (CW) direction.		⊙	⊙
	i8094H_ARC_CCW	This function executes a 2-axis circular interpolation motion in a counter-clockwise (CCW) direction.		⊙	⊙
	i8094H_CIRCLE_CW	This function executes a 2-axis circular interpolation motion in a clockwise (CW) direction.		⊙	⊙
	i8094H_CIRCLE_CCW	This function executes a 2-axis circular interpolation motion in a counter-clockwise (CCW) direction.		⊙	⊙

7.6.7 Synchronous Action Functions





Icon	Function Name	Statement	IT	MP	ISR
	i8094H_SYNC_ACTION	This function sets the activation factors (provocative) and the specified action when a specified activation factor occurs.		<input type="radio"/>	<input type="radio"/>
	i8094H_SET_COMPARE	This function sets the values of COMPARE registers. However, it will disable the functions of software limits.		<input type="radio"/>	<input type="radio"/>
	i8094H_GET_LATCH	This function gets the values from the LATCH register.		<input type="radio"/>	<input type="radio"/>
	i8094H_SET_PRESET	This function sets the PRESET value for synchronous action.		<input type="radio"/>	<input type="radio"/>
	i8094H_SET_OUT	This function sets the pulse by collocated the i8094H_SYNC_ACTION function.		<input type="radio"/>	<input type="radio"/>

7.6.8 Continuous Interpolation Functions









Icon	Function Name	Statement	IT	MP	ISR
	i8094H_RECTANGLE	Continuous interpolation will be performed to create a rectangular motion, which is formed by 4 lines and 4 arcs. The length of each side can be changed. The radius of each arc is the same and it can also be changed. The deceleration point will be calculated automatically. This is a command macro command that appears in various motion applications.		<input type="radio"/>	
	i8094H_LINE_2D_INITIAL	This function sets the necessary parameters for a 2-axis continuous linear interpolation using symmetric T-curve speed profile.		<input type="radio"/>	
	i8094H_LINE_2D_CONTINUE	This function executes a 2-axis continuous linear interpolation.		<input type="radio"/>	
	i8094H_LINE_3D_INITIAL	This function sets the necessary parameters for a 3-axis continuous linear interpolation using symmetric T-curve speed profile.		<input type="radio"/>	
	i8094H_LINE_3D_CONTINUE	This function executes a 3-axis continuous linear interpolation.		<input type="radio"/>	
	i8094H_MIX_2D_INITIAL	This function does the initial settings for mixed linear and circular 2-axis motions in continuous interpolation.		<input type="radio"/>	
	i8094H_MIX_2D_CONTINUE	This function executes mixed linear and circular 2-axis motion in continuous interpolation.		<input type="radio"/>	
	i8094H_HELIX_3D	This function performs a 3-axis helical motion.		<input type="radio"/>	
	i8094H_RATIO_INITIAL	This function sets the Initial values for ratio motion (motion in ratio) using a symmetric T-curve speed profile.		<input type="radio"/>	

	i8094H_RATIO_2D	This function executes a two-axis ratio motion.		<input type="radio"/>	
---	-----------------	---	--	-----------------------	--

7.6.9 Interrupt Control Functions













Icon	Function Name	Statement	IT	MP	ISR
	i8094H_ENABLE_INT	This function enables the interrupt.		<input type="radio"/>	
	i8094H_DISABLE_INT	This function disables the interrupt.		<input type="radio"/>	
	i8094H_INTFACTOR_ENABLE	This function sets the interrupt factors.		<input type="radio"/>	
	i8094H_INTFACTOR_DISABLE	This function disables the interrupt factors.		<input type="radio"/>	

7.6.10 Other Functions

Icon	Function Name	Statement	IT	MP	ISR
	i8094H_DRV_START	This command is usually used when users desire to start multi-axis driving simultaneously. When this command is issued, users may write other driving commands to the control card. All the driving commands will be held after i8094H_DRV_HOLD() is issued, and these commands will be started once the i8094H_DRV_START() is issued. However, if in driving, this command will not cause the driving to be stopped. But the next command will be held.		<input type="radio"/>	<input type="radio"/>
	i8094H_DRV_HOLD	This command releases the holding status, and start the driving of the assigned axes immediately.		<input type="radio"/>	<input type="radio"/>
	i8094H_STOP_SUDDENLY	This function immediately stops the assigned axes.		<input type="radio"/>	
	i8094H_STOP_SLOWLY	This function decelerates and finally stops the assigned axes slowly.		<input type="radio"/>	
	i8094H_VSTOP_SUDDENLY	This function stops interpolation motion of the assigned module immediately.		<input type="radio"/>	
	i8094H_VSTOP_SLOWLY	This function stops interpolation motion of the assigned module in a decelerating way.		<input type="radio"/>	
	i8094H_CLEAR_STOP	After using anyone of the stop functions, please solve the malfunction, and then issue this function to clear the stop status.		<input type="radio"/>	
	i8094H_INTP_END	1. If the current motion status is running a interpolation motion and you would like to issue a single axis motion or change the coordinate definition,		<input type="radio"/>	

		<p>you should call this function before the new command is issued.</p> <p>2. You can redefine the MAX V for each axis. In this way, you do not have to execute i8094H_INTP_END() function.</p>			
--	--	--	--	--	--

7.6.11 Macro Program Functions

Icon	Function Name	Statement	IT	MP	ISR
	i8094H_MP_CALL	This function sets the number of the macro program for jump executing in the next procedure layer.		⊙	
	i8094H_MP_SET_VAR	This function sets VARn to be the global value.		⊙	
	i8094H_MP_SET_RVAR	This function sets VARn to be the global return value.		⊙	
	i8094H_MP_VAR_CALCULATE	This function issues the “addition”, “subtraction” “multiplication”, and the “division” for supporting the variable arithmetic operation. For example: varNo (+-*/) varNo1 = varNo2.		⊙	
	i8094H_MP_FOR	This function issues the “for loop” condition statement.		⊙	
	i8094H_MP_NEXT	This function issues the “end of for loop” condition statement.		⊙	
	i8094H_MP_IF	This function issues the “If” condition statement.		⊙	
	i8094H_MP_ELSE	This function issues the “else” condition statement.		⊙	
	i8094H_MP_IF_END	This function issues the “end of If” condition statement.		⊙	
	i8094H_MP_TIMER	This function issues a procedure delay.		⊙	
	i8094H_MP_STOP_WAIT	This function can be used to assign commands to be performed while waiting for all motion to be completed (stopped).		⊙	
	i8094H_MP_SET_RINT	This function sets the i-8094H module to produce an interrupt signal with the 0x04 code to the WinCon controller.		⊙	