

EzProg-I EzTemplate tutorial Manual

(Version 1.1)

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 ICPDAS Product Support.

Email: Service@icpdas.com

TABLE OF CONTENTS

1	EZPROG-I	5
1.1	INTRODUCTION	5
1.2	LIBRARY AND UTILITIES	6
1.3	APPLICATION FRAMEWORK	9
1.4	IO AND REGISTER TABLE	11
1.5	EZPROG-I APPLICATION EXAMPLES	11
1.5.1	<i>EzTemplate introduction</i>	12
1.5.2	<i>EzTemplate HMI pages</i>	13
1.5.3	<i>EzTemplate definitions</i>	13
1.5.4	<i>EzProg-I function definitions</i>	15
1.5.5	<i>Starting EzProg-I engine</i>	17
2	EZTEMPLATE APPLICATIONS	19
2.1	TUTORIAL 1: USING EzCONFIG FOR IO REGISTER MAPPING	19
2.2	TUTORIAL 2: "HELLO WORLD"	20
2.2.1	<i>Using EzTemplate</i>	20
2.2.2	<i>Using EzHMI objects</i>	22
2.2.3	<i>Building and downloading an execution file</i>	24
2.2.3.1	Creating an execution file	24
2.2.3.2	To prepare Visual Studio for connecting	25
2.2.3.3	Connect to the target device	27
2.2.3.4	Download to the target device	28
2.3	TUTORIAL 3: CREATING A MULTI-PAGE USER INTERFACE	29
2.3.1	<i>Page setting</i>	29
2.3.2	<i>Page switching</i>	30
2.4	TUTORIAL 4: MULTI-LANGUAGE	34
2.4.1	<i>Page setting</i>	35
2.4.2	<i>EzHMI setting</i>	35
2.5	TUTORIAL 5: DIGITAL IO	40
2.5.1	<i>HMI design and IO register linking</i>	40
2.6	TUTORIAL 6: ANALOG IO	46
2.6.1	<i>Slot module configuration</i>	46
2.6.2	<i>EzHMI design and register settings</i>	48
2.7	TUTORIAL 7: USER THREAD	53
2.7.1	<i>EzHMI design and register settings</i>	53
2.7.2	<i>UserThread activation and code implementation</i>	57
2.8	TUTORIAL 8: RTSR	60
2.8.1	<i>EzHMI design and register settings</i>	60
2.8.2	<i>RTSR activation and code implementation</i>	62
2.9	TUTORIAL 9: TIMER	66
2.9.1	<i>EzHMI design and register settings</i>	66
2.9.2	<i>UserThread activation and code implementation</i>	72
2.10	TUTORIAL 10: COUNTER	76
2.10.1	<i>EzHMI design and register settings</i>	76
2.10.2	<i>UserThread activation and code implementation</i>	82
2.11	TUTORIAL 11: ADVANCED ENCRYPTION STANDARD (AES)	88
2.11.1	<i>EzHMI design and register settings</i>	88
2.11.2	<i>UserThread activation and code implementation</i>	92
2.11.3	<i>Registry key generation</i>	95
2.12	TUTORIAL 12: USING MULTILANGUAGE MESSAGE FILES	99
2.12.1	<i>EzHMI design and register settings</i>	99

2.12.2	<i>UserThread activation and code implementation</i>	104
2.12.3	<i>Create multi-language message files</i>	106
2.13	TUTORIAL 13: TEXT FONTS	109
2.13.1	<i>EzHMI design and register settings</i>	109
2.13.2	<i>RTSR activation and code implementation</i>	116
2.14	TUTORIAL 14: STEPS	120
2.14.1	<i>EzHMI design and register settings</i>	120
2.14.2	<i>UserThread activation and code implementation</i>	129

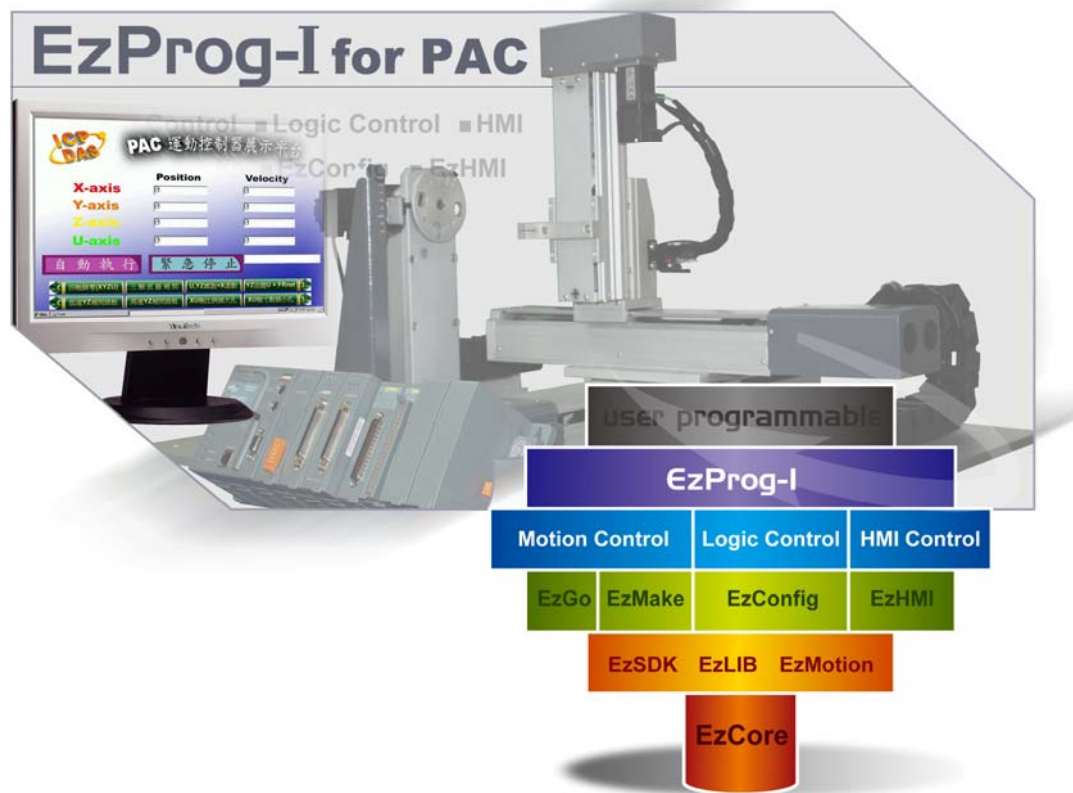
1 EzProg-I

1.1 Introduction

For the past few years the number of programmable automation controller (PAC) in the industry and their field of applications has grown rapidly. 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.

The main purpose of the EzCore engine is to assist the programmer to develop a logic control program on the evc platform. The EzCore comprises of numerous tools, libraries and HMI to assist the system developer to implement the control machine operation. EzHMI is a set of HMI ActiveX controls, which consist of graphic controls for the user interface.

EzConfig is a utility to read, set and test the system framework configuration. The framework comprises of IO, retentive and none retentive registers. The motion control part offers the EzGo and EzMake utility to assist the developer to set up a motion control system by enabling him to configure and to test run the servo motors. The EzCore allows the system engineer to focus his energy on designing the logic control system and thereby achieving greater results in a shorter period of time.

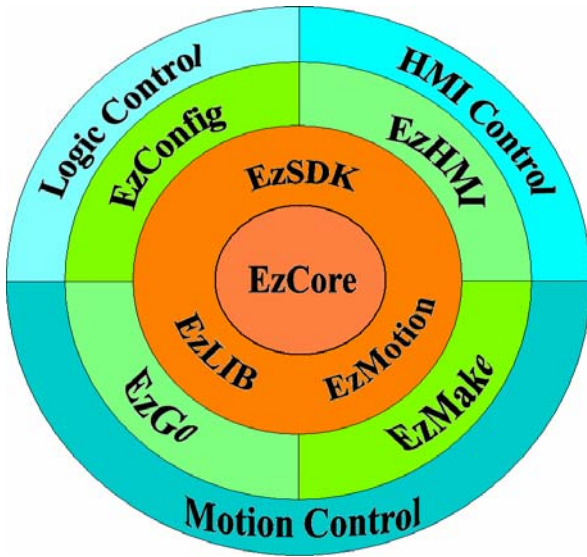


The EzProg-I is installed on the following directory on the Windows CE PAC:

- for WinCon W-8x8x-GM1: “\CompactFlash\EzProg-I”
- for MPac: “\System_Disk\EzProg-I”

1.2 Library and utilities

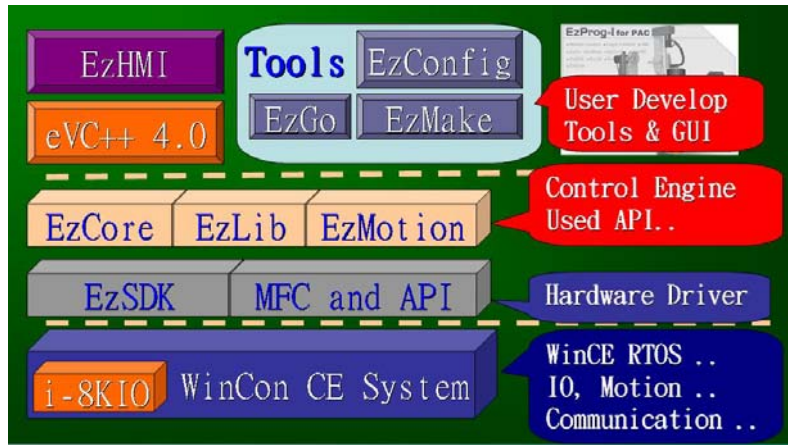
EzCore and VS2008:



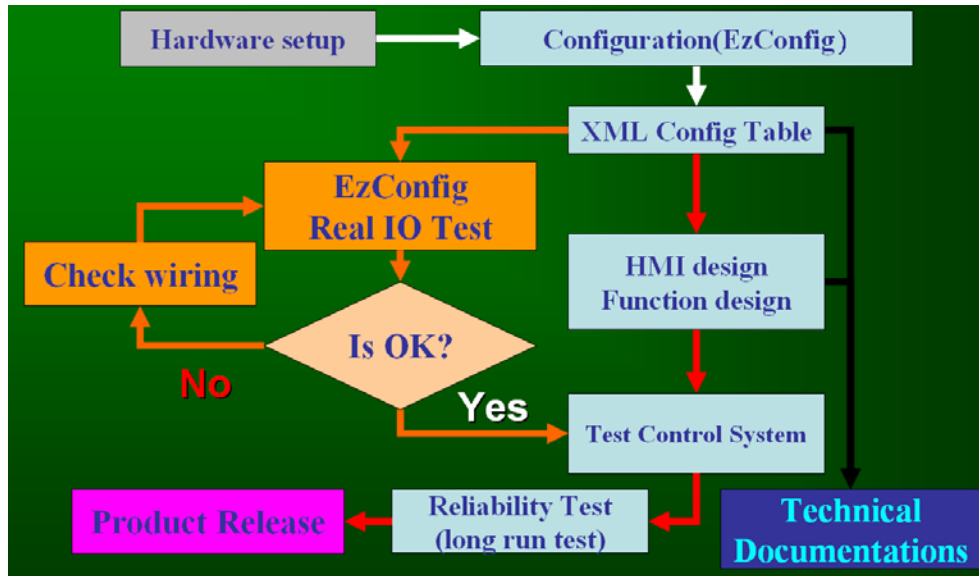
EzProg-I is a developing toolkit consisting of libraries, utilities and HMI controls for the Windows CE 6.0 platform. EzCore is the engine driving EzProg-I and its APIs are responsible for the communication between the hardware (IO), HMI controls and the registers. All the libraries are written for the VS2008 environment.

The basic development 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 composed of EzCore, EzLib and EzMotion
- The top layer is made up of the control program created by VS2008, the human machine interface (EzHMI), the configuration and testing utilities



Installing, configuration and testing the devices of the control system (IO, wiring, switches etc.) can be done without the actual control program. Therefore hardware and software implementation can be done independently from one another. This reduces development time and cost.

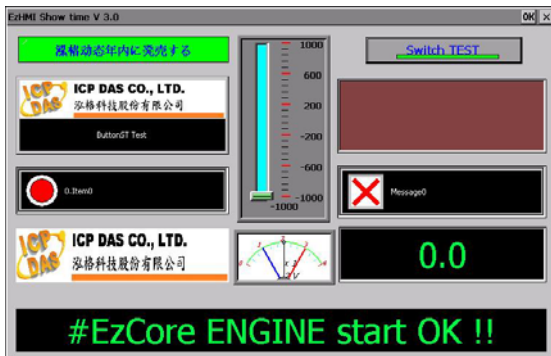


The main task of EzCore:

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. The hardware interrupt is supported by the following modules

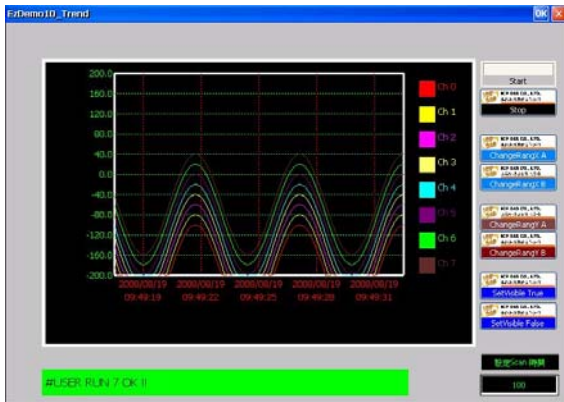
- DI module: 8094H in slot 1
- Motion modules: 8094A, 8094F, 8092F (in slot 1 to 3)

EzHMI ActiveX controls:



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

EzLIB application library



- EzLIB offers easy to use applications
- Various kinds of conversion functions
- Date, time functions
- Drawing functions
- BMP save to file function
- TCP/IP functions
- FTP functions
- Trend diagram support

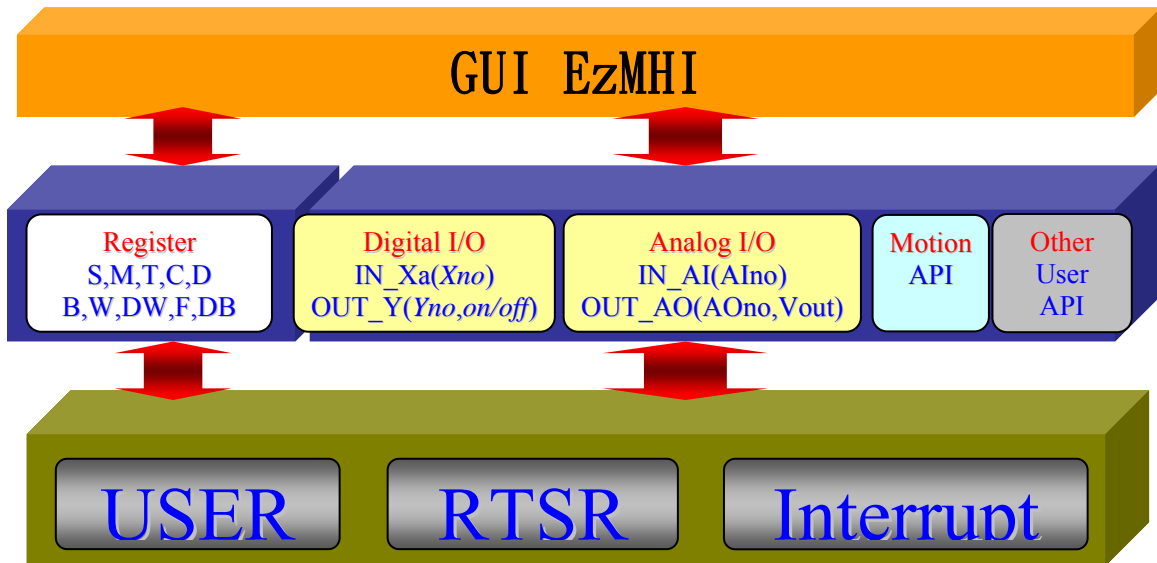
Servo motor control

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 motion manuals.

1.3 Application framework

The EzProg-I main framework structure has the following layout:



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
EzProg1 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 a value.
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 of function with lower priority levels will be executed.

1.4 IO and register table

For a detailed description of the EzProg-IO and register table please refer to the EzProg-I_Tool manual.

Currently the following the register types are defined:

	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	

The following registers are already reserved for multi-language and virtual keyboard support:

Register Type	Register Number	Note
D	8000	Multilanguage selection
M	16000	Enabling or disabling virtual keyboard support for the ColorEdit Control

1.5 EzProg-I application examples

To make full use of the functionalities of the EzProg-I, ICPDAS provides the template EzTemplate to demonstrate how to integrate EzProg-I in the VS2008 environment. The EzTemplate simplifies for the beginner the concept of EzProg-I and allows the implementation of a control program within a short time.

The following examples illustrate the usage of the EzTemplate.

1.5.1 EzTemplate introduction

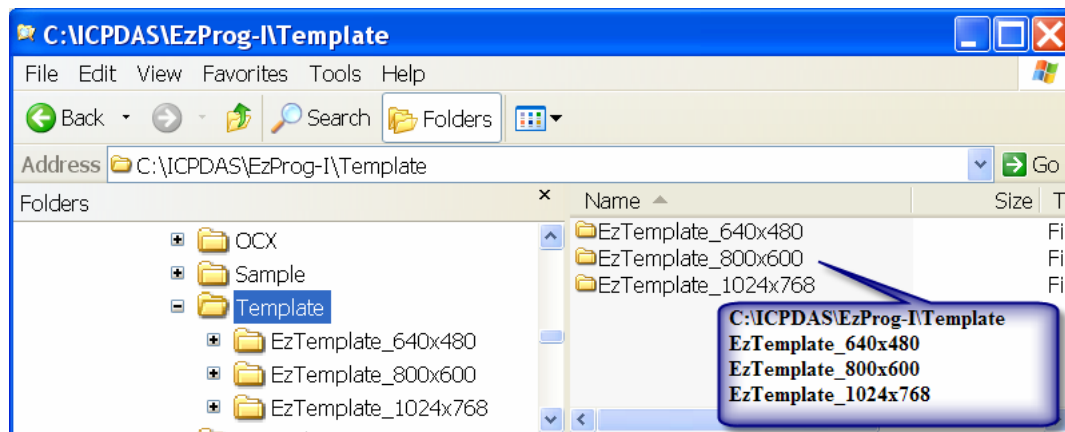
The EzTemplate provides the following default settings:

- All the basic EzProg-I settings are already done.
- The project includes 50 dialog pages which can be used as HMI pages. The programmer can select any of these pages for his HMI configuration.
- ButtonST control supports the viewing of the different HMI dialog pages during runtime
- Programming interfaces for user thread and RTSR. Eight user threads and real time service routine (multitasking) are available to run the control program in a multitasking and deterministic environment.
- Dialog pages for the following resolution are provided:
 - 640x480
 - 800x600
 - 1024x760

The EzTemplate projects with the different dialog resolution are located on the following directory:

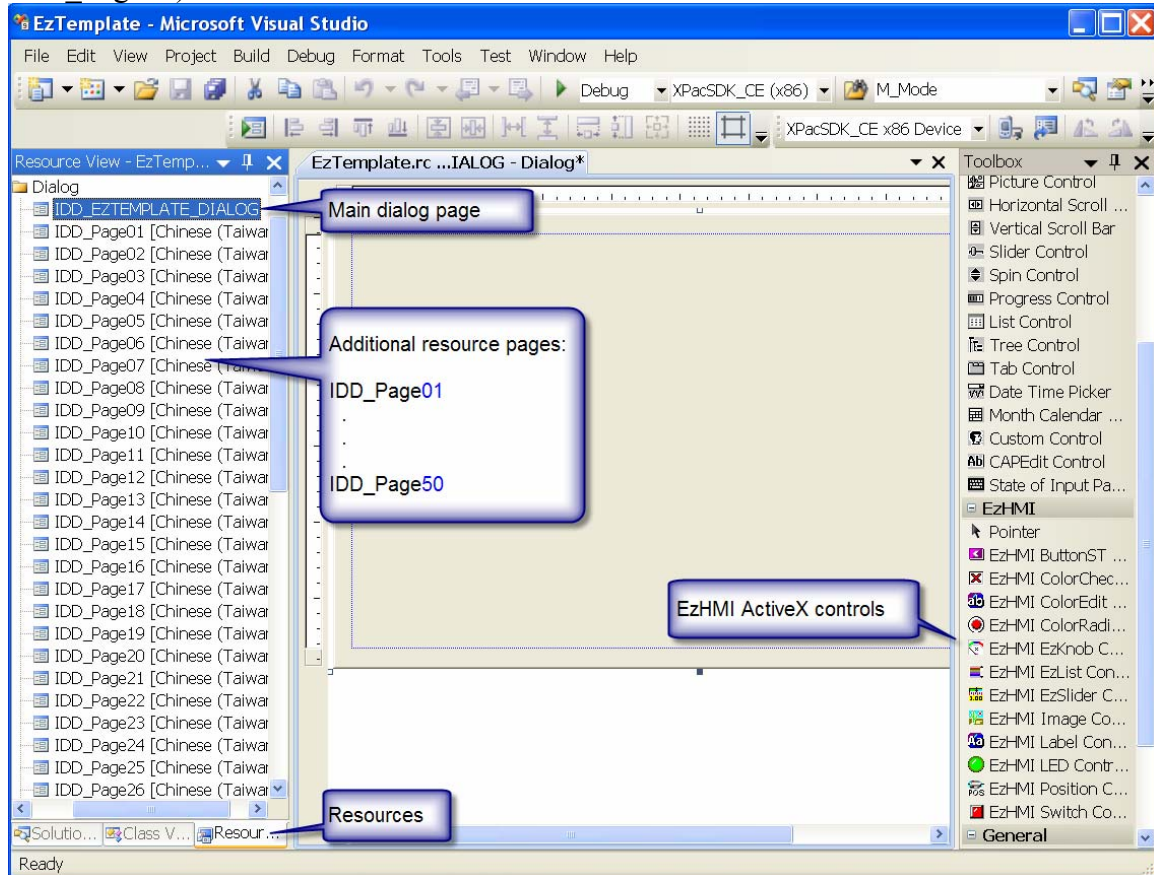
C:\ICPDAS\EzProg-I\Template

Select the project with the required resolution.



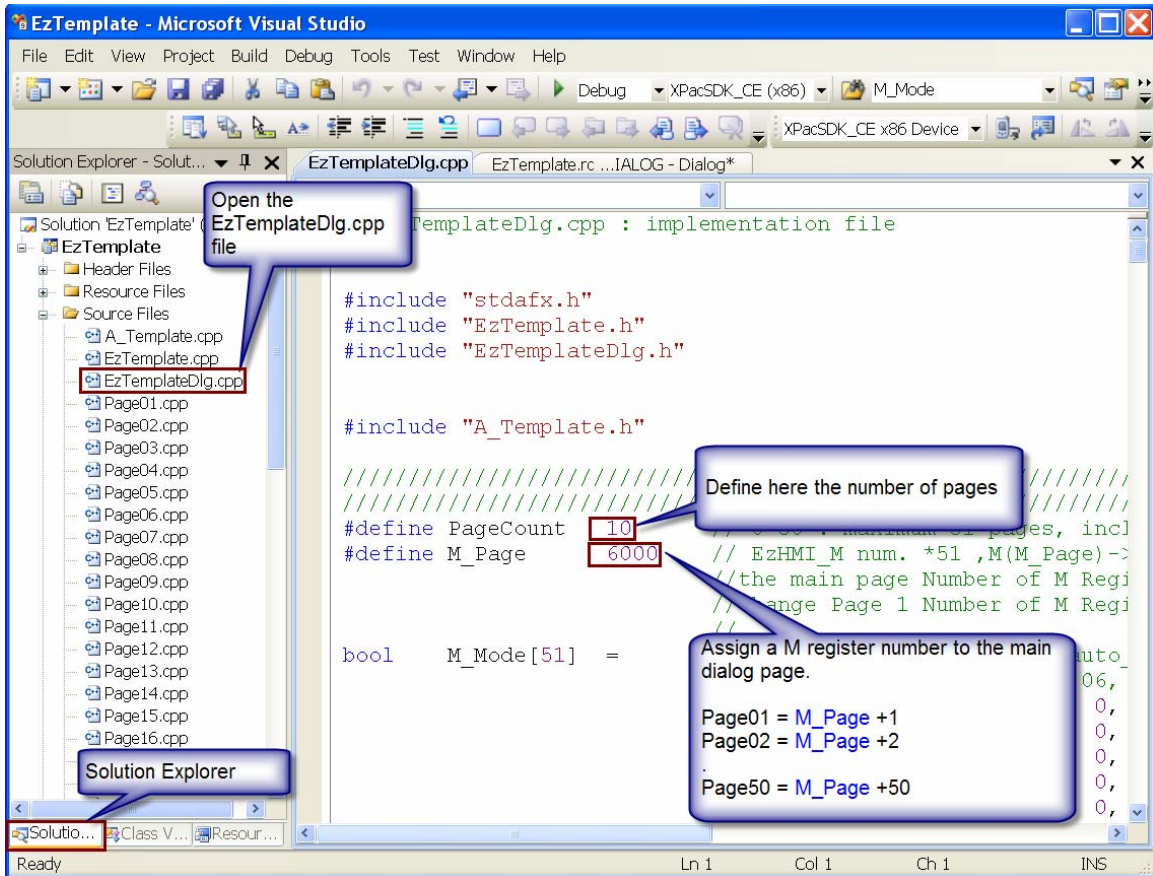
1.5.2 EzTemplate HMI pages

EzTemplate development framework consist of the main page (IDD_EZTEMPLATE_DIALOG) and 50 additional resource pages (IDD_Page01 to IDD_Page50)

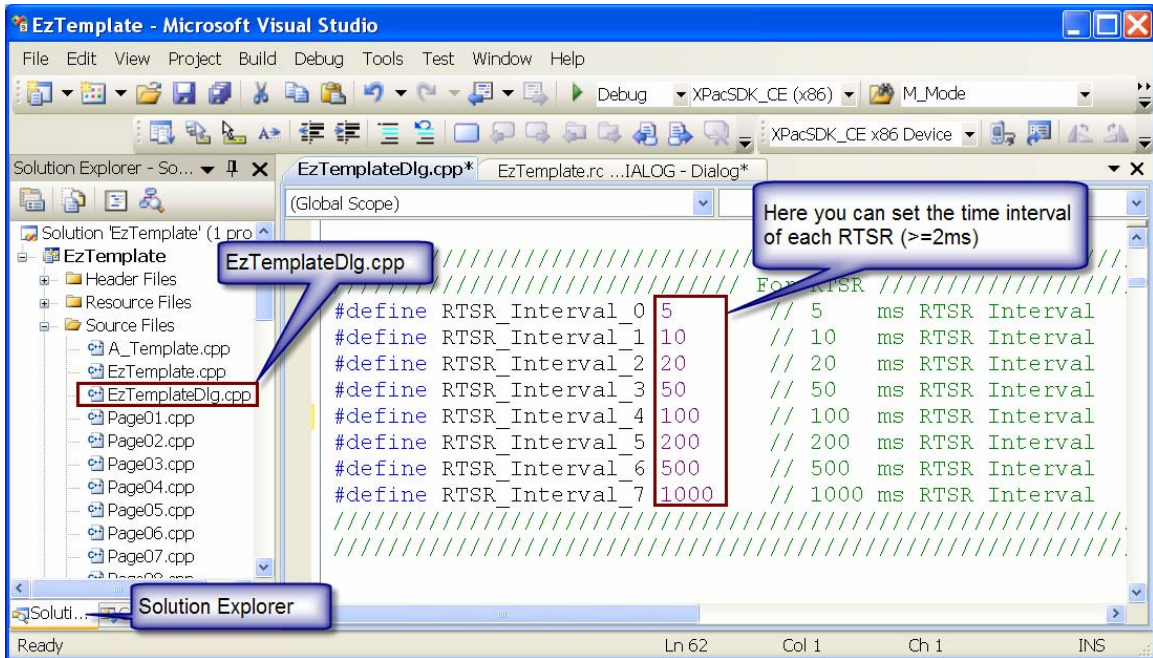


1.5.3 EzTemplate definitions

The page number setting and register number assignment. The number of dialog pages can be set by changing the **PageCount** value definition. The value 0 for **PageCount** indicates that only the main dialog window (IDD_EZTEMPLATE_DIALOG) is available as user interface. The value 1 (PageCount 1) allocates beside the main page another page (IDD_Page01). The program developer can now add EzHMI objects to the main page and the added page to create the user interface. During runtime the user can switch between different pages. The implementation necessary for page switching will be explained by an example in the next chapter. By default the PageCount value is set to 10, that means beside the main page ten additional dialog pages are available as user interface.



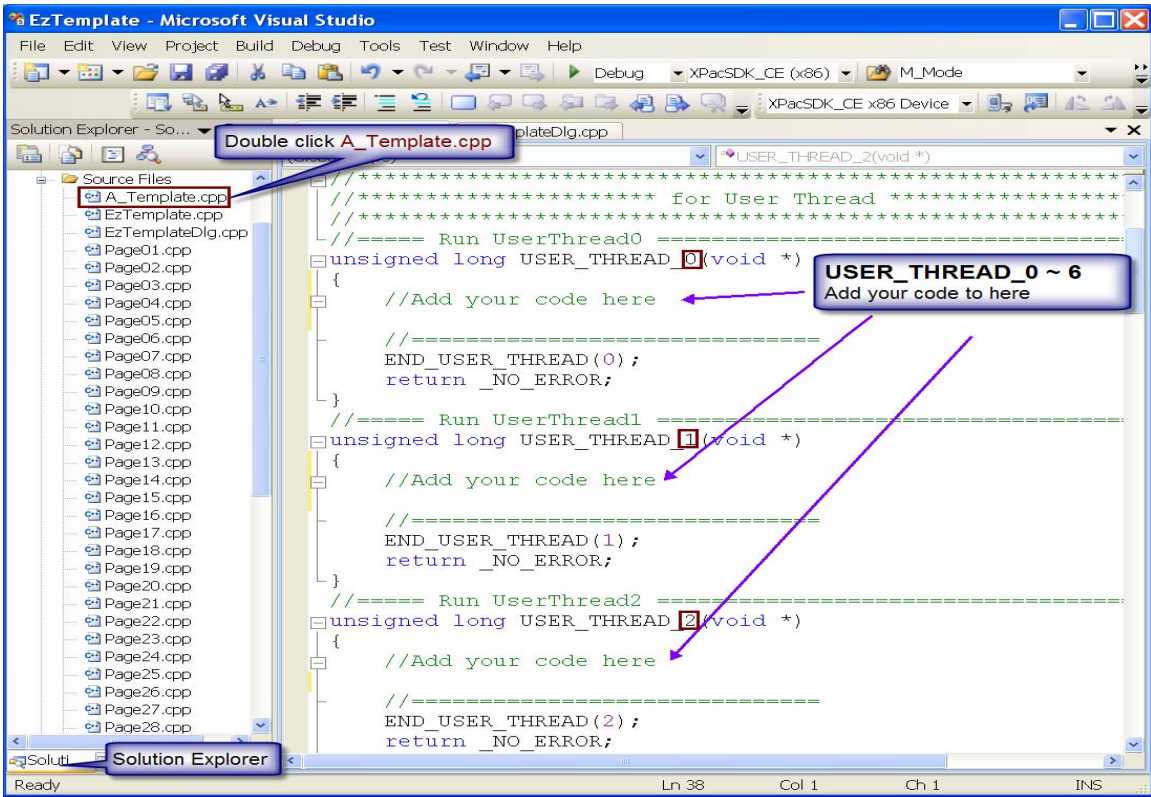
The following figure shows the default scanning interval settings of the eight real time service routine (RTSR). The programmer has to set the interval time according to his requirements. The smallest interval value supported is 2 milliseconds.



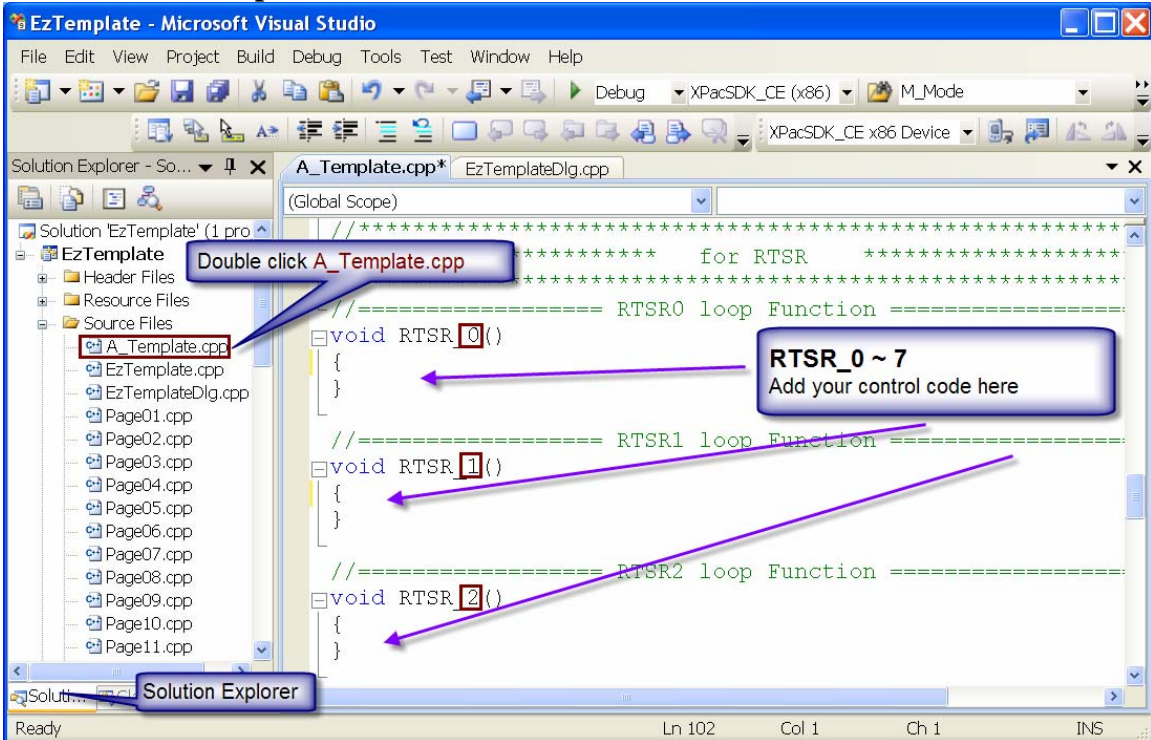
1.5.4 EzProg-I function definitions

The header and source file of f A_Template contains the function definition and implementing for the real time service routines (RTSR) and the user threads. The programmer only has to add his code to the function implementation in the A_Template.cpp file.

USER_THREAD function implementation:

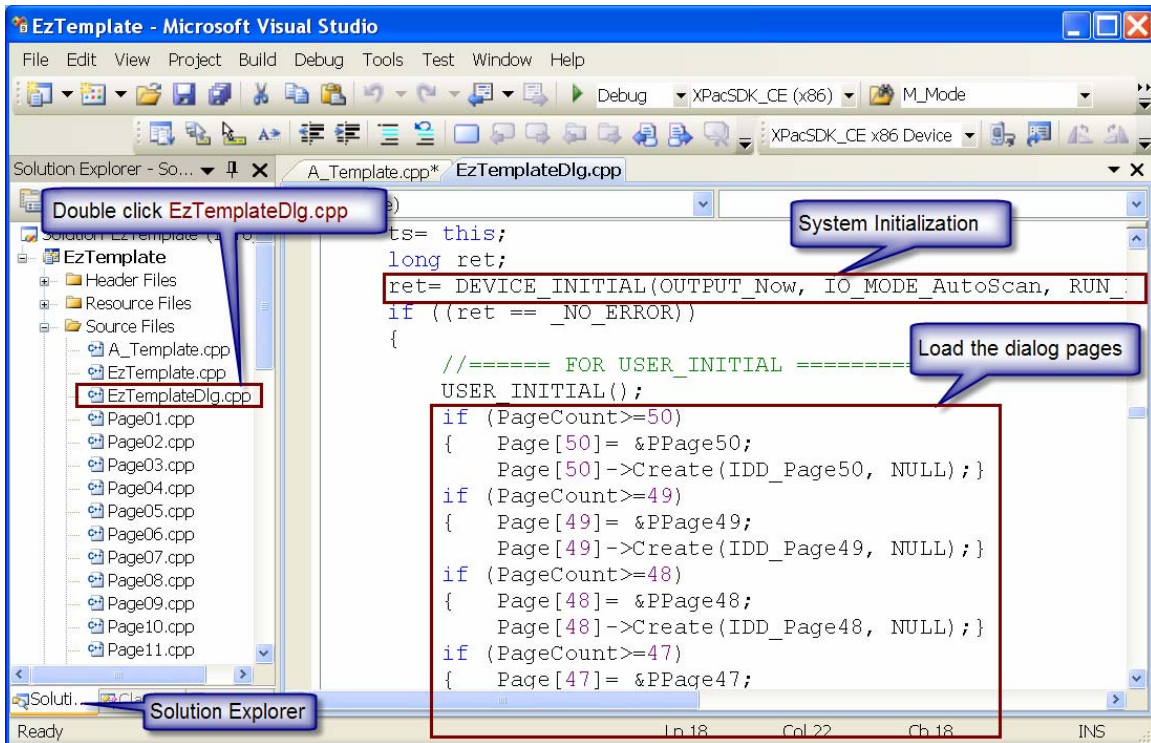


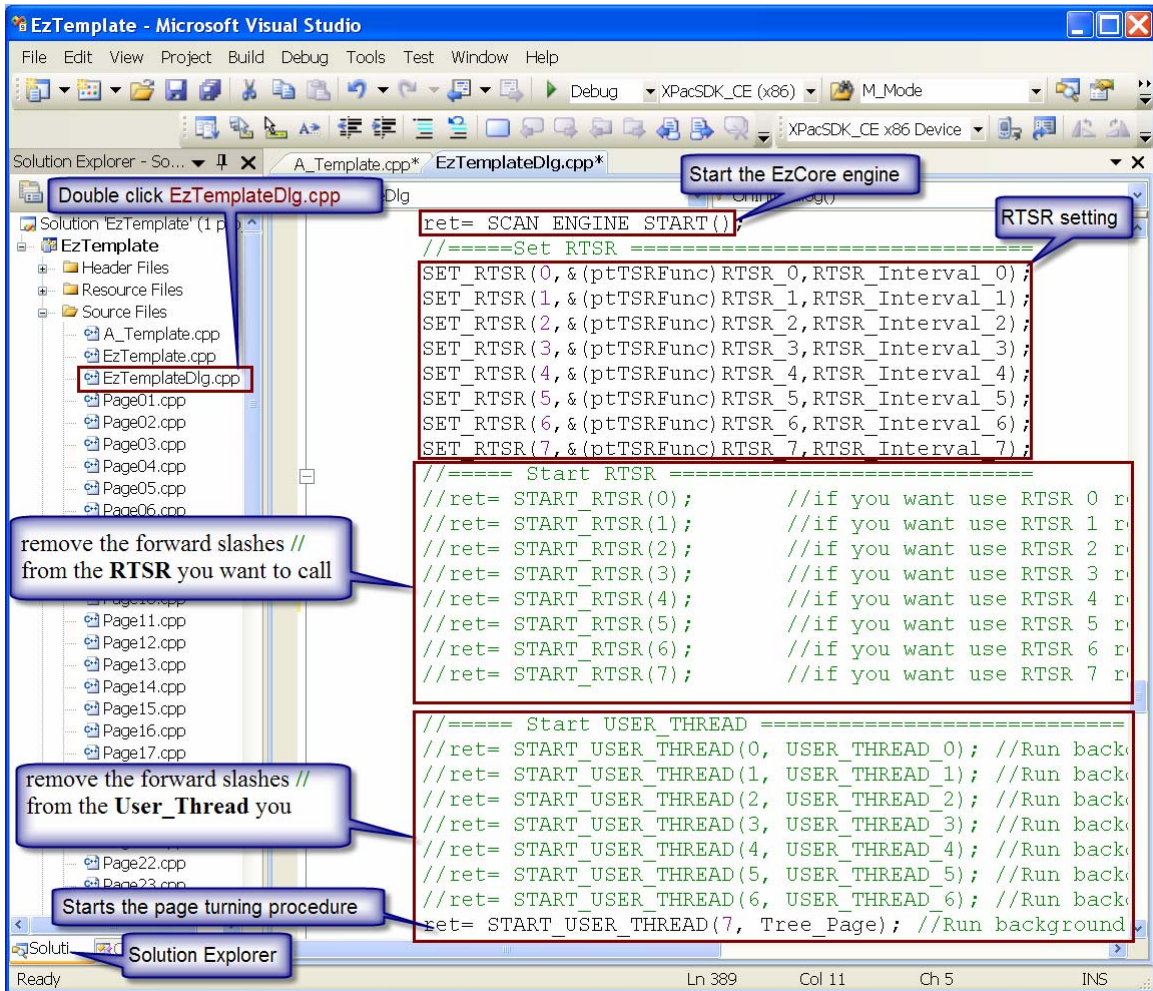
RTSR function implementation:



1.5.5 Starting EzProg-I engine

This example shows you the initialization, page loading and launching of scan engine RTSR and User_Thread.





2 EzTemplate Applications

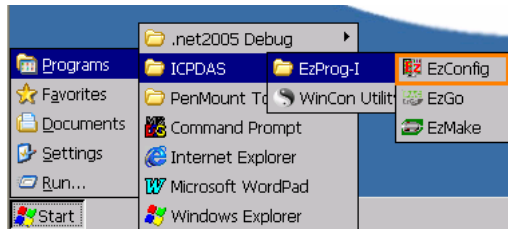
Before you start studying the examples of this chapter it is suggested that you consult the “EzProg-I Getting Started” manual to do all the necessary preparations. It is necessary to install the following software

- Visual Studio 2008
- Platform SDK
- EzProg-I development package

2.1 Tutorial 1: Using EzConfig for IO register mapping

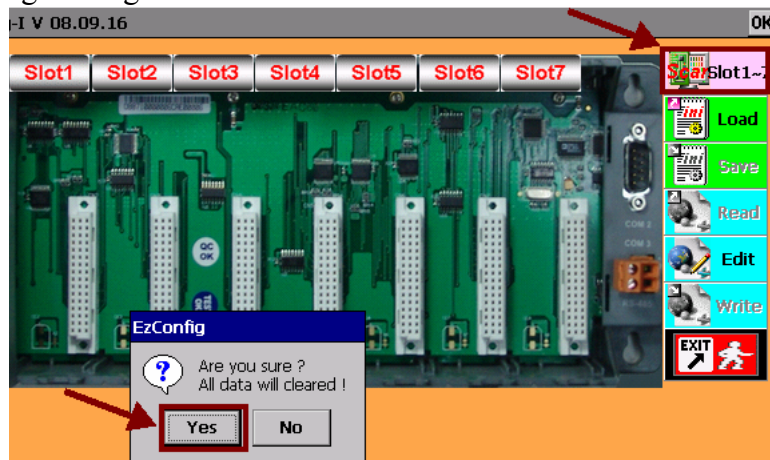
Start the the EzConfig utility on the Windows CE platform:

Start →Program→ICPDAS→EzProg-I→EzConfig



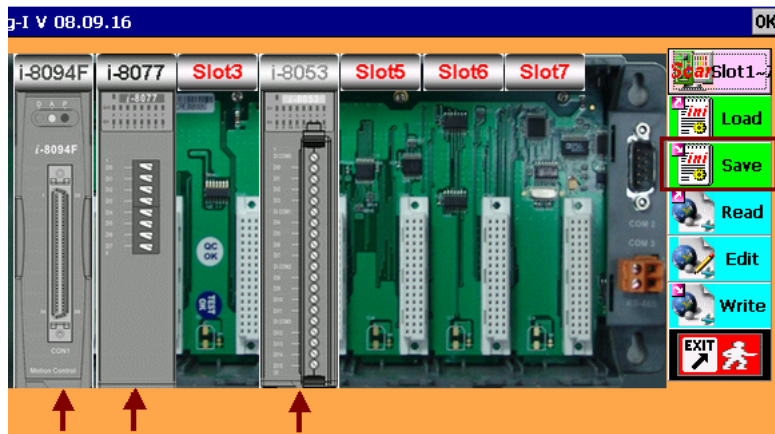
Scan all the slots to automatically detect the device module in each slot.

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



After a successful scan all the modules detected will be displayed with their names in their respective slots. In the background every IO channel of all the modules will be mapped to an IO register number.

Press “Save” to save the IO register mapping and “Exit” to close the program.



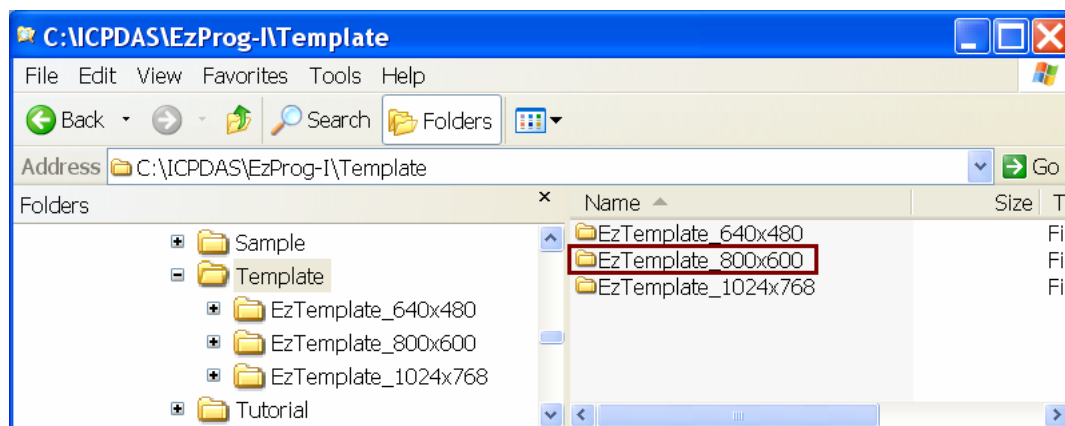
Notice: It is required to close the EzConfig utility in order for a program running on the EzCore engine to work properly.

2.2 Tutorial 2: “Hello World”

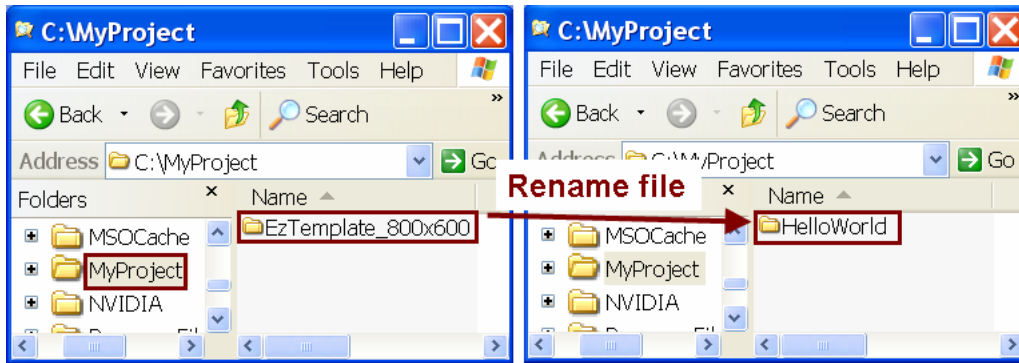
This example shows you how to use EzTemplate to create in a fast and easy way a program.

2.2.1 Using EzTemplate

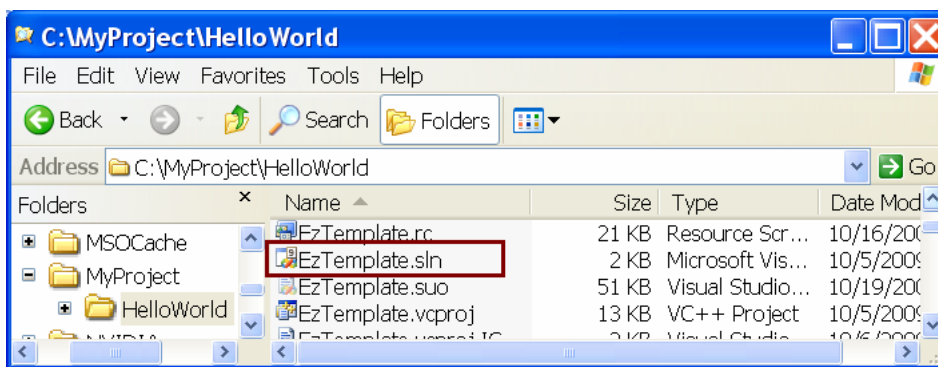
STEP 1: Copy the file from the directory *C:\ICPDAS\EzProg-I\Template\EzTemplate_800x600* to any directory of your choice. In this example the file is copied to the “MyProject” directory.



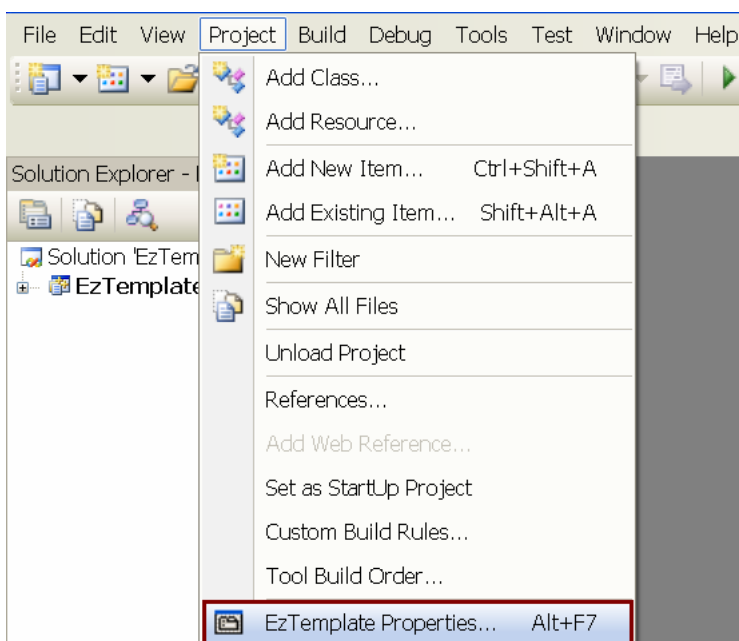
STEP 2: Rename the file “EzTemplate_800x600” to “HelloWorld”.



STEP 3: Open the project file with Visual Studio 2008 by double clicking the solution file “EzTemplate.sln”



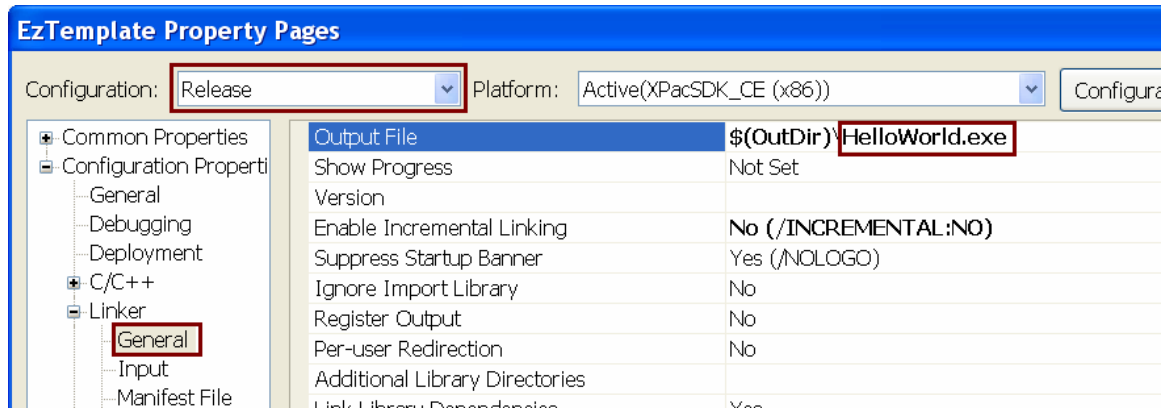
STEP 4: Rename the execution file.
Open the Property Page: **Project** → **EzTemplate Properties...**



Configuration Properties → Linker → General

Select the “Release” configuration and change the output file to “\$(OutDir)\HelloWorld.exe” as shown below and click “Apply”.

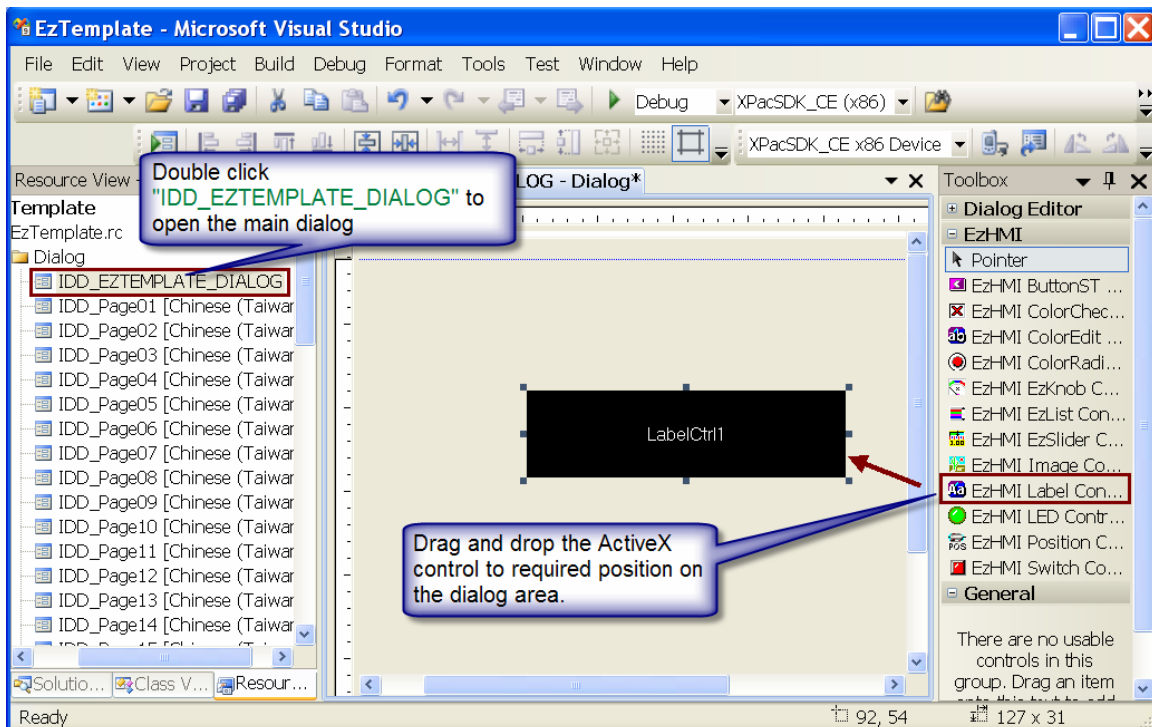
Select the “Debug” configuration and change the output file to “\$(OutDir)\HelloWorld.exe” and click “Apply”.



2.2.2 Using EzHMI objects

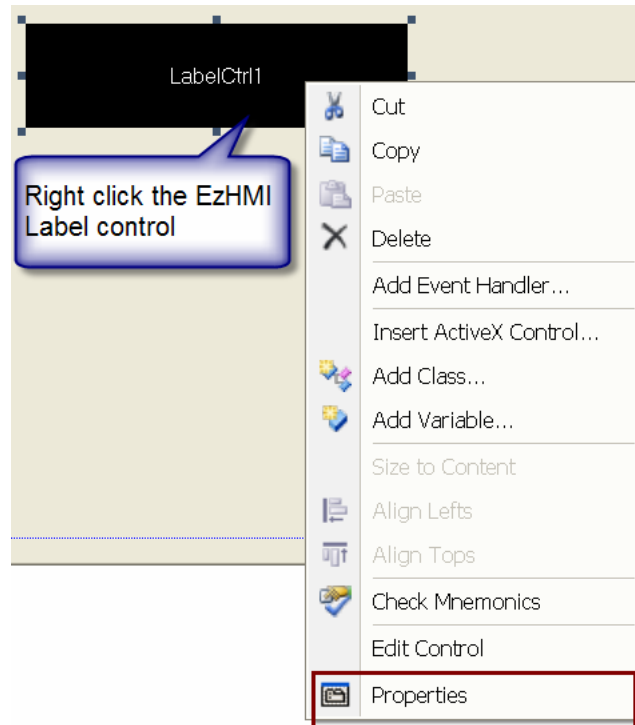
Add a new EzHMI Label control to the main dialog page

(IDD_EZTEMPLATE_DIALOG). The purpose of this program is to display the “Hello World!” text in the Label control.

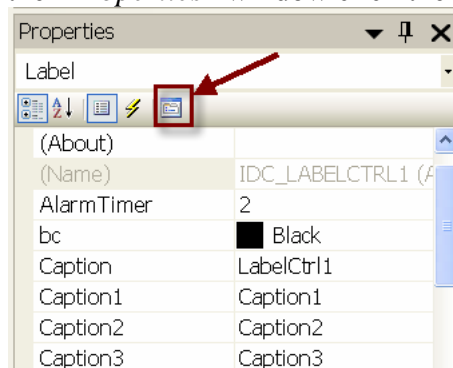


Open the property sheet of the EzHMI Label control to set its properties.

STEP 1: Right click the control on the main page and select “Properties”



STEP 2: On the “Properties” window click the “Properties Pages” icon.

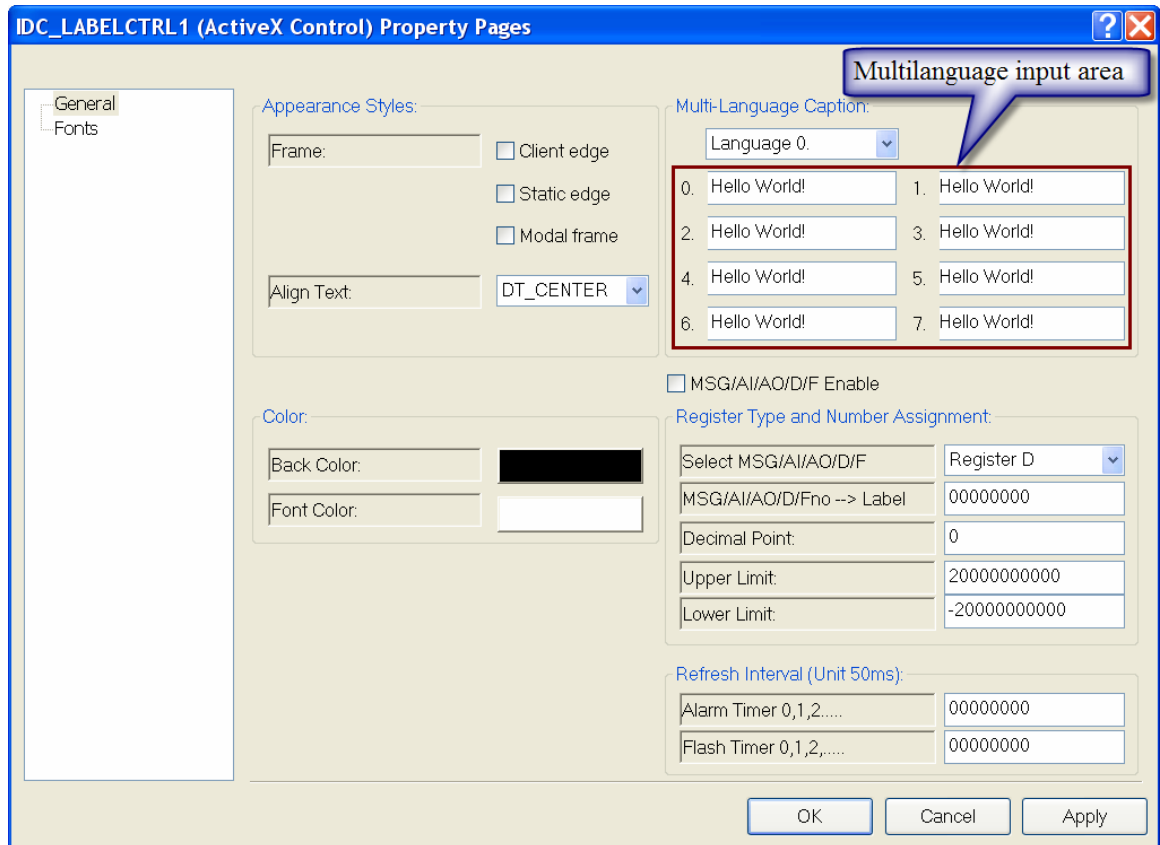


STEP 3: Now you can do the necessary configurations on the “Properties Pages”.

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

STEP 4: The eight edit boxes in the Multilanguage input area represents eight different languages. Enter in each edit box "Hello World!".

In this example the setting is language independent that means no matter in which language setting the program is running the label object will always display the same "Hello World!" string.



2.2.3 Building and downloading an execution file

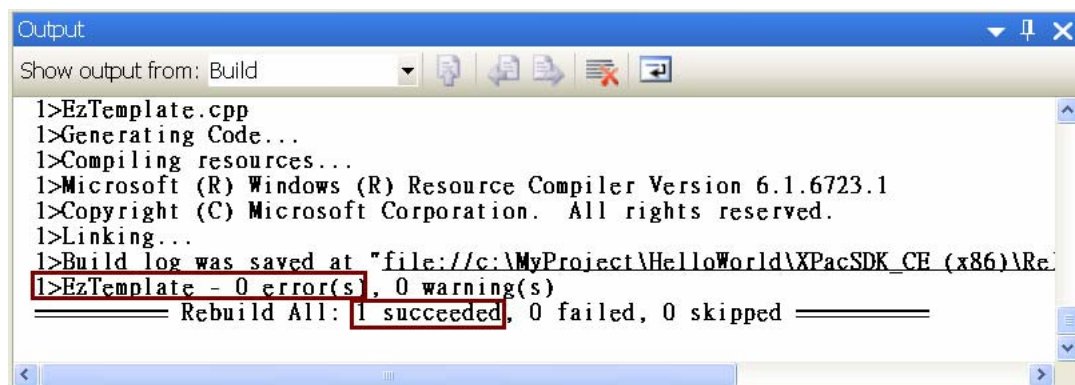
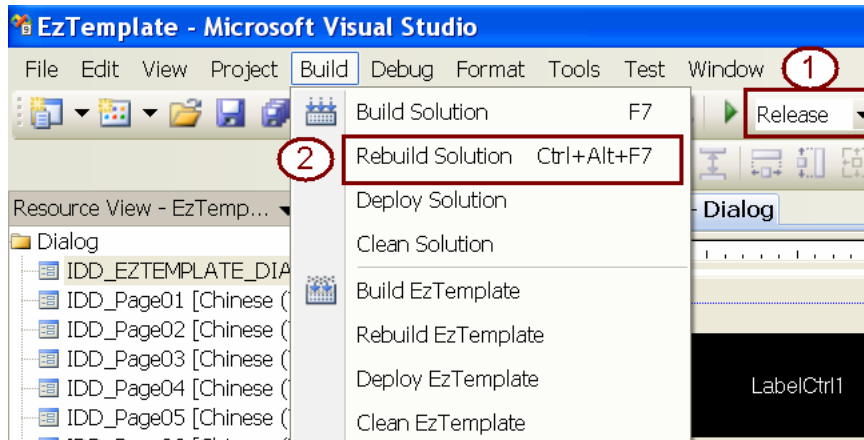
The following steps describe required to build an application and download the execution file to the PAC running with Window CE 6:

2.2.3.1 Creating an execution file

STEP 1: Set the compile configuration to release mode: Release (Nr1)

STEP 2: Compile the project: Build→Rebuild Solution

If the building of the execution file was successful the VS2008 output window will display: “0 error(s)”



2.2.3.2 To prepare Visual Studio for connecting

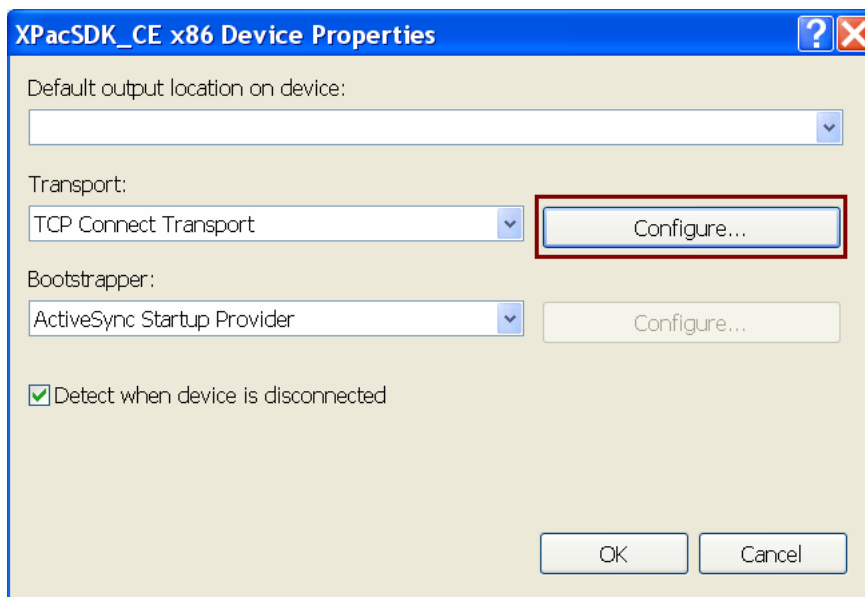
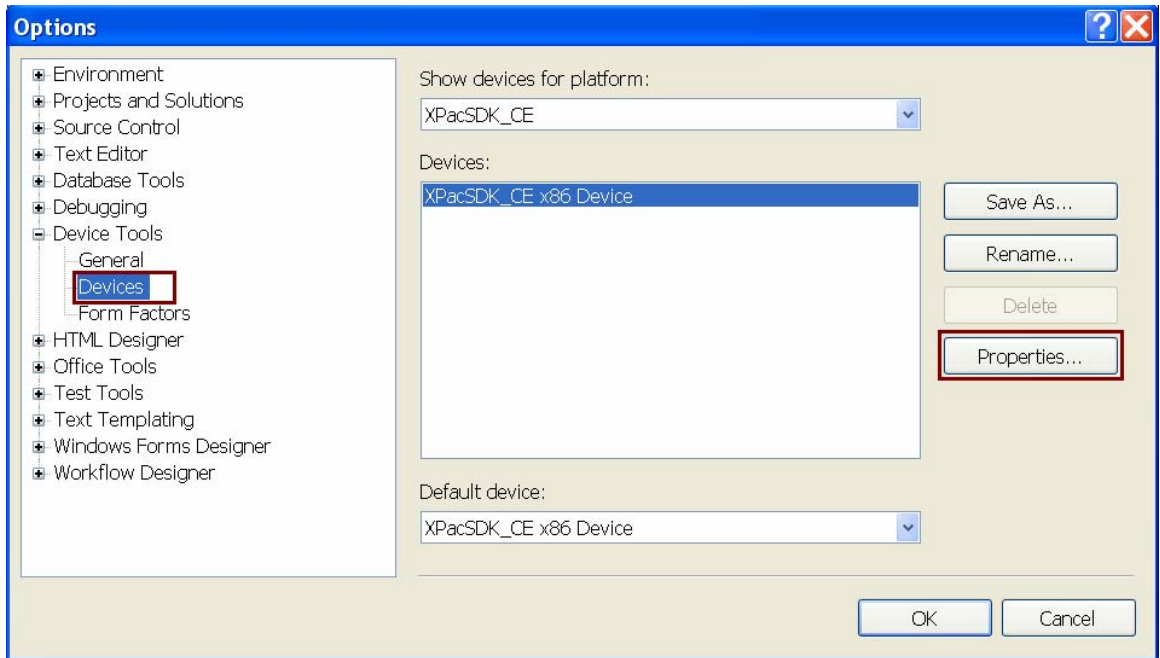
Before the execution file can be downloaded to the PAC you have to tell the Visual Studio 2008 the IP address of the target device.

STEP 1: Click Tools→Options...

STEP 2: In the "Option" window open the "Device Tools" tree and click "Device" branch.

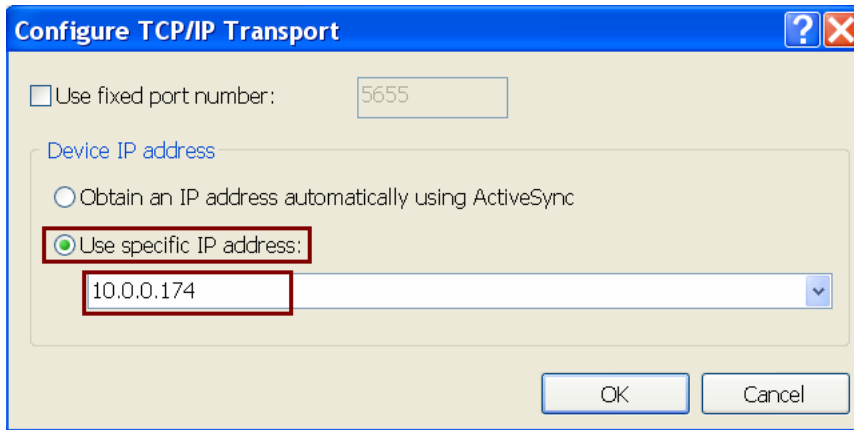
STEP 3: Select "XPacSDK_CE x86 Device" as your platform and click "Properties..." button.

STEP 4: Click the "Configure ..." button of the "XPacSDK_CE x86 Device Properties" window



STEP 5: Select the “Use specific IP address” option and enter the IP address of the remote PAC device.

STEP 6: Confirm your setting by clicking the “OK” button of each open dialog window.

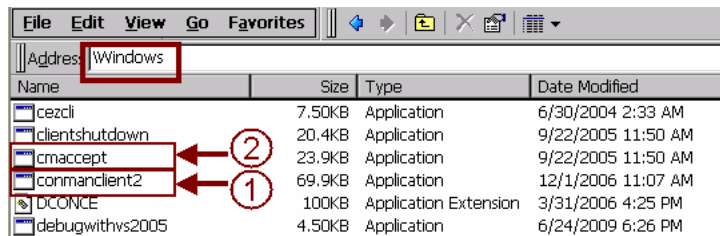


2.2.3.3 Connect to the target device

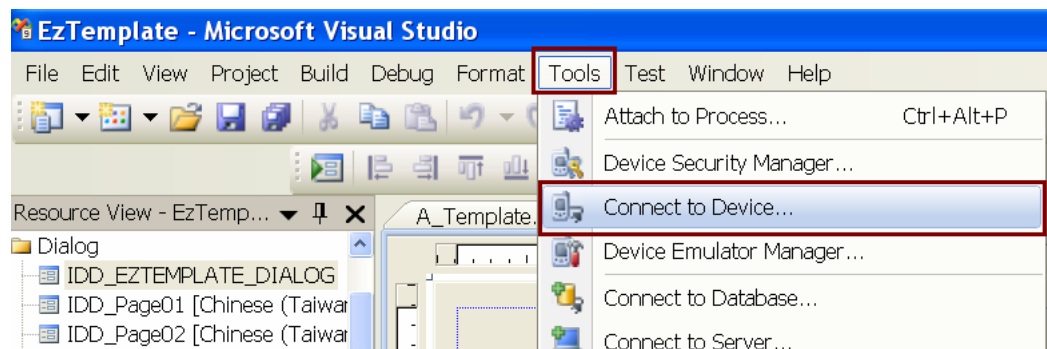
STEP 1: On the PAC device go to the **|Windows** directory and double click the following execution files in the shown sequence:

- *conmanclient2.exe*
- *cMaccept.exe*

Make sure that the *conmanclient2.exe* file is being opened first.



STEP 2: Connect the Visual Studio 2008 to the device: Tools→Connect to Device...

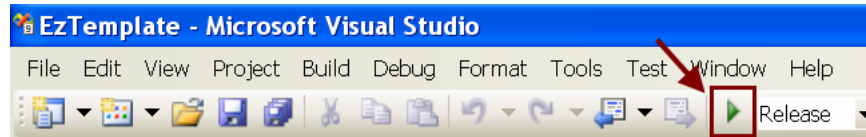


2.2.3.4 Download to the target device

The execution file of the compiled project is downloaded with VS2008 as follows:

STEP 1: Make sure the VS2008 is connected to the PAC device

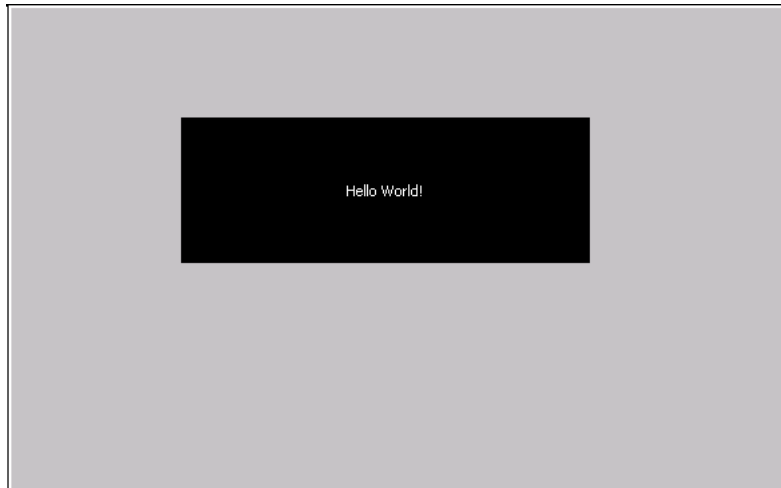
STEP 2: Click the green error icon as indicated on the figure.



STEP 3: The execution file will be downloaded to the following directory

\\Program Files\EzTemplate

After the download has completed the program is being started automatically by VS2008. In this example the EzHMI Label object shows the “Hello World!” text string.



2.3 Tutorial 3: Creating a multi-page user interface

This example demonstrates how to create with EzTemplate a program which provides three windows as user interfaces. In addition it explains how to switch between pages during runtime.

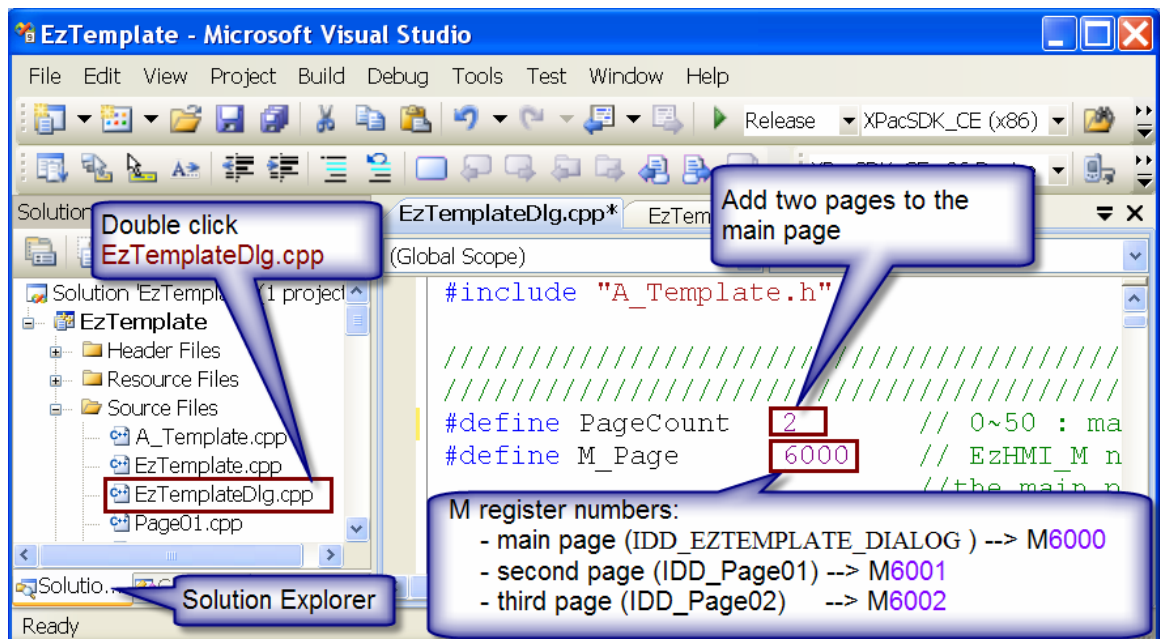
Use the EzTemplate project and change its execution file name to HMI_page as described in chapter “2.2.1 Using EzTemplate”

2.3.1 Page setting

The number of pages used for the project has to be set in the *EzTemplateDlg.cpp* file.

STEP 1: Open the *EzTemplateDlg.cpp* file

STEP 2: Every EzTemplate has got at least one page which is the main page. The main page is called `IDD_EZTEMPLATE_DIALOG` and can not be removed. The programmer has got the option to increase the number of page by changing the value of `PageCount`. The `PageCount` constant sets the number of pages in addition to the main page. Change the constant value to **two** so that the project has a total number of three pages. The main page (`IDD_EZTEMPLATE_DIALOG`) is accessed by the M register number 6000 and the page following by the M register number 6001 and 6002. Setting one of the registers to true will display the respective page.

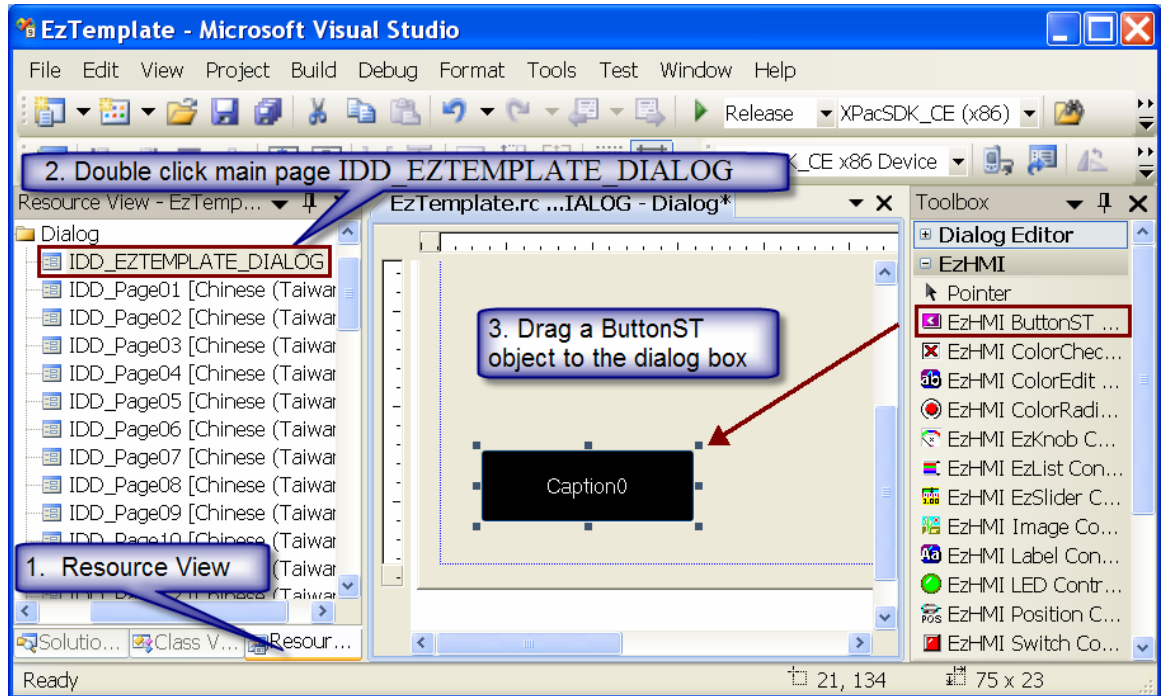


2.3.2 Page switching

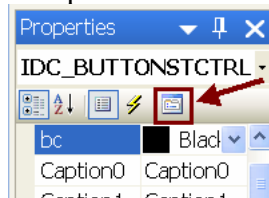
In the following two buttons are added to the main dialog page. The buttons are linked to the second (IDD_Page01) and third page (IDD_Page02). Clicking these buttons during runtime will display the corresponding page. Furthermore one button is added to the second and third page and linked to the main page to enable the user to return to the main page.

STEP 1: Open the IDD_EZTEMPLATE_DIALOG dialog

STEP 2: Add a EzHMI ButtonSt object to the dialog. This button will be used for switching between pages.



STEP 3: Open the property pages: Right click the button object and select “Properties”. Click the “Property Pages” icon in the “Properties” window

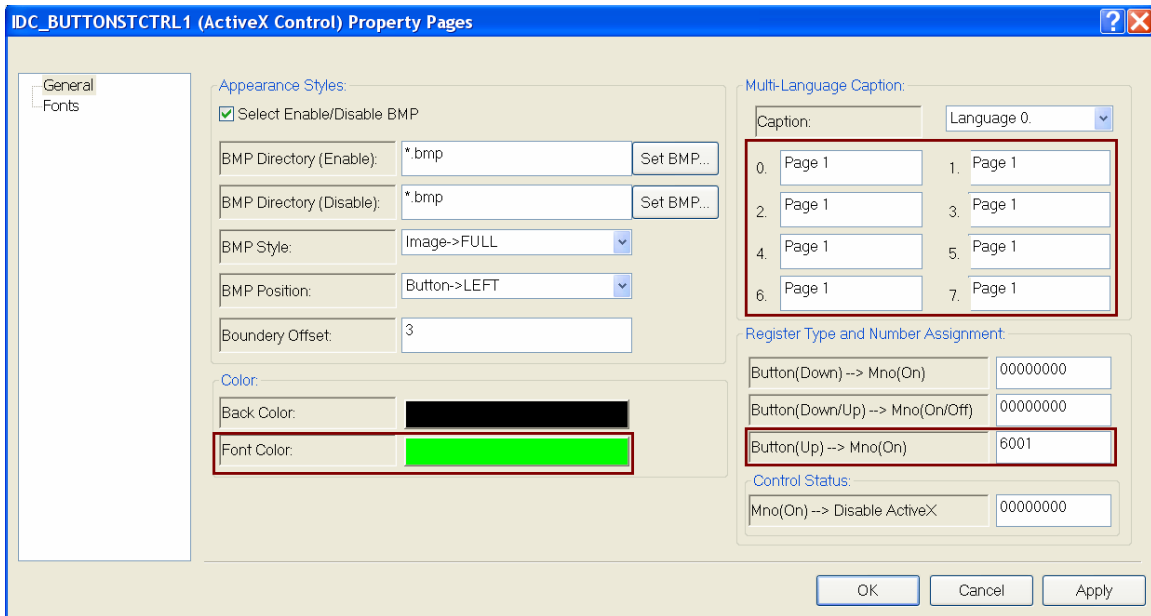


STEP 4: Set the button caption to “Page 1”.

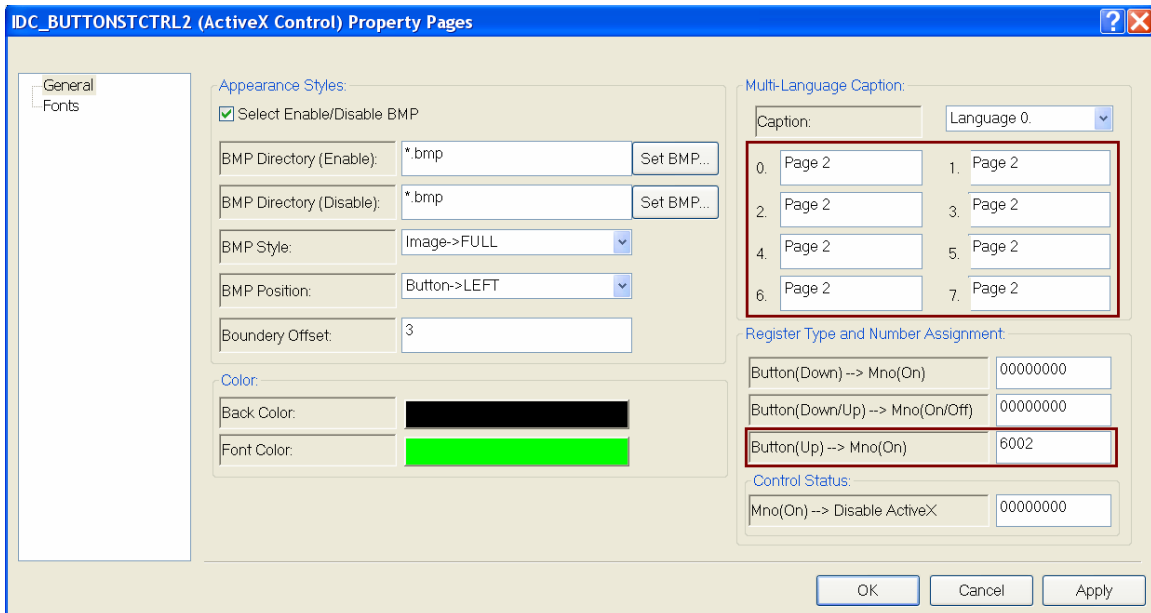
STEP 5: Enter the M register number 6001 of page 1 to “Button (Up)→Mno(On)”. After the button has been clicked and released by the mouse the register 6001 is set to true. This causes the page 1 to be displayed.

STEP 6: Set the caption text color to green

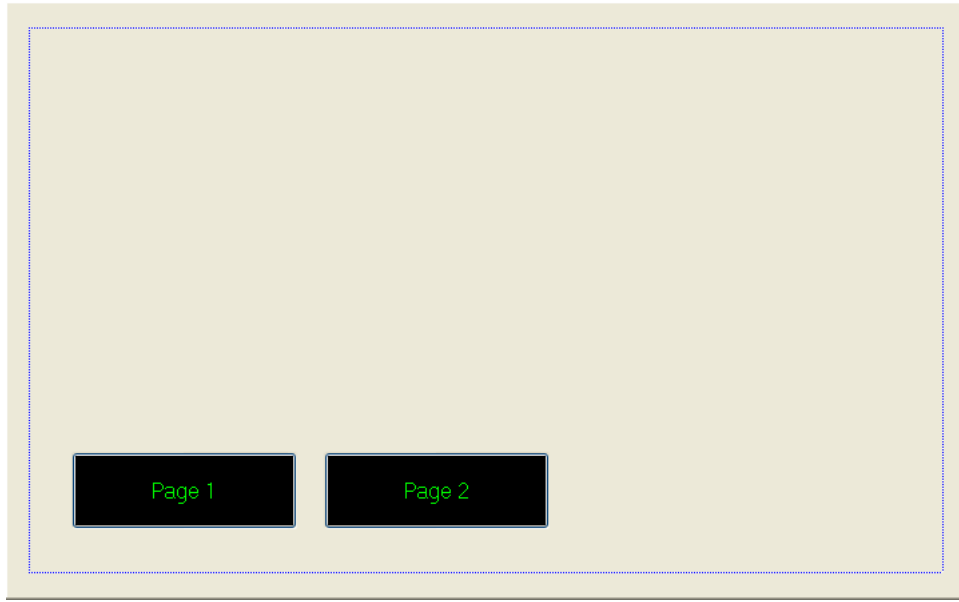
STEP 7: Click “OK”



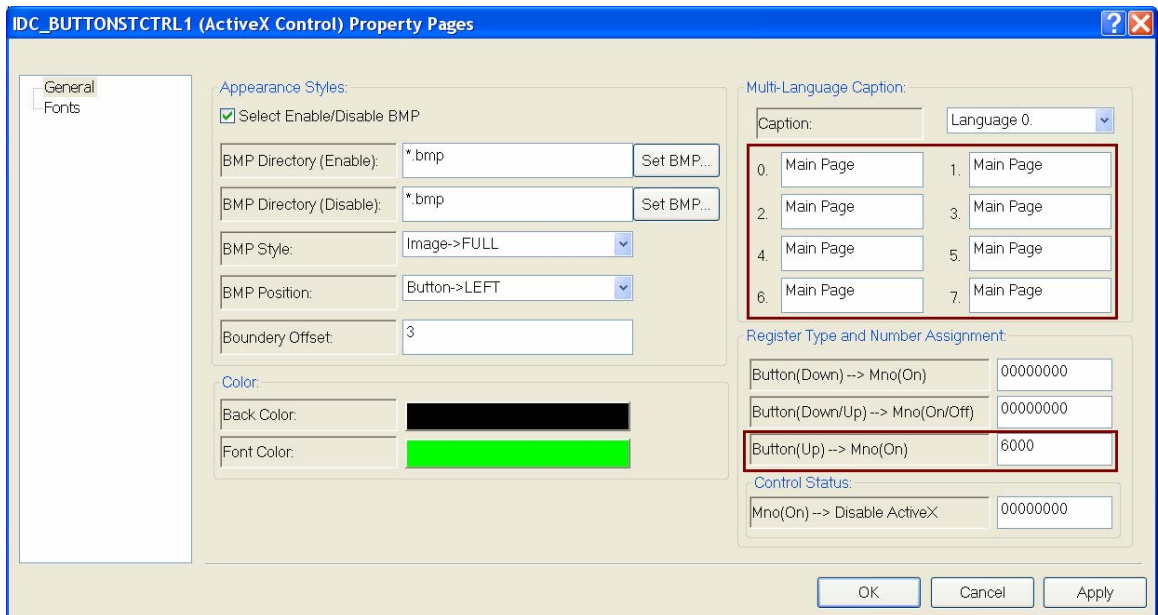
- STEP 8: Create a second button with identical properties by copying and pasting the first one. Open the property pages and set the following properties:
- STEP 9: Set the button caption to “Page 2”.
- STEP 10: Link the page 2 to the button: Enter for “Button (Up)→Mno(On)” the M register number 6002 of page 2 .
- STEP 11: Click “OK”



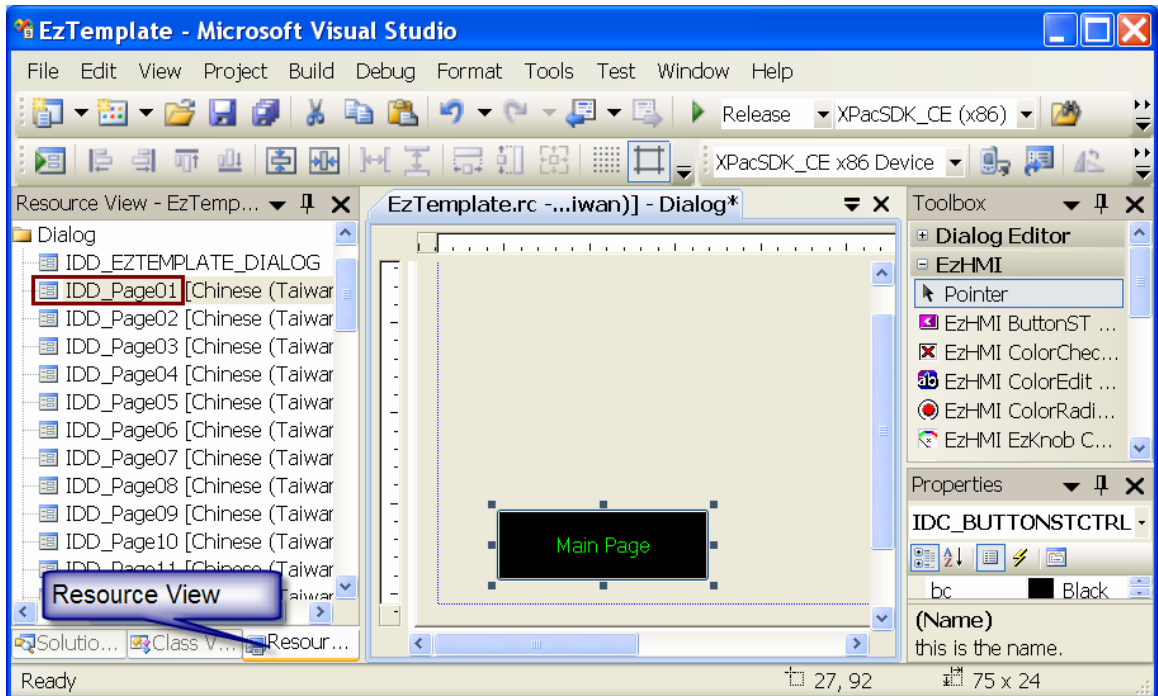
The appearances of the buttons on the main page are shown in the next figure:



- STEP 12: Copy the first button and paste it on page 1 (IDD_Page01). Open its property pages and set the following properties:
- STEP 13: Set the button caption to “Main Page”.
- STEP 14: Link the main page to the button: Enter for “Button (Up)→Mno(On)” the M register number 6000 of the main page .
- STEP 15: Click “OK”



The result is shown in the next figure:



- STEP 16: Copy the button on page 1 (IDD_Page01) and paste it on page 2 (IDD_Page02). The button properties do not have to be changed.
- STEP 17: Compile and download the project. All three pages can be accessed by clicking the corresponding buttons.

2.4 Tutorial 4: Multi-language

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 in Runtime. Texts such as caption text, label text and messages can be changed to a different language by simple entering the index to the D register number 8000. The following ActiveX controls support multi-language setting:

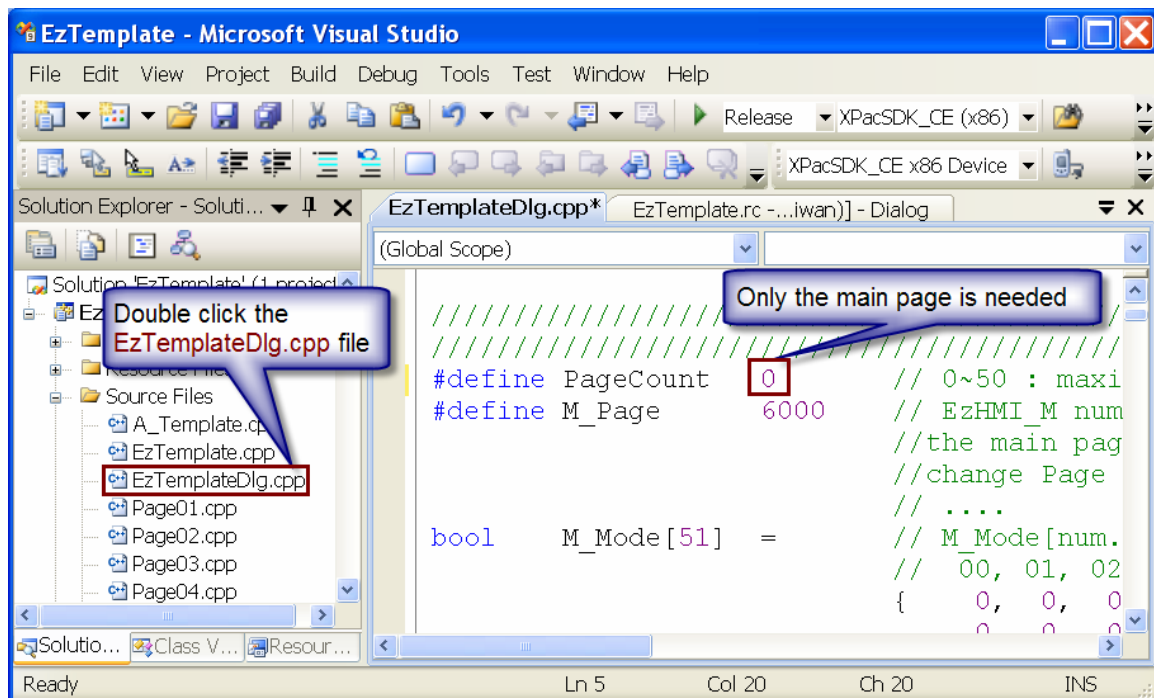
- LED
- SWITCH
- Label
- ButtonST
- ColorCheck

This example demonstrates how to implement Multilanguage selection during runtime by using an EzHMI ColorRadio object. The user only has to select the language from the item list of the ColorRadio to change the language setting. The languages listed in the following table will be supported by the demo program.

Index	Language	Text
0.	English	Multilanguage demonstration
1.	繁體中文	多語言的示範
2.	简体中文	多語言的示範
3.	日文	Multilanguage デモンストレーション
4.	German	Mehrsprachige Demonstration
5.	Spanish	Demostración multilingue
6.	Русско	Multilanguage демонстрация
7.	Portugese	Demonstração multilíngue

2.4.1 Page setting

This example only requires one page and that is the main page.

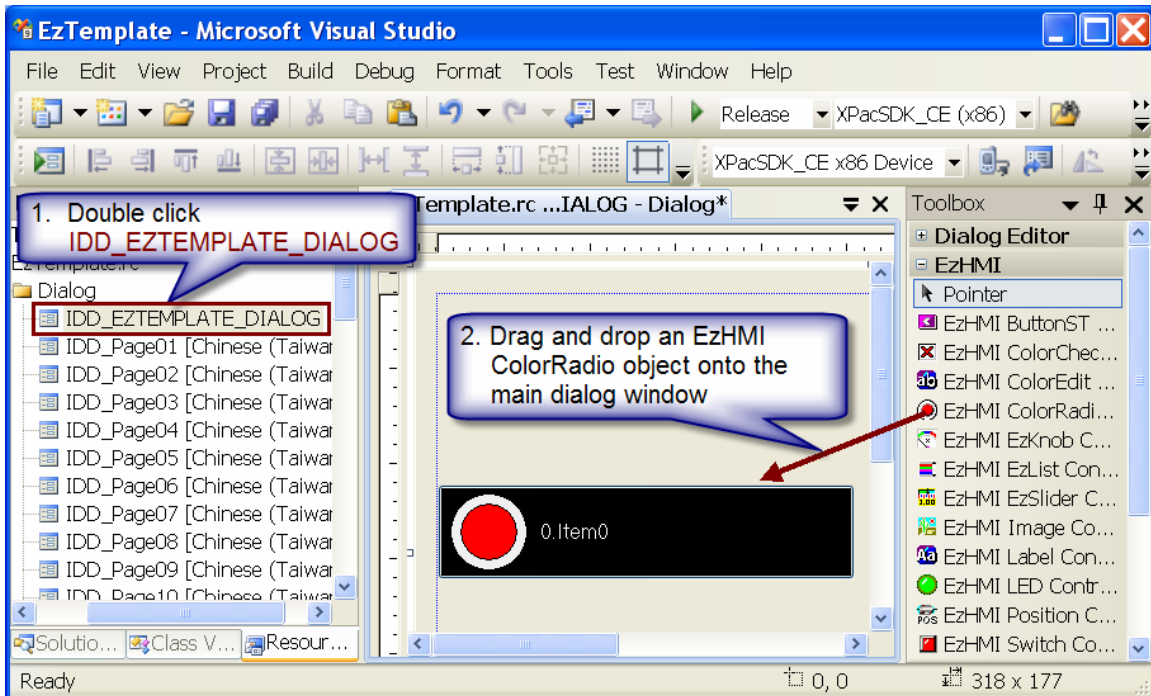


2.4.2 EzHMI setting

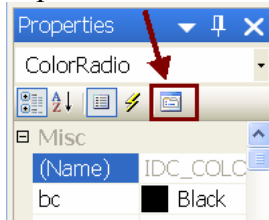
Add an EzHMI ColorRadio object to the main page and set the property according to the figure.

STEP 1: Open the IDD_EZTEMPLATE_DIALOG window

STEP 2: Drag and drop an EzHMI ColorRadio object onto the main dialog window



STEP 3: Open the property pages: Right click the ColorRadio object and select “Properties”. Click the “Property Pages” icon in the “Properties” window

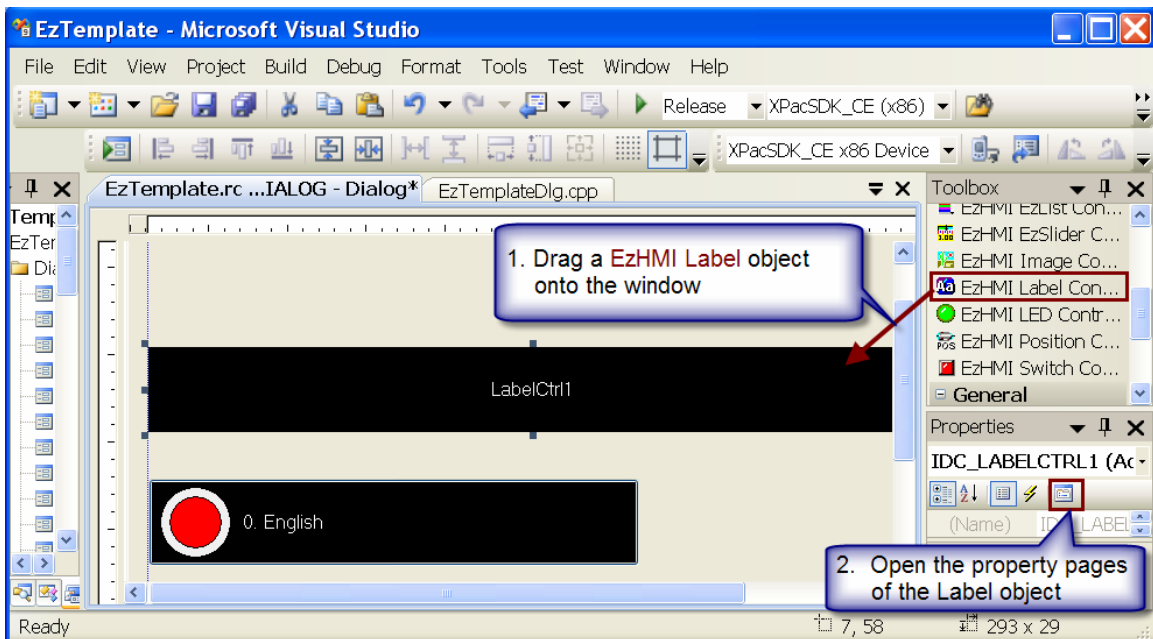


- STEP 4: Enter the language supported by this example to the “List Items” of the property sheet. The text boxes 0 to 7 represent the eight languages listed in the second column of the language table.
- STEP 5: Select the default language as “Language 0.”. In this example “Language 0.” represents English.
- STEP 6: Link the ColorRadio object to the D register number 8000. The value assigned to this register determines the current language.
- STEP 7: Close the property sheet by clicking “OK”.



STEP 8: Add a EzHMI Label object to the window

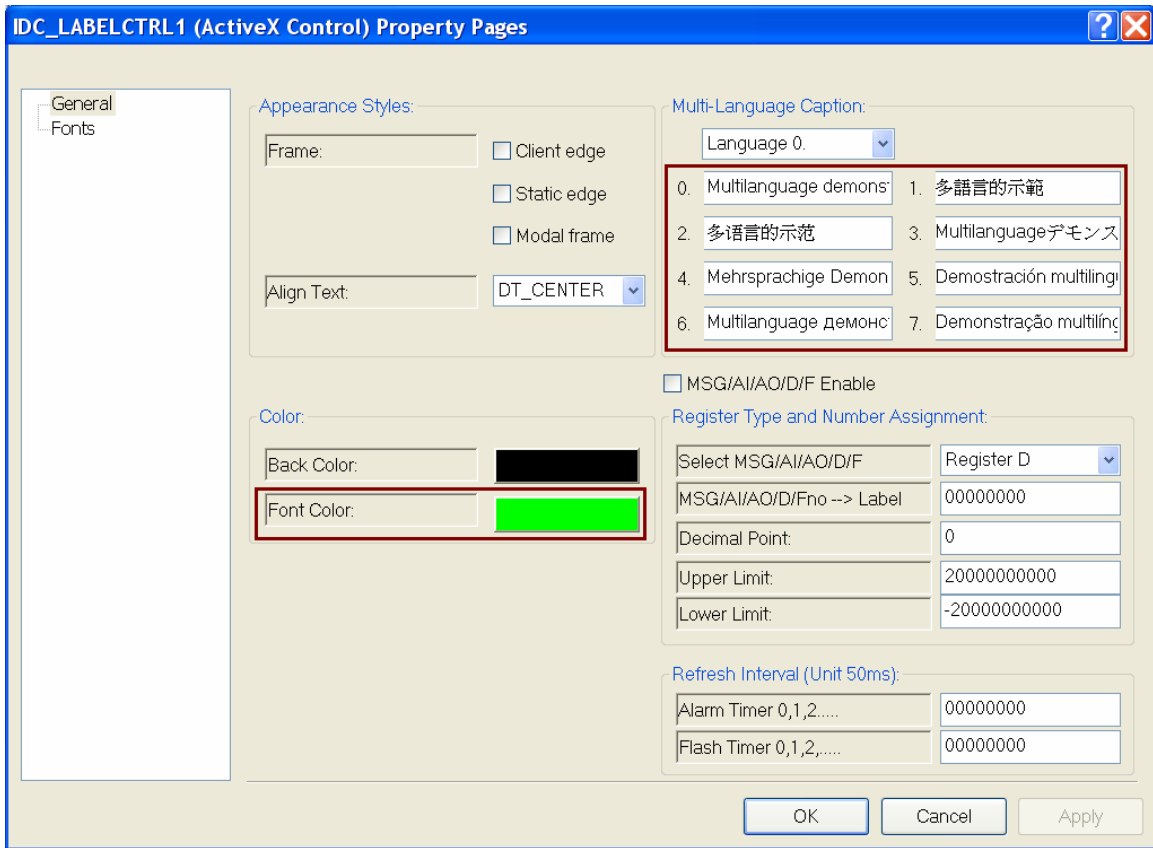
STEP 9: Open the property sheet of the Label object



STEP 10: Enter the text listed in the third column of the language table to the multi-language property caption section.

STEP 11: Set the caption text color to green

STEP 12: Click "OK"



STEP 18: Compile and download the project.

The default language setting is English. Therefore after program start the English text is displayed in the label object.



Click the ColorRadio object and a window with the list of language pops up:



Select Spanish as the HMI language and the text in the Label object switches to Spanish.



2.5 Tutorial 5: Digital IO

This example shows how to connect EzHMI LED and Switch object directly to digital IO channel without writing any code.

Plug one or more digital IO modules provided by ICPDAS in the PAC device. Map the individual IO channels of the modules to IO register by following the procedure described in chapter 2.1.

Open an EzTemplate project and rename

- the file “EzTemplate_800x600” to “HMI_DIO” and
- the execution file “EzTemplate.exe” to “HMI_DIO.exe”.

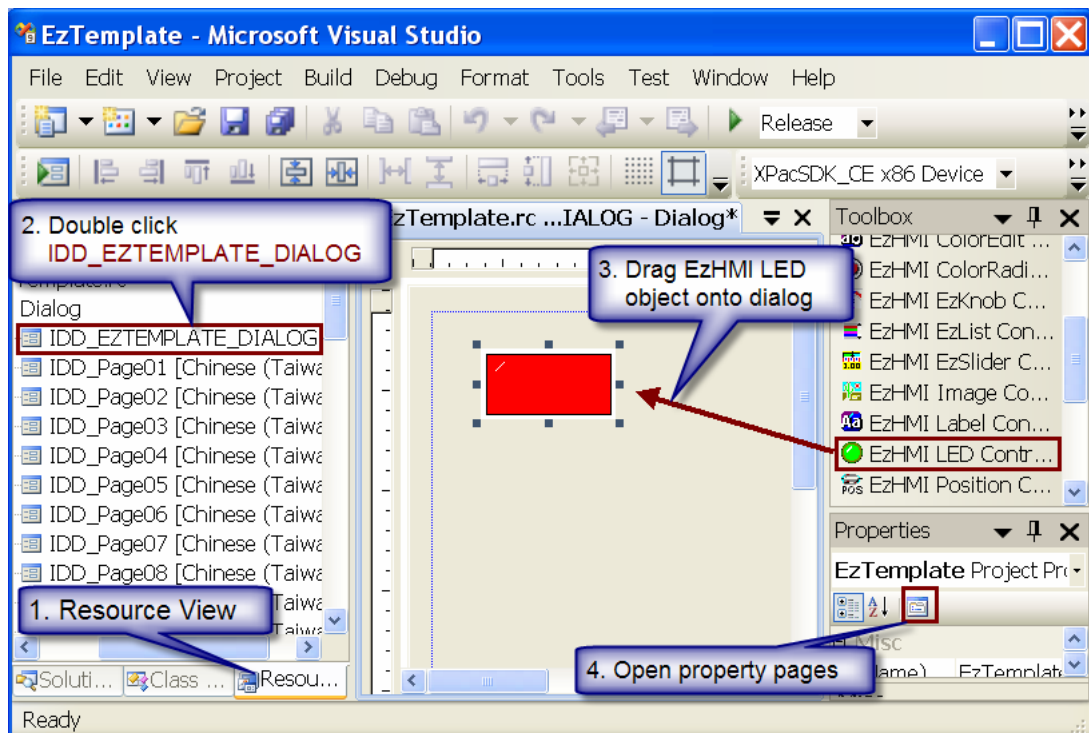
This procedure is described in chapter 2.2.1.

2.5.1 HMI design and IO register linking

STEP 1: Open the IDD_EZTEMPLATE_DIALOG window

STEP 2: Drag an EzHMI LED object onto the main dialog window

STEP 3: Open the property pages: Right click the LED object and select “Properties”.
Click the “Property Pages” icon in the “Properties” window



Make the following property settings:

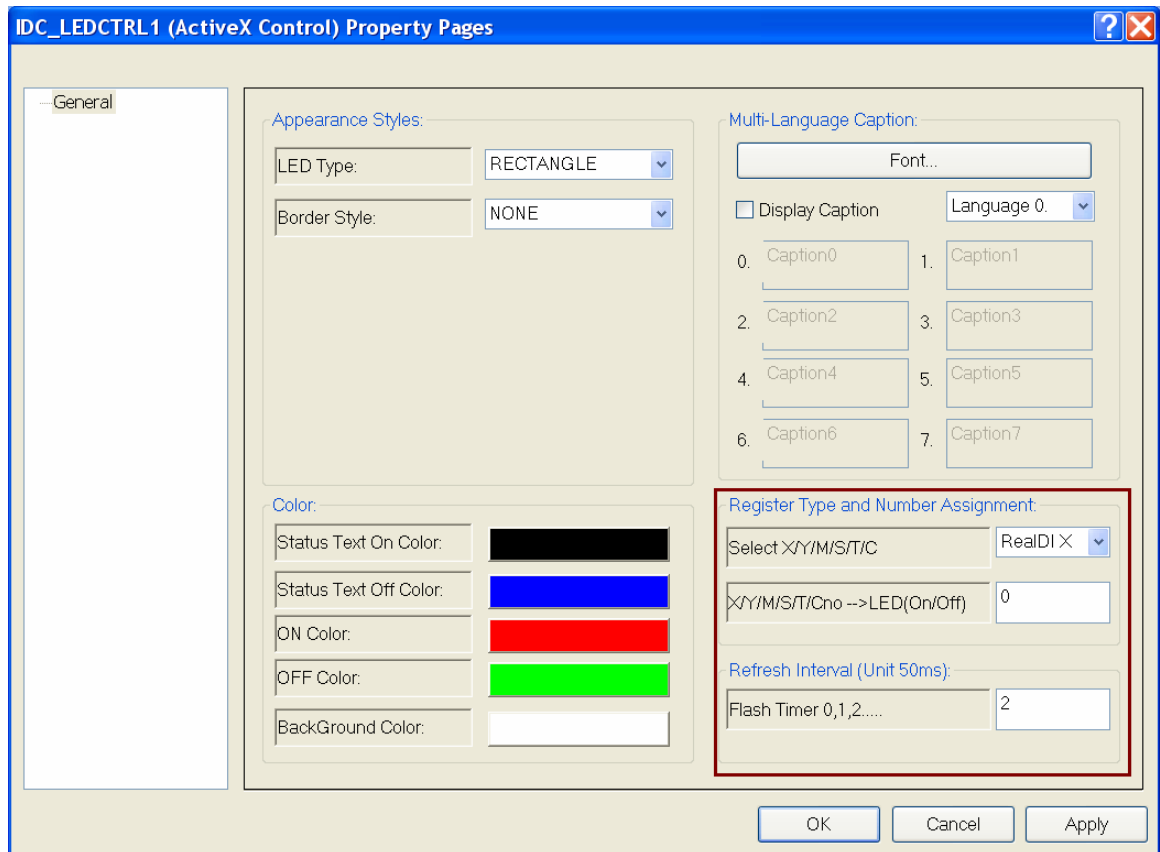
STEP 4: Link the LED object to the X (digital input) register number 0.

- Register type: Select “RealDI X” for “Select X/Y/M/S/T/C”
- Register number: Assign zero to “X/Y/M/S/T/Cno→LED(On/Off)”

STEP 5: Set the LED object refresh rate to 100 milliseconds.

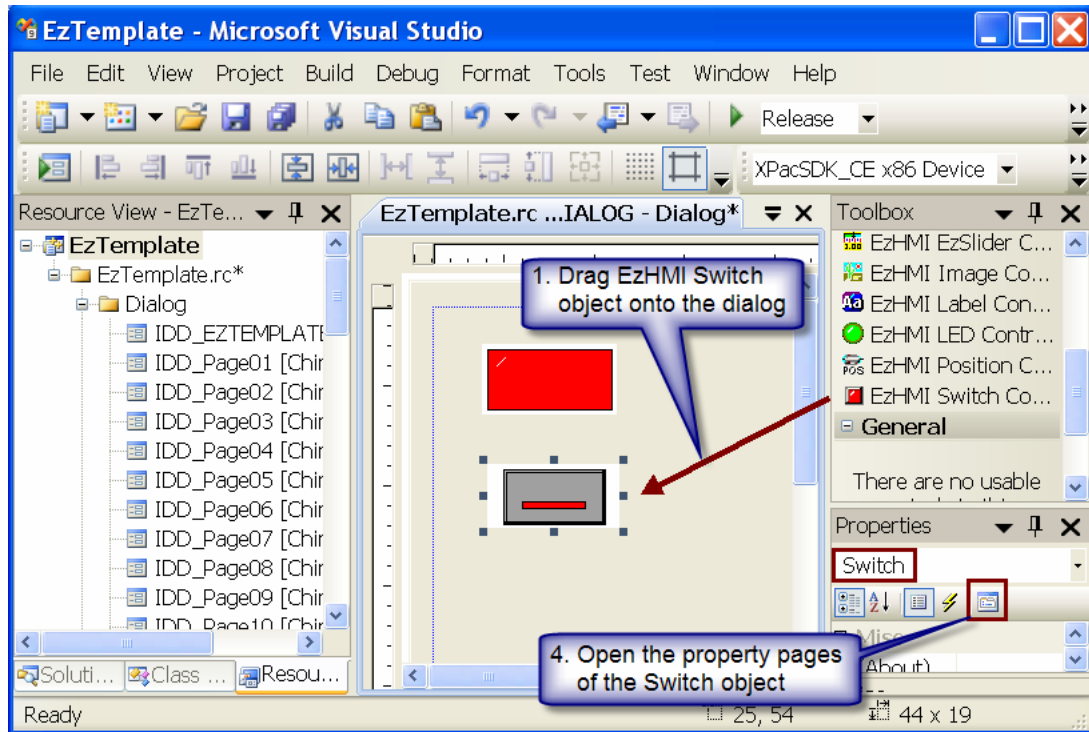
- Assign “Flash Timer 0,1,2...” the value 2.

STEP 6: Close the property page by clicking “OK”



STEP 7: Drag an EzHMI SWITCH object onto the main dialog window

STEP 8: Open the property pages: Right click the SWITCH object and select “Properties”. Click the “Property Pages” icon in the “Properties” window



Make the following property settings:

STEP 9: Choose “SWITCH TOGGLE” as switch style.

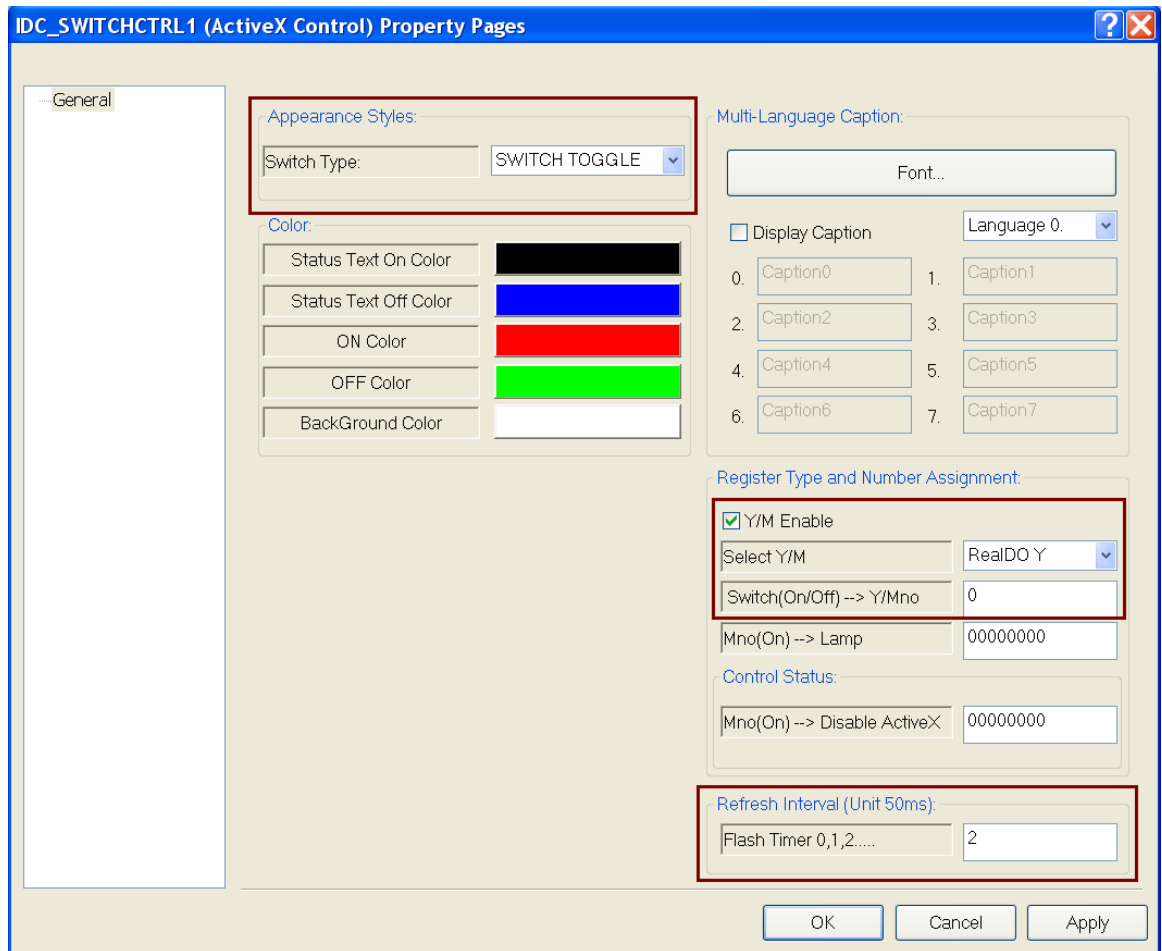
STEP 10: Link the Switch object to the Y (digital output) register number 0.

- Register type: Select “RealDO Y” for “Select Y/M”
- Register number: Assign zero to “Switch (On/Off)→Y/Mno”

STEP 11: Set the Switch object refresh rate to 100 milliseconds.

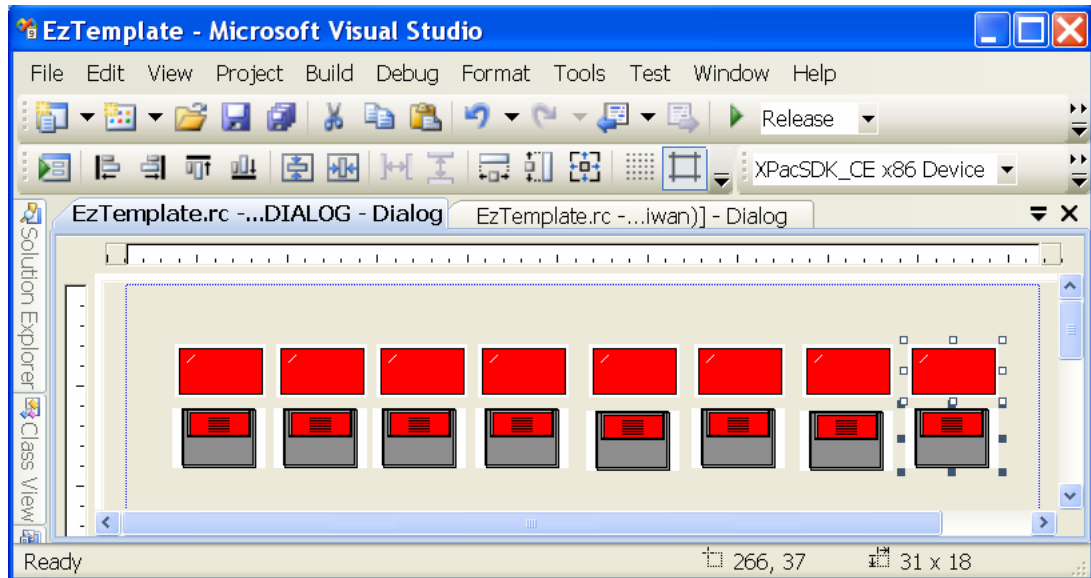
- Assign “Flash Timer 0,1,2...” the value 2.

STEP 12: Close the property page by clicking “OK”



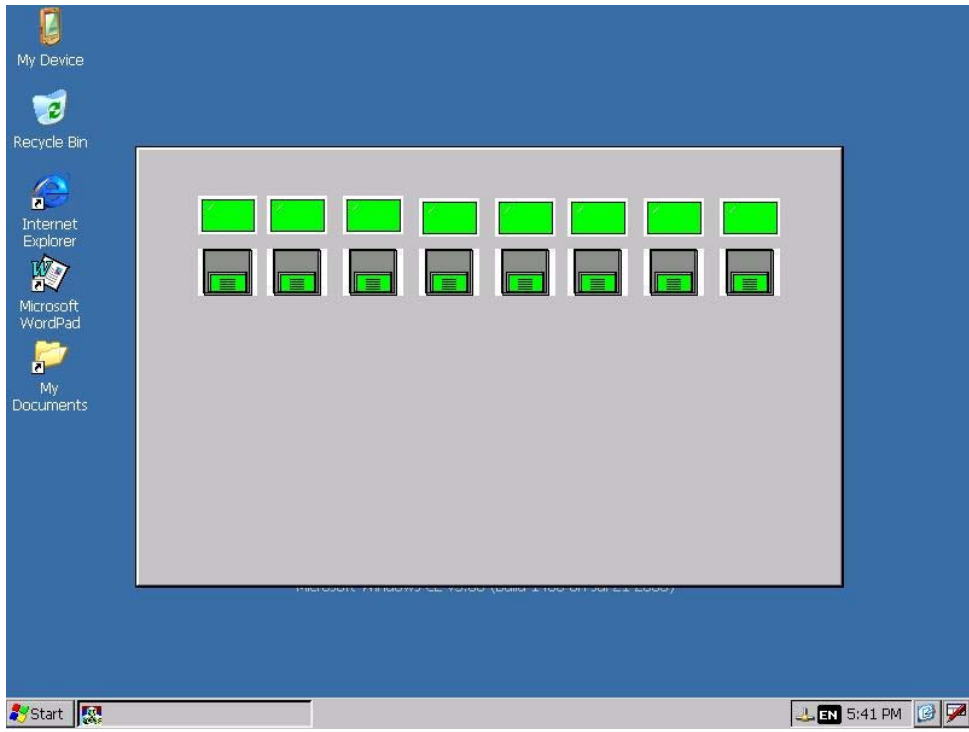
STEP 13: Copy and paste both LED and Switch objects eight times.

- Select both objects at once:
 - i. position the mouse pointer at the upper left corner of the objects
 - ii. hold left mouse button down and drag a rectangle over the LED and Switch objects
 - iii. release the mouse button
- Copy the objects (Ctrl+C)
- Paste the objects (Ctrl+V)



Open the property sheets for each copied object and make the following changes:

- STEP 14: Link the copied LED object in ascending sequence to the X register number 1 to 7.
Assign the first copied LED the register number 1 in the “X/Y/M/S/T/Cno→LED(On/Off)” text box and the second copied LED the number 2, etc.
- STEP 15: Link the copied Switch objects in ascending sequence to the Y (digital output) register number 1 to 7. Enter the register number 1 for the first copied Switch object in “Switch (On/Off)→Y/Mno” edit box and for the second Switch object the register number 2, etc.
- STEP 16: Compile the project and download the execution file as described in chapter 2.2.3.



2.6 Tutorial 6: Analog IO

With the help of EzTemplate and the ActiveX controls analog IO values can be read and set without implementing any additional code. This example demonstrates this by using the EzSlider and EzKnob ActiveX controls.

2.6.1 Slot module configuration

This example uses the analog output module I-8024 and the analog input module I-8017H.

First of all analog IO modules have to be plugged in the PAC device. The individual IO channels have to be mapped to IO register by using the EzConfig utility. Follow the procedure described in chapter 2.1.

Now each analog module has to be configured:

For each module and channel specify the

- physical unit of the analog input or output data (e.g. ampere, volt, temperature)
- data range (e.g. -50 to 50mV, 4 to 20mA).
- Offset, etc

STEP 1: Start the EzConfig utility.

STEP 2: Click the “Scan Slot 1~7” button to scan the slots for any device modules. All the channels of the detected modules will be automatically be assigned a register number of the respective register type.

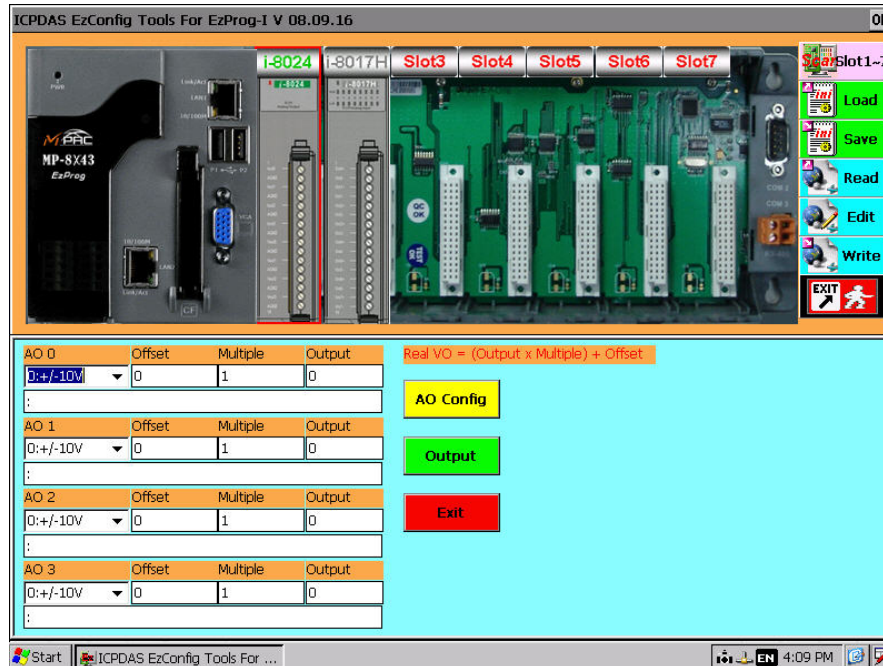
Next do the necessary module and channel configurations

STEP 3: Click on the I-8024 picture to open its property window. The register mapping for each channel is displayed. AO 0 to AO3 indicates that the channels are linked to the AO register with the numbers 0 to 3. The register mappings are fixed and can not be changed by the user.

In this example the EzSlider will be linked to the first channel of the I-8024 module. The slider has therefore to be linked to the AO register number 0. The channel should output voltage in the range of -10V to 10V. Select the physical unit and data range of channel 0 according the following figure.

STEP 4: Click the “AO Config” button to configure the module

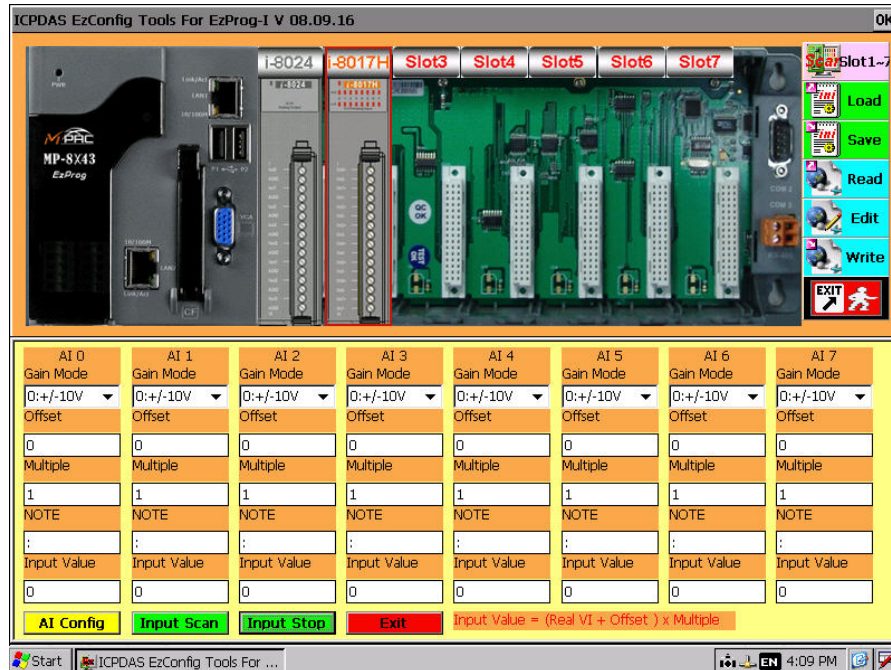
STEP 5: Click “Exit”



STEP 6: Click on the I-8017H picture to open its property window. Again the register mappings for the eight input channel are displayed. AI 0 to AO7 indicates that the input channels are linked to the AI registers with the numbers 0 to 7. The EzKnob object will be linked to the last channel of the I-8017H module that is to the AO register number 7. The channel should read input voltage in the range of -10V to 10V. Select the physical unit and data range of channel 7 according the following figure.

STEP 7: Click the “AI Config” button to send the configuration to the module.

STEP 8: Click “Exit”



2.6.2 EzHMI design and register settings

Open an EzTemplate project and rename

- the file “EzTemplate_800x600” to “HMI_AIO” and
- the execution file “EzTemplate.exe” to “HMI_AIO.exe”.

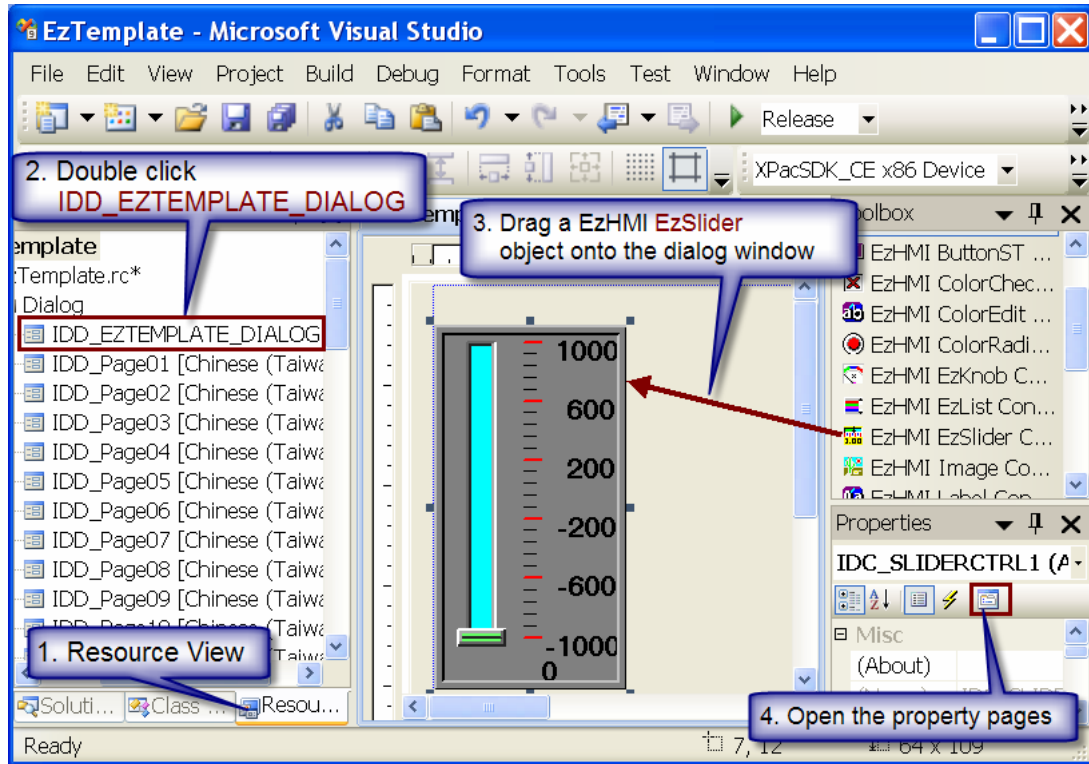
This procedure is described in chapter 2.2.1.

The following procedure describes how to use a EzSlider and EzKnob in conjunction with analog IO channels without writing any code:

STEP 1: Open the IDD_EZTEMPLATE_DIALOG window

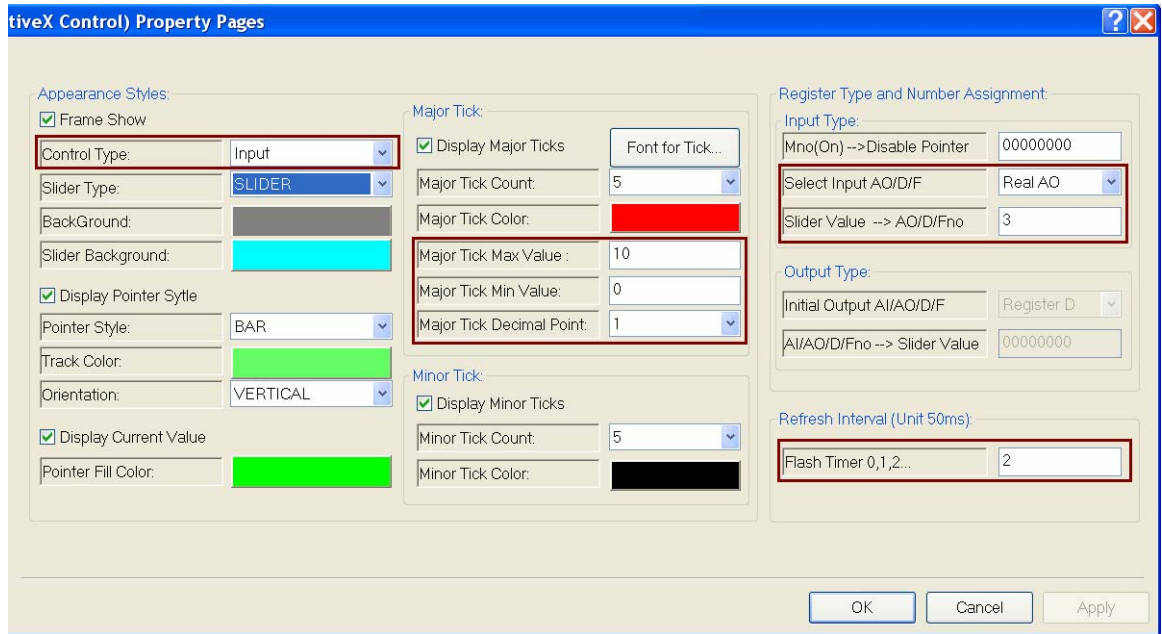
STEP 2: Drag an EzHMI EzSlider object onto the main dialog window

STEP 3: Open the property pages: Right click the EzSlider object and select “Properties”. Click the “Property Pages” icon in the “Properties” window



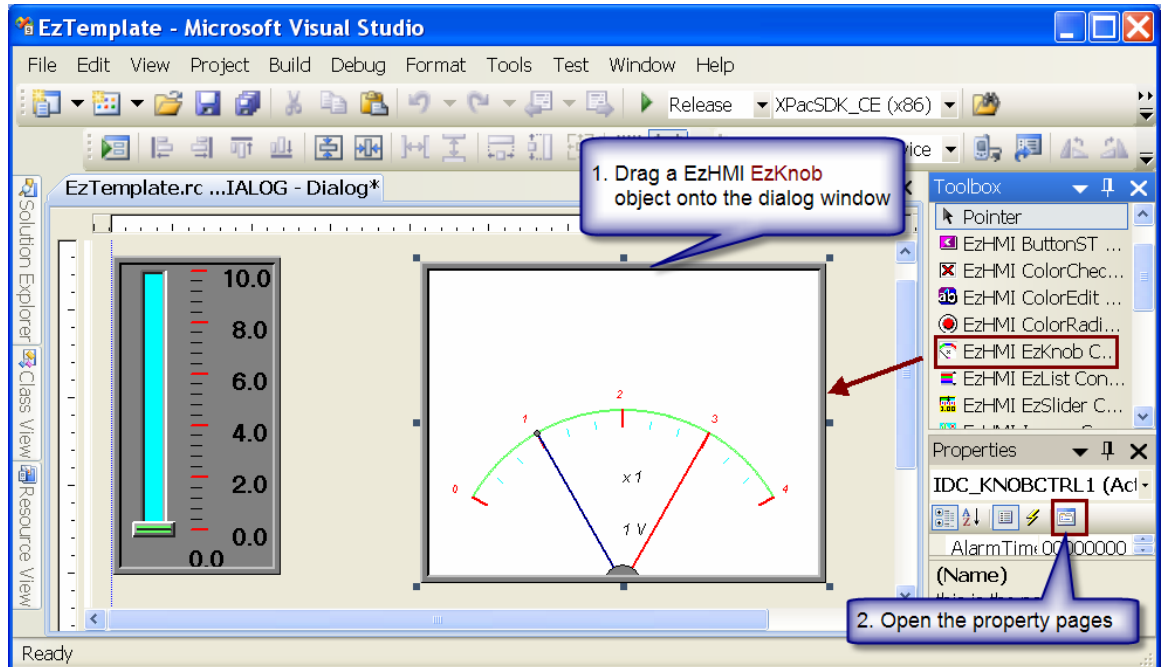
Enter the following property settings:

- STEP 4: Select “Input” as control type. In this mode the EzSlider object allows the user to change the slider position with the mouse. The associated register will immediately be updated to the new position value.
- STEP 5: Define the number of ticks to be displayed on the slider. Set the minimum value on the slider scale to zero and the decimal point of the tick values to one
- STEP 6: Link the EzSlider object to the AO (analog output) register number 3.
- Register type: Select “Real AO” for “Select Input AO/D/F”
 - Register number: Enter for “Slider Value →AO/D/Fno” the number 3.
- STEP 7: Set the EzSlider object refresh rate to 100 milliseconds.
- Assign “Flash Timer 0,1,2...” the value 2.
- STEP 8: Close the property page by clicking “OK”



STEP 9: Drag an EzHMI EzKnob object onto the main dialog window

STEP 10: Open the property pages: Right click the EzKnob object and select “Properties”. Click the “Property Pages” icon in the “Properties” window



STEP 11: Select “Output” as control type. In this mode the EzKnob object allows no user input and it just displays the value of the associated register.

STEP 12: Set the number of ticks to be displayed on the EzKnob to 10. Set the minimum value on the EzKnob to zero and the decimal point of the tick label values to one.

STEP 13: Link the EzKnob object to the AI (analog input) register number 7.

- Register type: Select “Real AI” for “Select Output AI/AO/D/F”
- Register number: Enter for “AI/AO/D/Fno → Knob Value ” the number 7.

STEP 14: Set the EzKnob object refresh rate to 100 milliseconds.

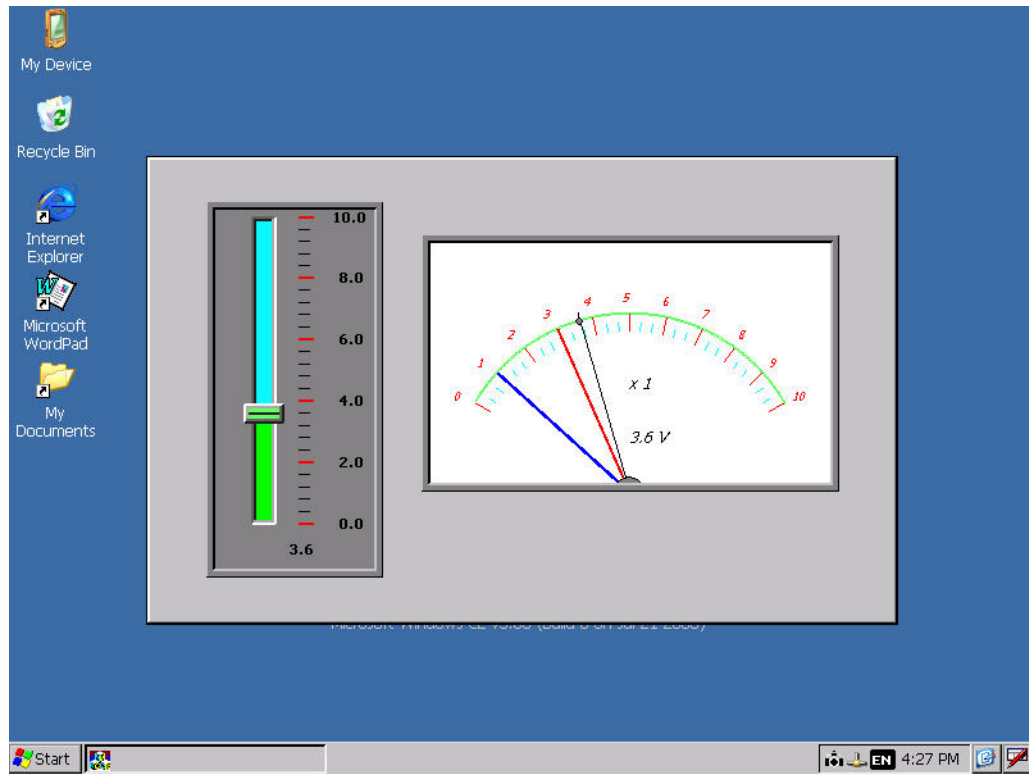
- Assign “Flash Timer 0,1,2...” the value 2.

STEP 15: Close the property page by clicking “OK”

The screenshot shows the configuration interface for an EzKnob. The settings are organized into several panels:

- Appearance Styles:**
 - Control Type: Output
 - Knob Type: Knob1
 - Major Tick:
 - Display Major Ticks: checked
 - Major Tick Color: Red
 - Major Tick Count: 10
 - Major Tick Min Value: 0
 - Engineering Scale: 1
 - Major Tick Max Value: 10
 - Decimal Format: #
 - Minor Tick:
 - Display Minor Tick: checked
 - Minor Tick Count: 3
 - Minor Tick Color: Cyan
 - Color:
 - BackGround Color: White
 - Knob BackGround Color: Cyan
 - Track Color: Green
 - Indicator Type: 1
 - Indicator Color: Black
 - Indicator Center Color: Gray
- Dial Text Labels:**
 - Tick:
 - Tick Font Color: Red
 - Tick Back Color: White
 - Tick Font Size: 12
 - Tick Position(++Pixel): 0
 - Scale Factor:
 - Engineering Color: Black
 - Engineering Back Color: White
 - Engineering Font Size: 16
 - Engineering Position(++Pixel): 0
 - Tick Unit Label:
 - Value Color: Black
 - Value Back Color: White
 - Value Font Size: 16
 - Value Unit: V
 - Value Position(++Pixel): 0
- Register Type and Number Assignment:**
 - Input Type:
 - Select Input AO/D/F: Register D
 - Knob Value --> AO/D/Fno: 00000000
 - Control Status:
 - Mno(On) --> Disable Indicator: 00000000
 - Output Type:
 - Select Output AI/AO/D/F: Real AI
 - AI/AO/D/Fno --> Knob Value: 7
 - Input Limited or Alarm Level of Output:
 - Upper Limit: 3
 - Lower Limit: 1
 - Refresh Interval (Unit 50ms):
 - Alarm Timer 0,1,2...: 00000000
 - Flash Timer 0,1,2...: 2

STEP 17: Compile the project and download the execution file as described in chapter 2.2.3. The following figure shows the user interface on the PAC.



2.7 Tutorial 7: User Thread

The UserThread will execute just once after it has been called. You can use a loop (while-loop or for-loop) inside the user thread to execute the code more than once. It is recommended to add a sleep between each loop cycle to reduce the CPU load.

The application of a UserThread in connection with a EzHMI Label and ButtonST is being demonstrated in this chapter. Every Time the user presses the ButtonST the Label displays the string “Hello”.

2.7.1 EzHMI design and register settings

Open an EzTemplate project and rename

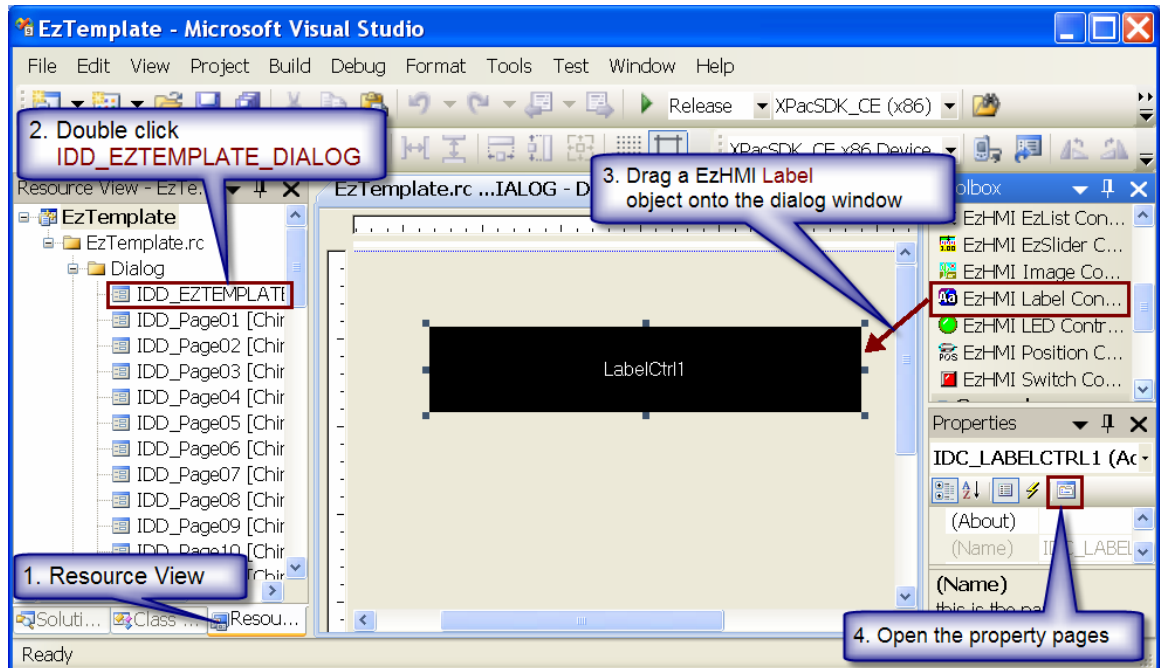
- the file “EzTemplate_800x600” to “UserThread” and
- the execution file “EzTemplate.exe” to “UserThread.exe”.

This procedure is described in chapter 2.2.1.

STEP 1: Open the IDD_EZTEMPLATE_DIALOG window

STEP 2: Drag an EzHMI Label object onto the main dialog window

STEP 3: Open the property pages: Right click the Label object and select “Properties”.
Click the “Property Pages” icon in the “Properties” window



Enter the following property settings:

STEP 4: Activate the “**MSG/AI/AO/D/F Enable**” check box. This tells the Label object that it is linked to a register.

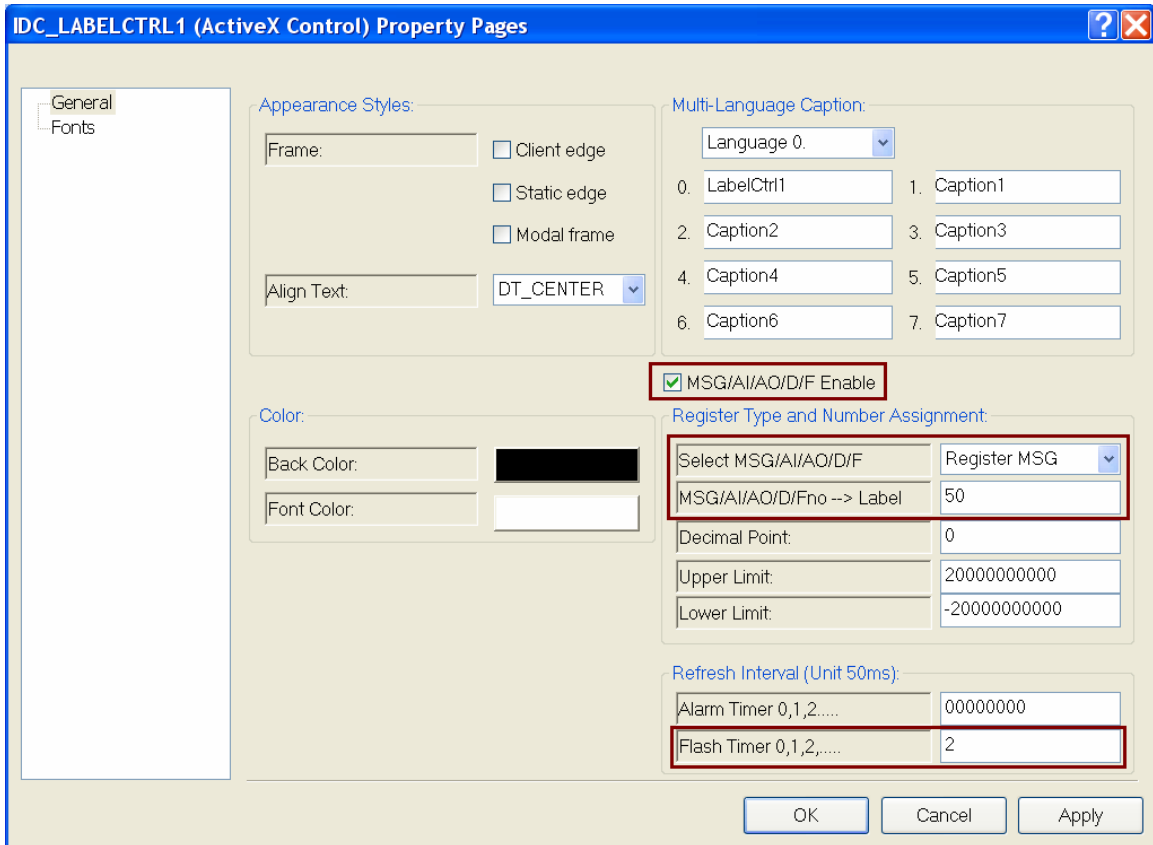
STEP 5: Link the Label object to the MSG (message) register number 50.

- Register type: Select “Register MSG” for “*Select MSG/AI/AO/D/F*”
- Register number: Enter for “*MSG/AI/AO/D/Fno →Label*” the number 50.

STEP 6: Set the Label object refresh rate to 100 milliseconds.

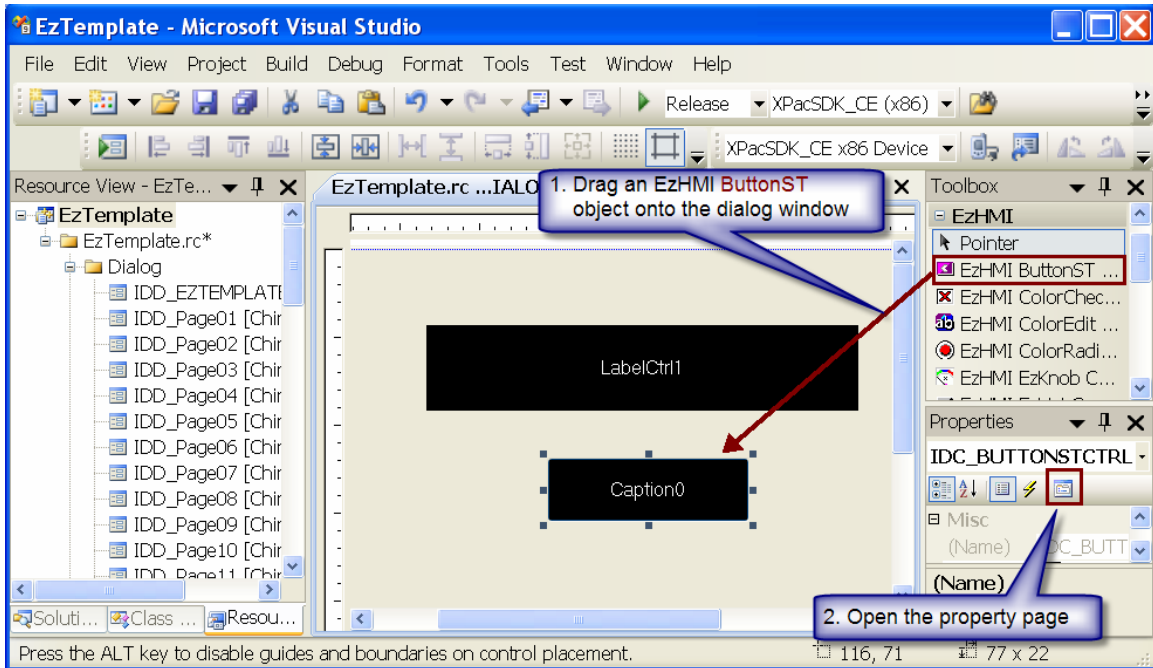
- Assign “Flash Timer 0,1,2...” the value 2.

STEP 7: Close the property page by clicking “OK”



STEP 8: Drag an EzHMI ButtonST object onto the main dialog window

STEP 9: Open the property pages: Right click the ButtonST object and select “Properties”. Click the “Property Pages” icon in the “Properties” window

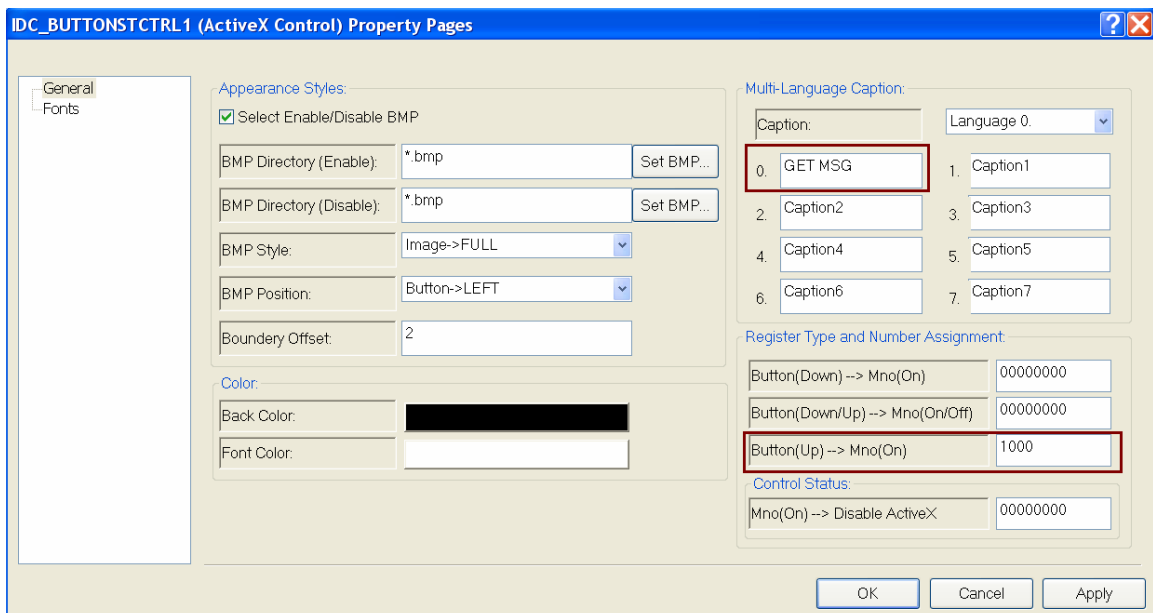


STEP 10: Enter the button caption “GET MSG”.

STEP 11: Link the button up event to the M register number 1000. Every time when the button up event is generated the M register 1000 is set to true. This example will show you how to process the event in the UserThread.

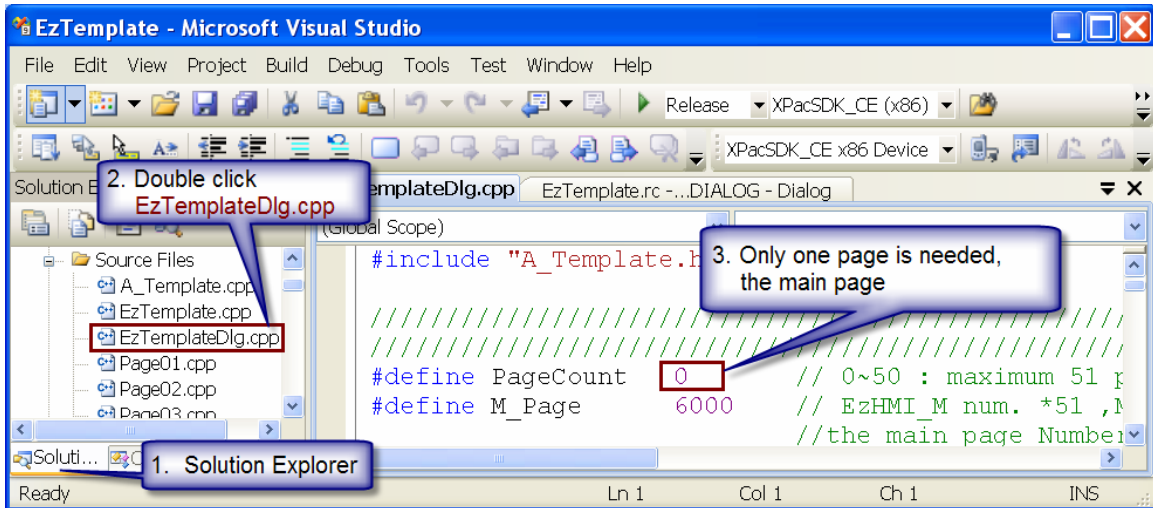
- Register number: Enter for “*Button(Up) → Mno(On)*” the number 1000.

STEP 12: Close the property page by clicking “OK”

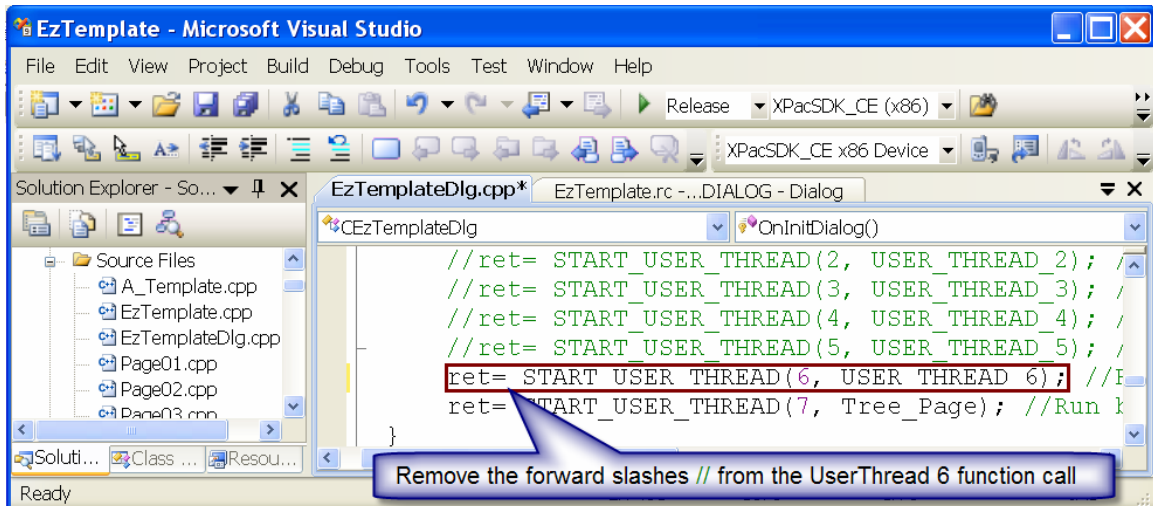


2.7.2 UserThread activation and code implementation

STEP 1: Set the number of dialog pages required for the project. For this example only the main page is needed therefore assign a 0 to the PageCount definition.

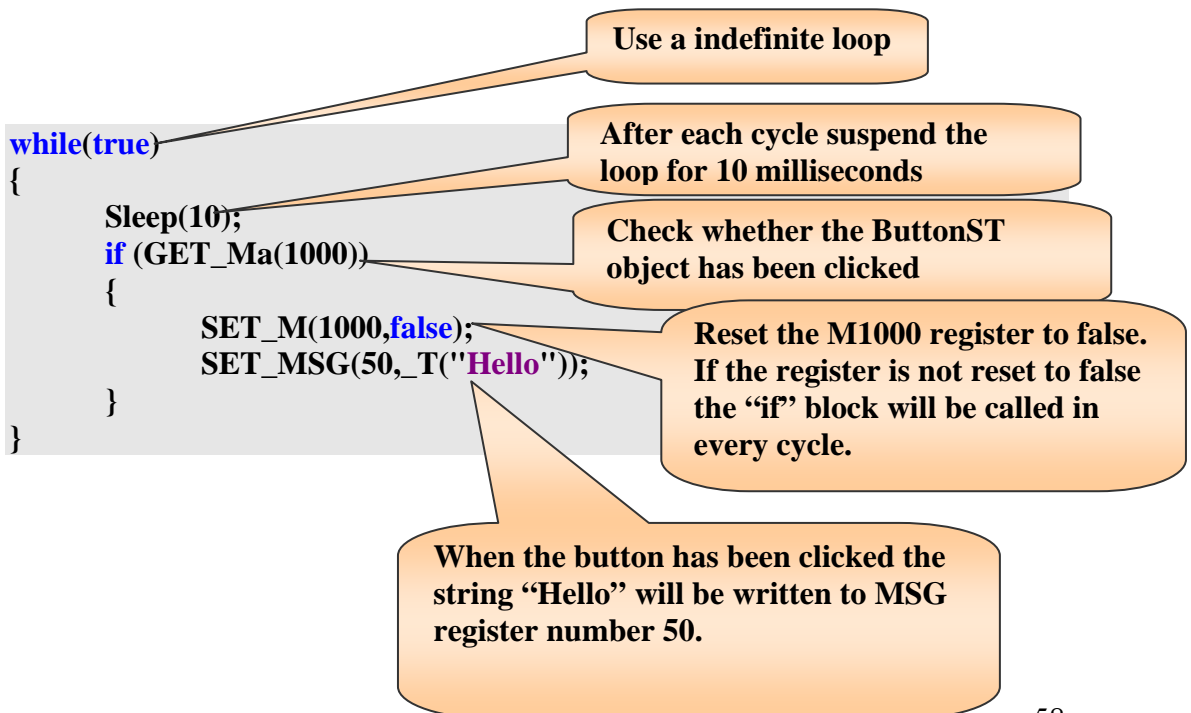
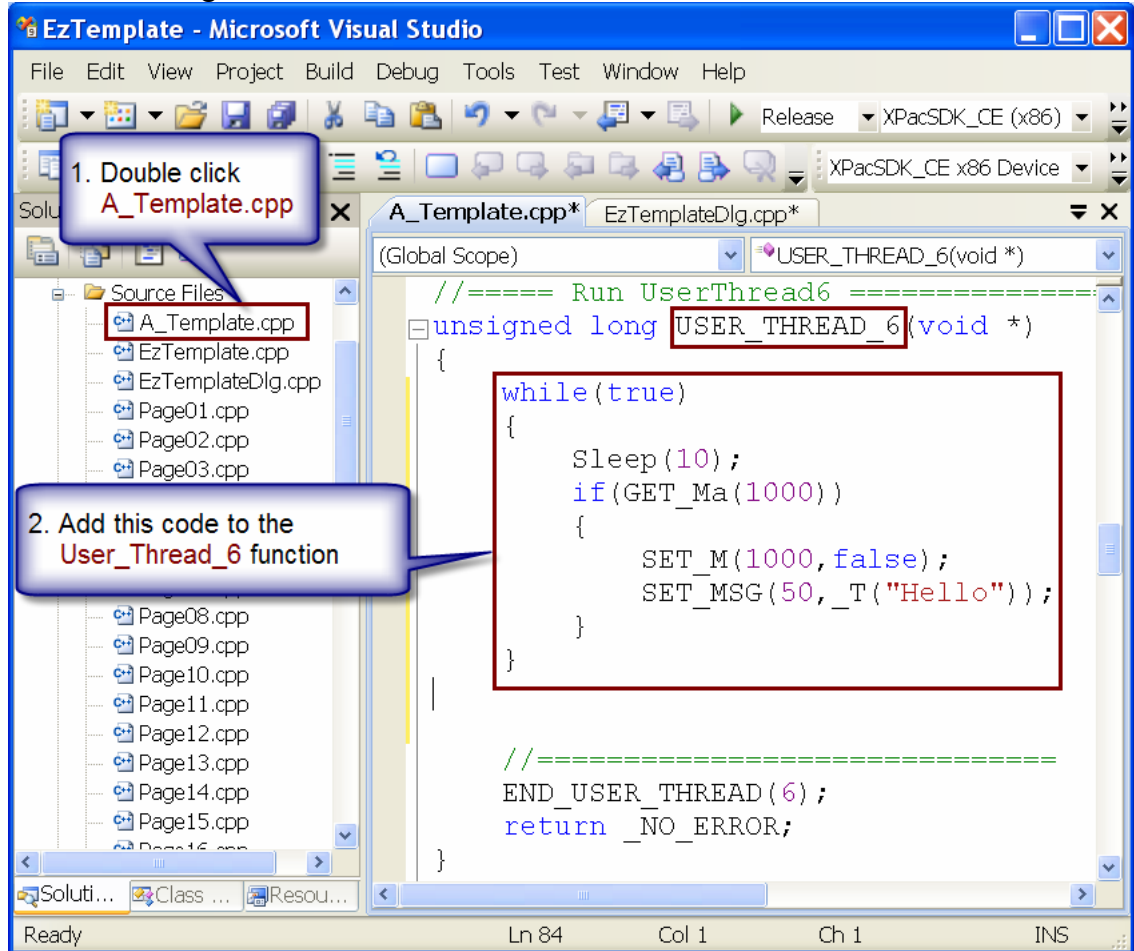


STEP 2: The User_Thread_6 will be used to handle the ButtonST event. Therefore it is necessary to remove the forward slashes from the User_Thread_6 function call.

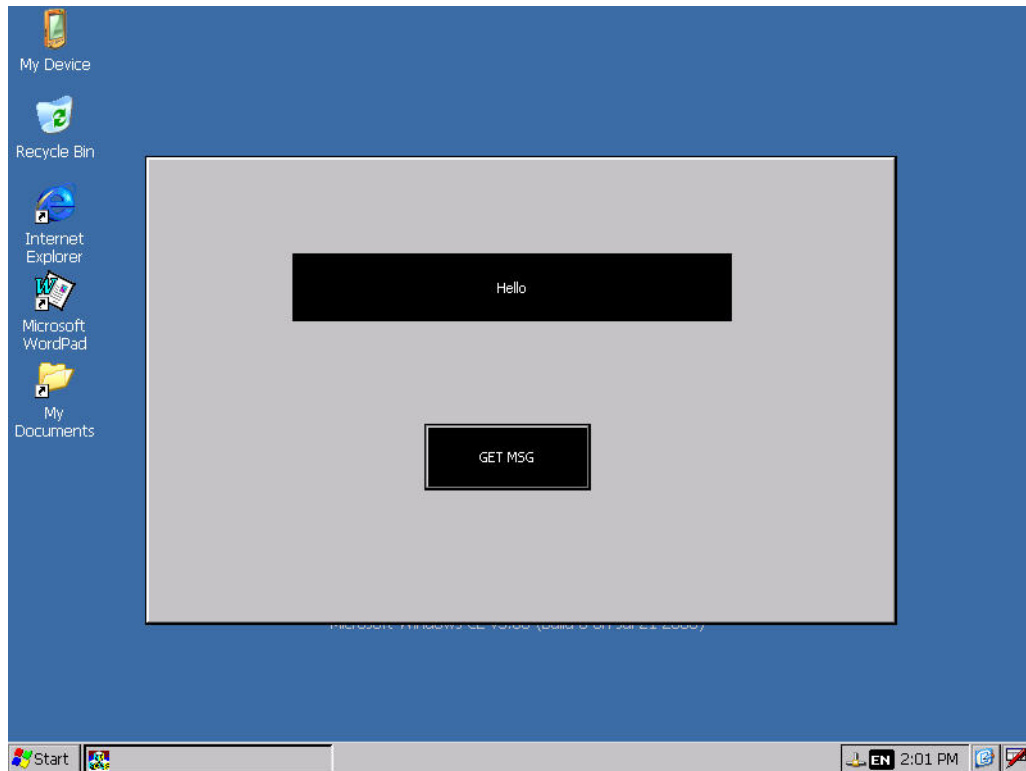


STEP 3: Add the following code to User_Thread_6 function
The indefinite loop checks every 10 millisecond whether the button has been clicked. If the button had been clicked the ButtonST M register is reset to

false and the Label object linked to MSG register number 50 linked receives the message “Hello”.



STEP 4: Compile the project and download the execution file as described in chapter 2.2.3. The following figure shows the user interface on the PAC. As soon as the user clicks the “GET MSG” button the text “Hello” appears on the label object.



2.8 Tutorial 8: RTSR

RTSR are functions which are being called at a set time interval by the EzCore engine. RTSR enables the developer to create a real time and deterministic control program. Eight functions are provided for the programmer whereby each has a different priority level. `USER_THREAD_0` has the highest and `USER_THREAD_7` the lowest priority. The priority level of each RTSR is fixed and can not be changed but the scanning interval of the RTSR can be set by changing the default function call interval. To guaranty real time and deterministic behavior no sleep or indefinite loop are allowed inside the RTSR functions.

This example shows how to activate a RTSR and set the scan time interval. In addition an label object is being used to display the time stamp of every RTSR function call.

2.8.1 EzHMI design and register settings

Open an EzTemplate project and rename

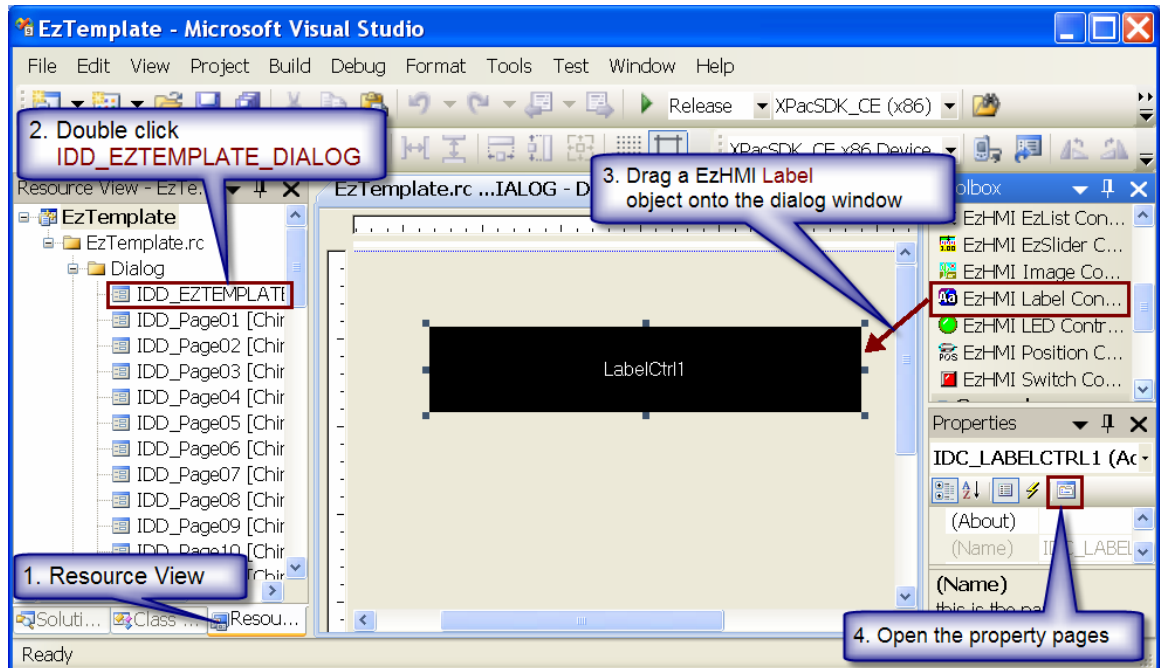
- the file “EzTemplate_800x600” to “RTSR” and
- the execution file “EzTemplate.exe” to “RTSR.exe”.

This procedure is described in chapter 2.2.1.

STEP 1: Open the `IDD_EZTEMPLATE_DIALOG` window

STEP 2: Drag an EzHMI Label object onto the main dialog window

STEP 3: Open the property pages: Right click the Label object and select “Properties”.
Click the “Property Pages” icon in the “Properties” window



Enter the following property settings:

STEP 4: Activate the “**MSG/AI/AO/D/F Enable**” check box. This tells the Label object that it is linked to a register.

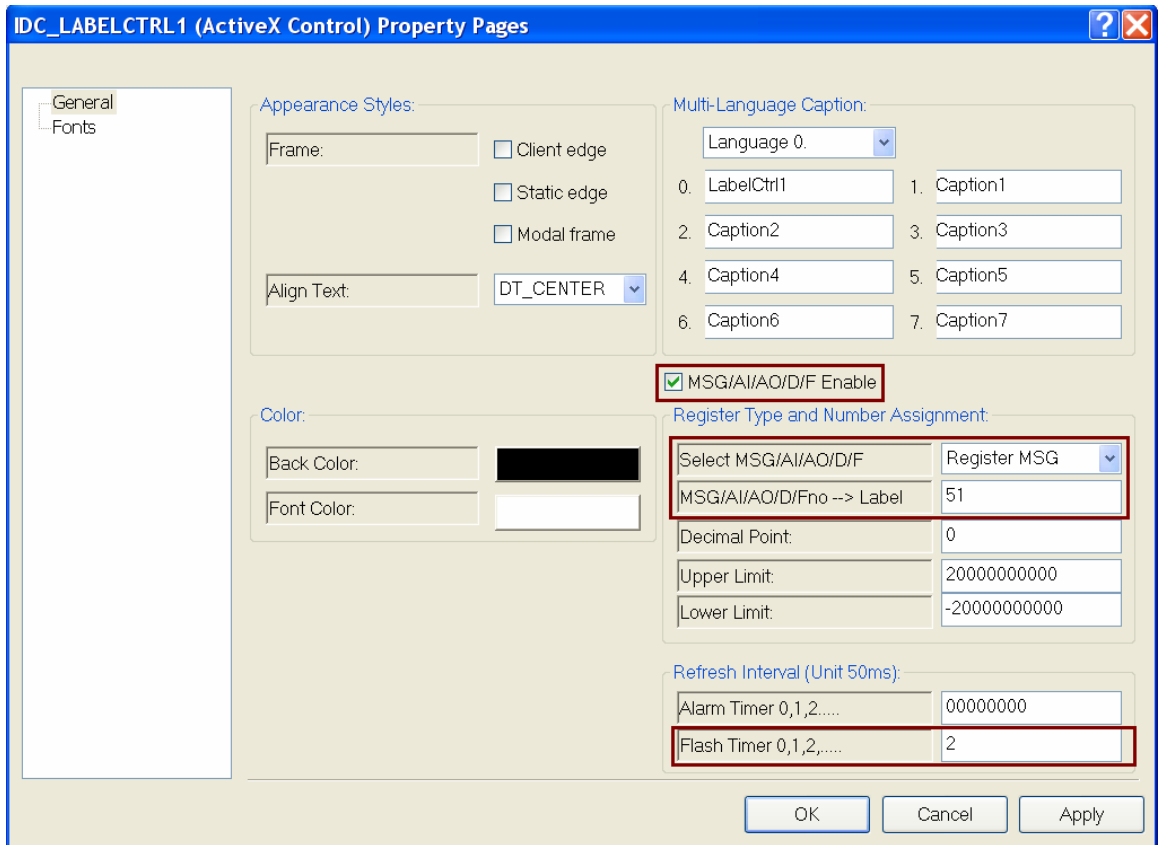
STEP 5: Link the Label object to the MSG (message) register number 51.

- Register type: Select “Register MSG” for “*Select MSG/AI/AO/D/F*”
- Register number: Enter for “*MSG/AI/AO/D/Fno →Label*” the number 51.

STEP 6: Set the Label object refresh rate to 100 milliseconds.

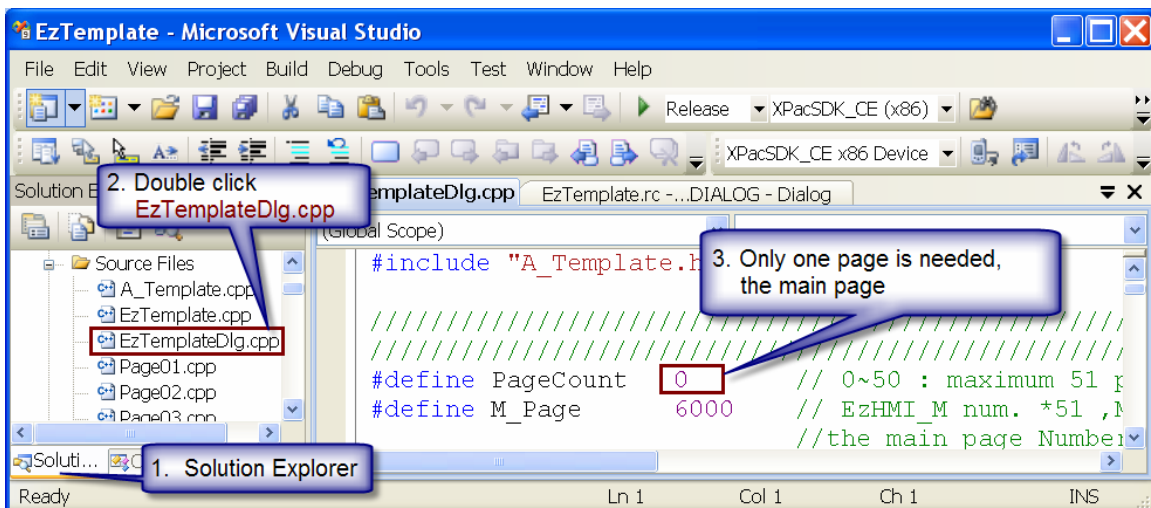
- Assign “Flash Timer 0,1,2...” the value 2.

STEP 7: Close the property page by clicking “OK”

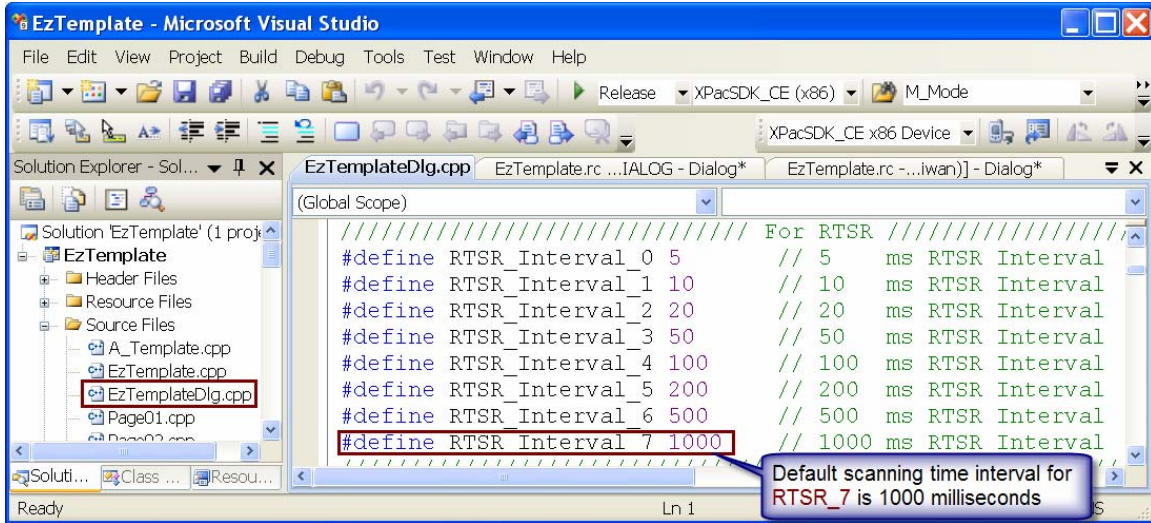


2.8.2 RTSR activation and code implementation

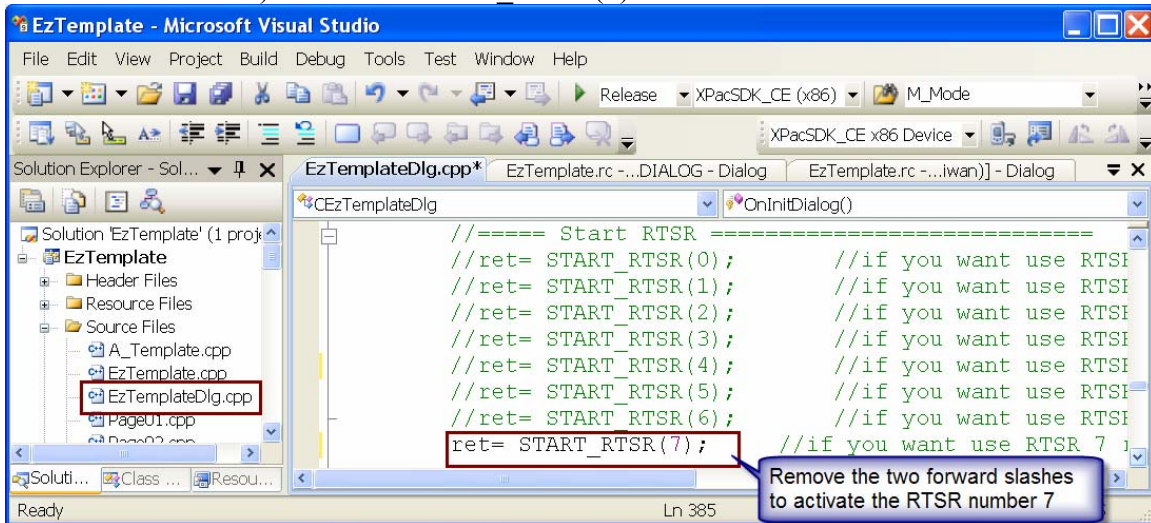
STEP 1: Determine the number of dialog pages required for the project. For this example only the main page is needed therefore assign a 0 to the PageCount definition.



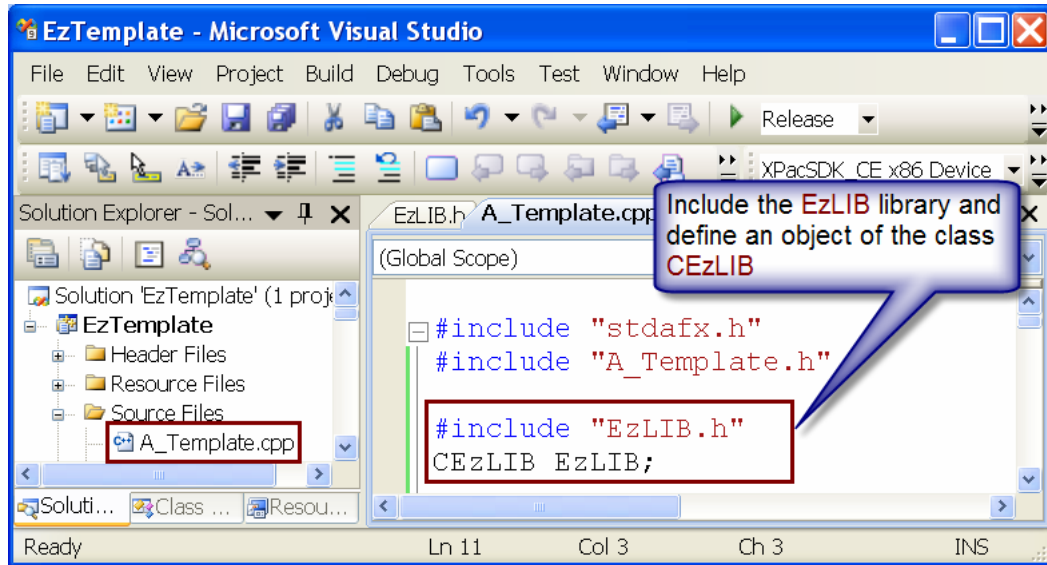
STEP 2: Set the RTSR scanning time interval. In this example the RSTR number 7 with the lowest priority level will be used. The default scanning time interval of 1000 milliseconds is not changed and taken over.



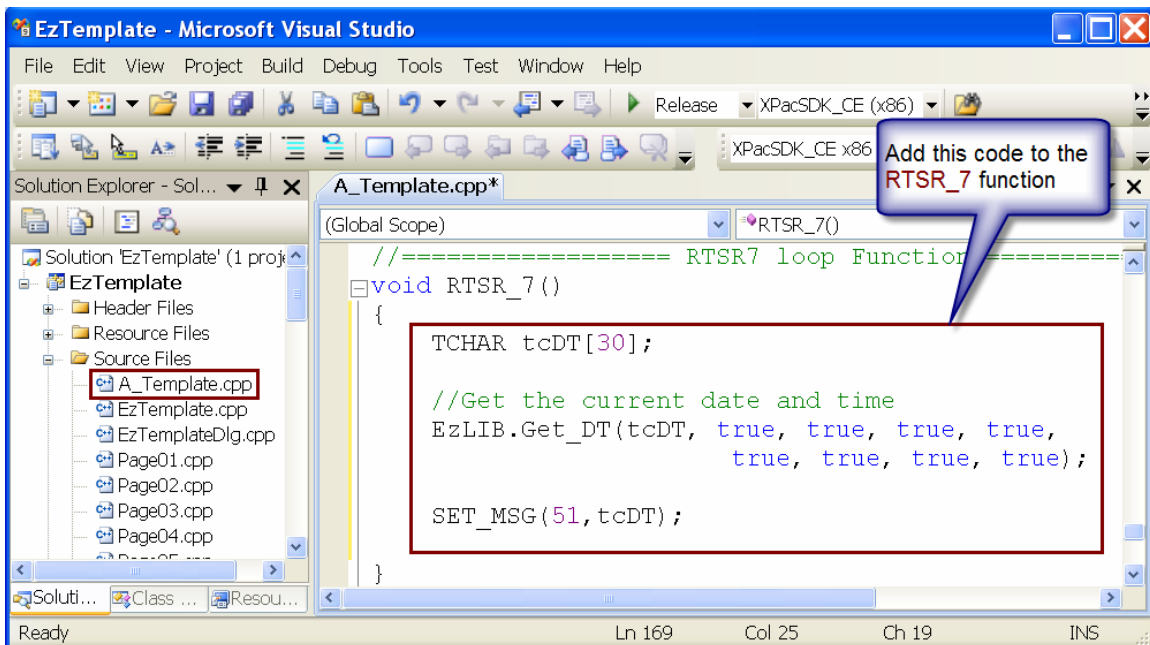
STEP 3: Activate the RTSR number 7 by removing the c++ comment (two forward slashes) from the START_RTSR(7) function call.



STEP 4: For each RTSR cycle the date and time will be displayed by the EzHMI Lable object. The function for getting the current date and time is defined in the class CEzLIB of the EzLIB library. Therefore the header file of the EzLib has to be included at the beginning of the `A_Template.cpp` file. In addition a global object of the CEzLIB class has to be defined in order for its member function to be called inside the RTSR.



STEP 5: Add the following code to the RTSR_7 function. The code basically reads the current date and time and writes this information to the MSG register number 51. The EzHMI Label object which is directly linked to this MSG register displays this information. The EzLIB manual describes the *Get_DT* function in full details.



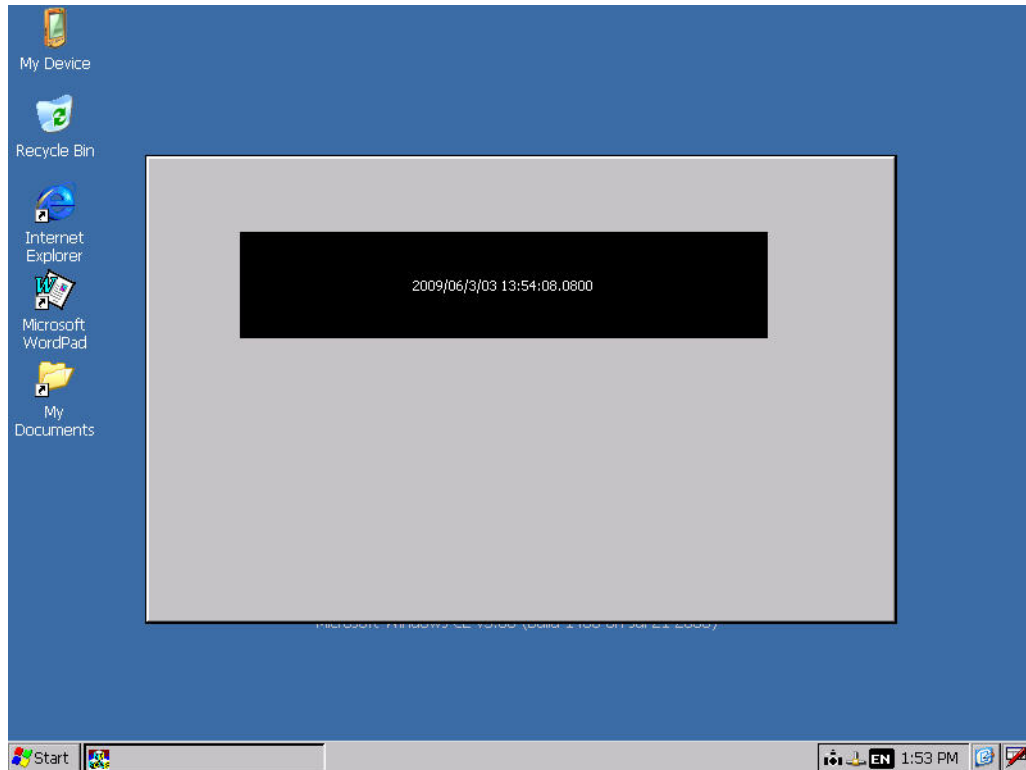
Define a array of 30 TCHARs

**Read the current date and time
and store it in the TCHAR array**

```
TCHAR tcDT[30];  
EzLIB.Get_DT(tcDT, true, true, true, true, true, true, true, true);  
SET_MSG(51,tcDT);
```

**Copy the information of the array to the
MSG register number 51**

STEP 6: Compile the project and download the execution file as described in chapter 2.2.3. The following figure shows the user interface on the PAC. The label object displays the time stamp of every RTSR_7 function call.



2.9 Tutorial 9: Timer

This chapter demonstrates how to start and stop a time countdown. A button will be used for starting and another for stopping the timer. A label displays the current countdown time and a LED informs the user when the countdown has reached zero. The code for controlling the timer and display is implemented in a user thread.

2.9.1 EzHMI design and register settings

Open an EzTemplate project and rename

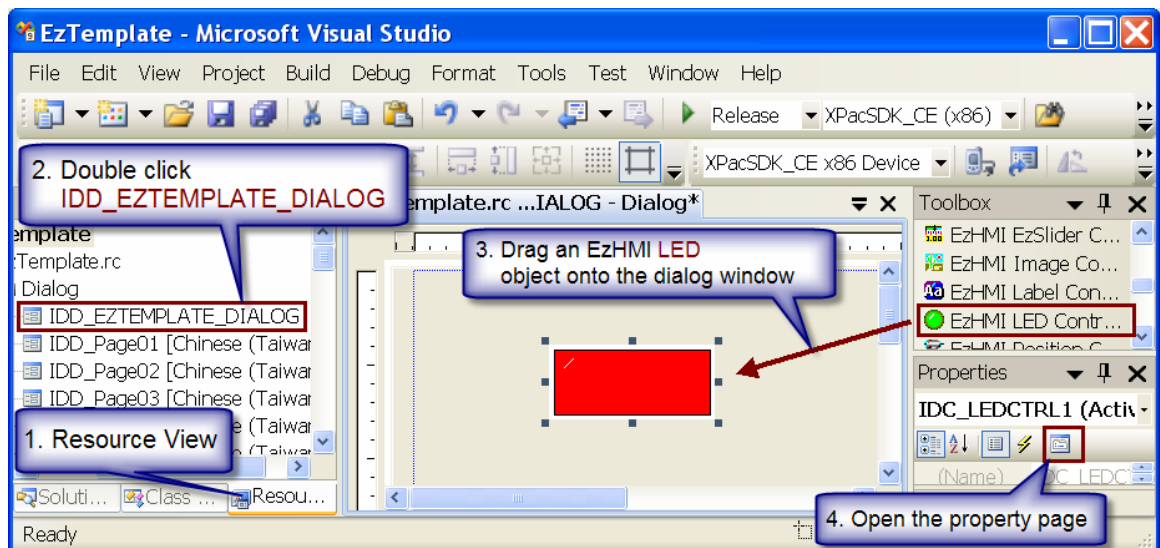
- the file “EzTemplate_800x600” to “Timer” and
- the execution file “EzTemplate.exe” to “Timer.exe”.

Chapter 2.2.1 describes this procedure in detail.

STEP 1: Open the IDD_EZTEMPLATE_DIALOG window

STEP 2: Drag an EzHMI LED object onto the main dialog window

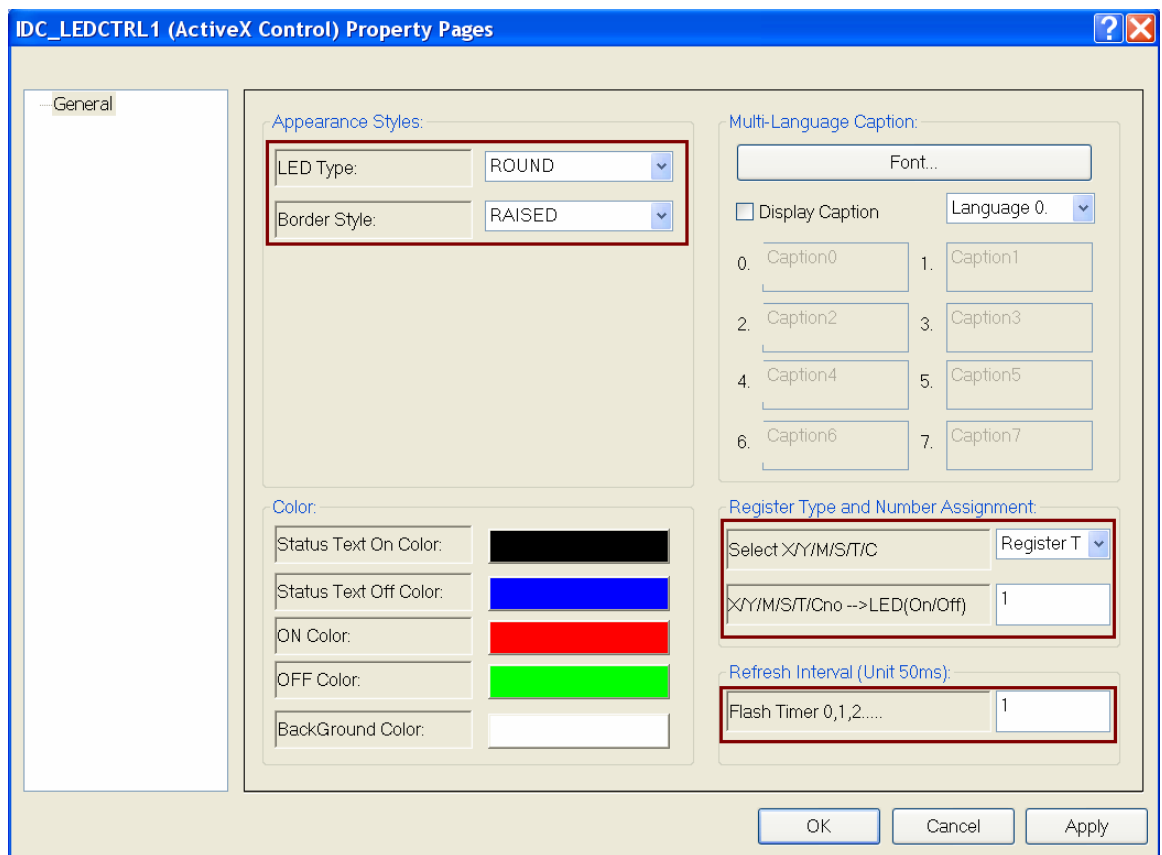
STEP 3: Open the property pages: Right click the LED object and select “Properties”.
Click the “Property Pages” icon in the “Properties” window



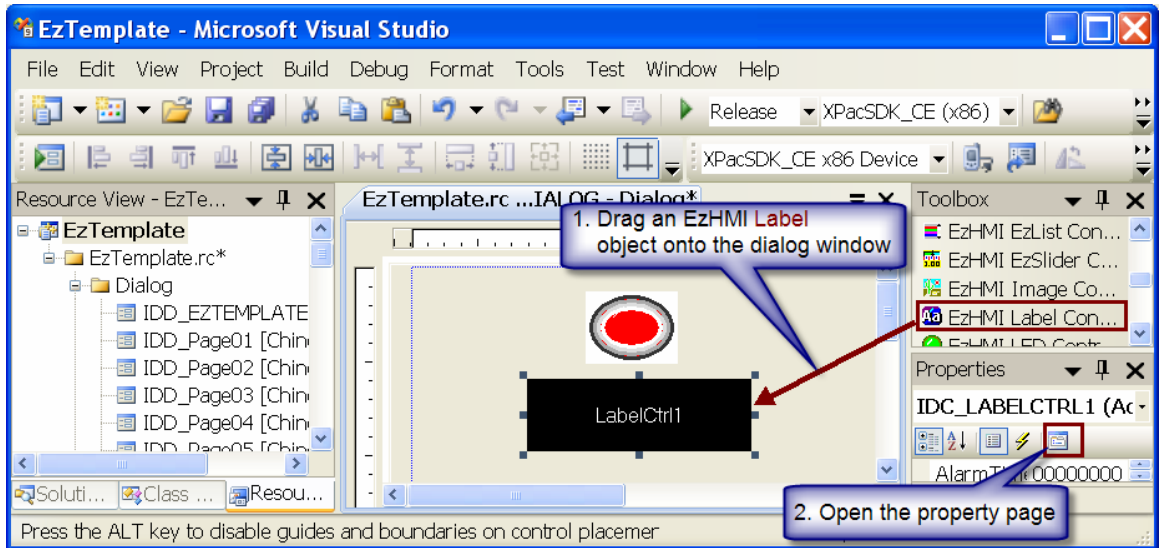
Enter the following property settings:

STEP 4: Select a “ROUND” LED type and a “RAISED” border style

- STEP 5: Link the LED object to the T (timer) register number 1.
- Register type: Select “Register T” for “*Select X/Y/M/S/T/C*”
 - Register number: Enter for “*X/Y/M/S/T/C no →LED(On/Off)*” the register number 1.
- STEP 6: Set the LED object refresh rate to 50 milliseconds.
- Assign “Flash Timer 0,1,2...” the value 1.
- STEP 7: Close the property page by clicking “OK”



- STEP 8: Add an EzHMI Label object to the main dialog window.
- STEP 9: Open the property pages: Right click the Label object and select “Properties”. Click the “Property Pages” icon in the “Properties” window.



Enter the following property settings:

STEP 10: Activate the “**MSG/AI/AO/D/F Enable**” check box. This tells the Label object that it is linked to a register.

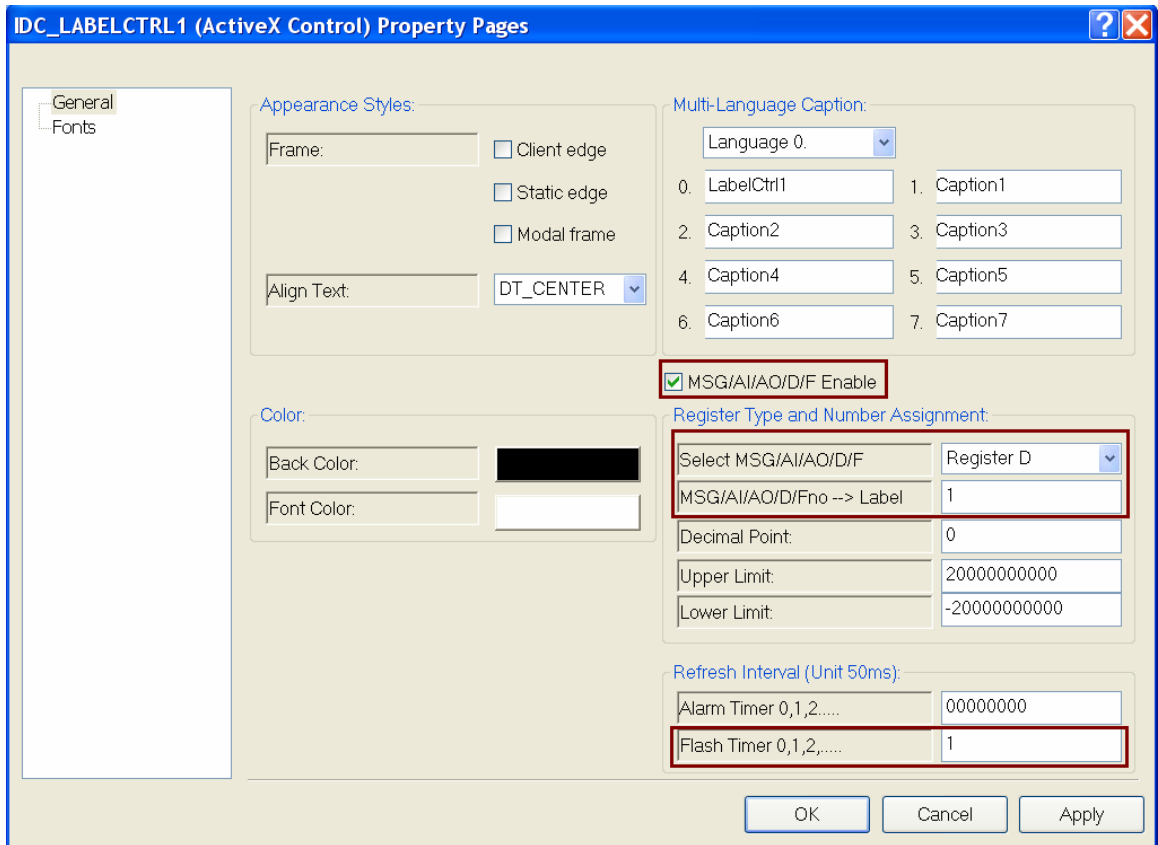
STEP 11: Link the Label object to the D register number 1.

- Register type: Select “Register D” for “**Select MSG/AI/AO/D/F**”
- Register number: Enter for “**MSG/AI/AO/D/Fno →Label**” the number 1.

STEP 12: Set the Label object refresh rate to 50 milliseconds.

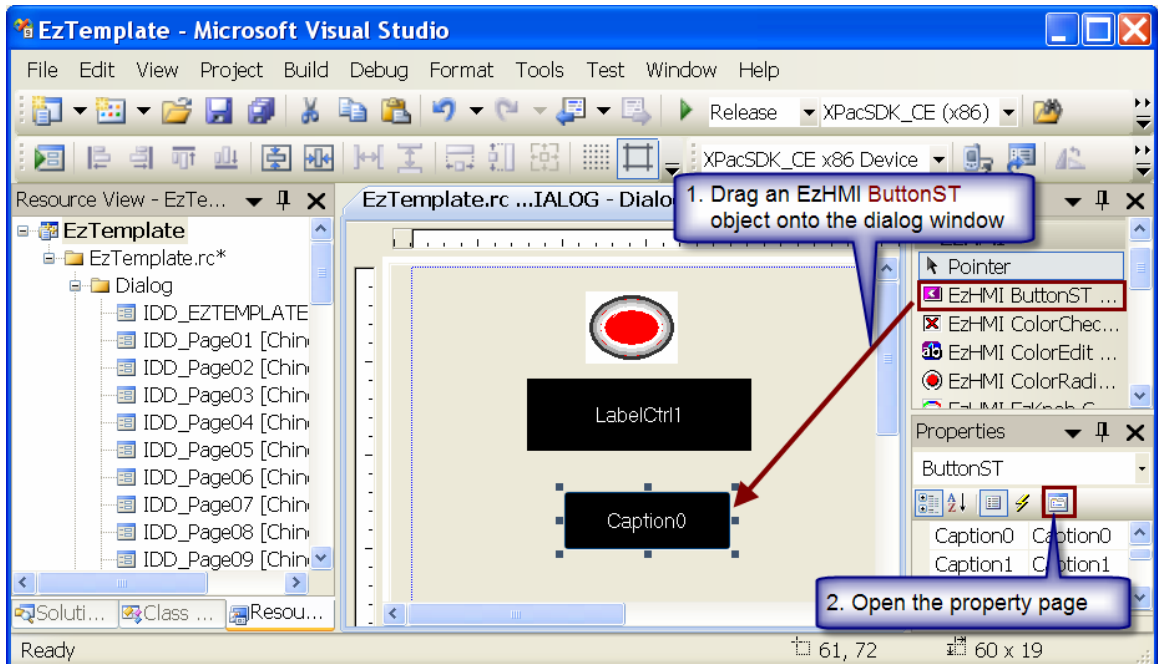
- Assign “Flash Timer 0,1,2...” the value 1.

STEP 13: Close the property page by clicking “OK”



STEP 14: Add an EzHMI ButtonST object to the main dialog window.

STEP 15: Open the property pages: Right click the ButtonST object and select “Properties”. Click the “Property Pages” icon in the “Properties” window.



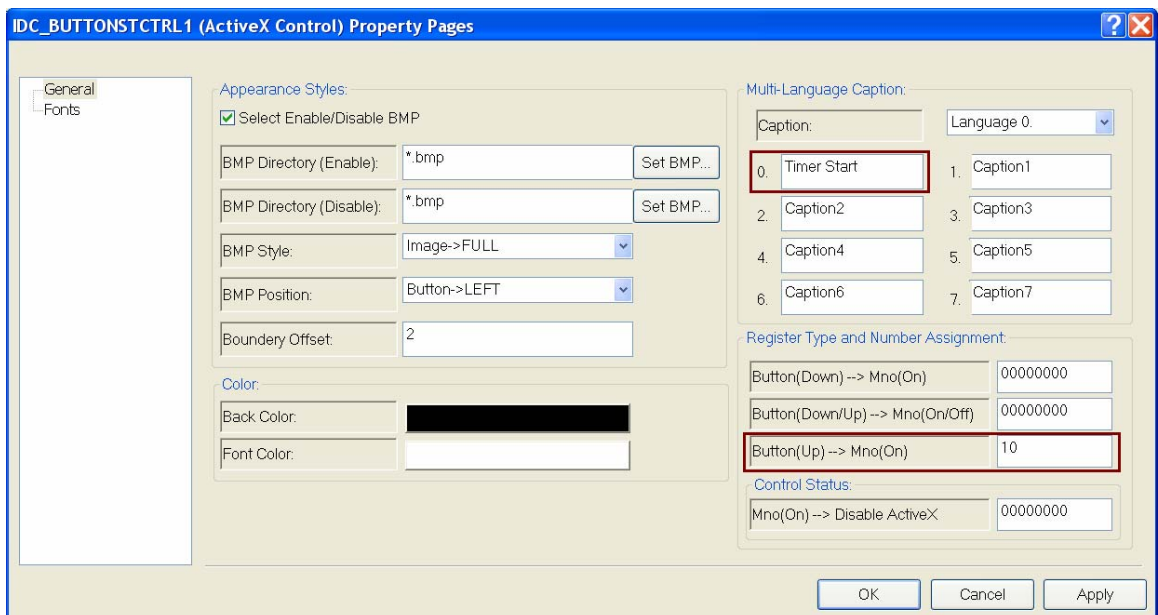
Enter the following property settings:

STEP 13: Change the button label to “Timer Start”.

STEP 14: Link the button up event to the M register number **10**. Every time when the button is released the M register 10 is set to true.

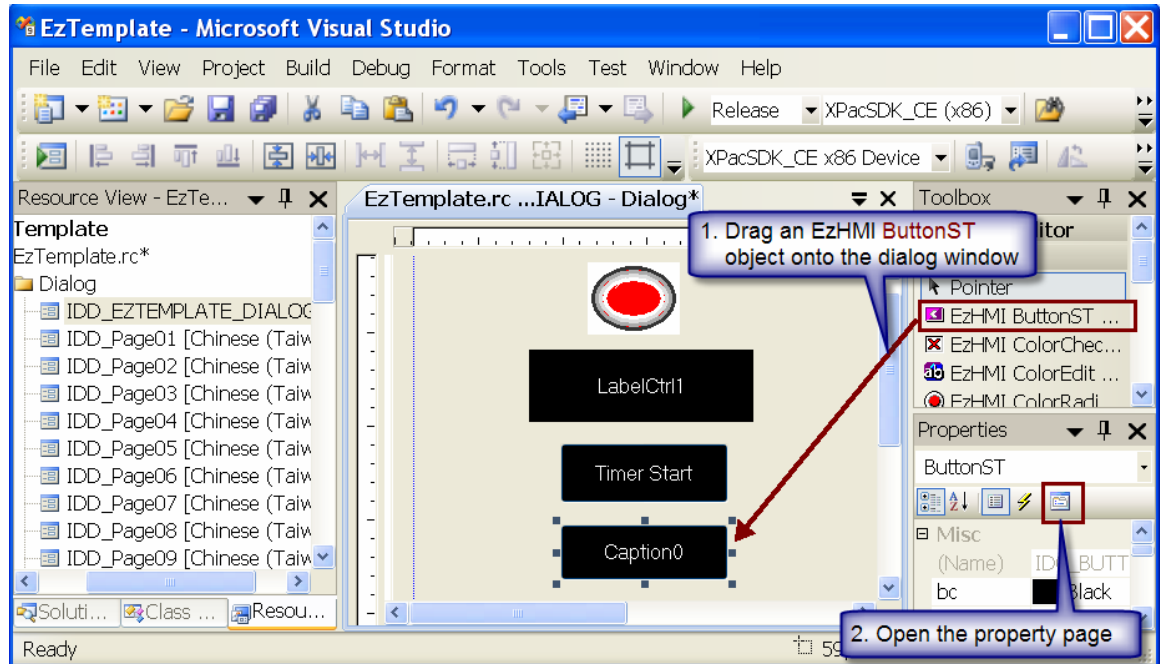
- Register number: Enter for “*Button(Up) → Mno(On)*” the number **10**.

STEP 15: Close the property page by clicking “OK”



STEP 16: Add another EzHMI ButtonST object to the main dialog window.

STEP 17: Open the property pages: Right click the ButtonST object and select “Properties”. Click the “Property Pages” icon in the “Properties” window.

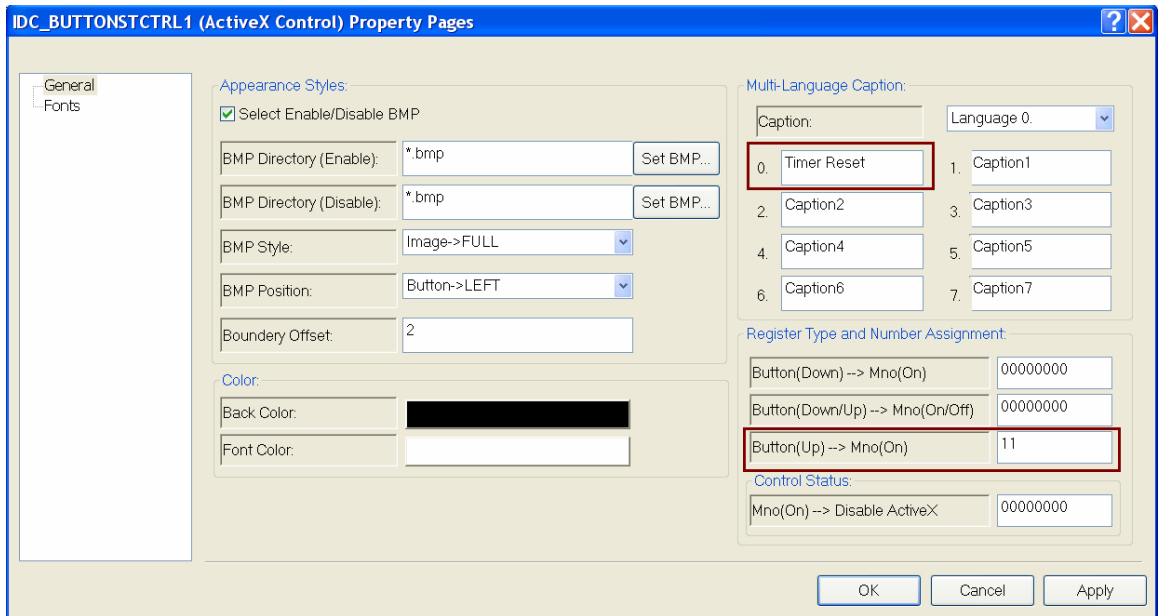


Enter the following property settings:

STEP 16: Change the label of the button to “Timer Reset”.

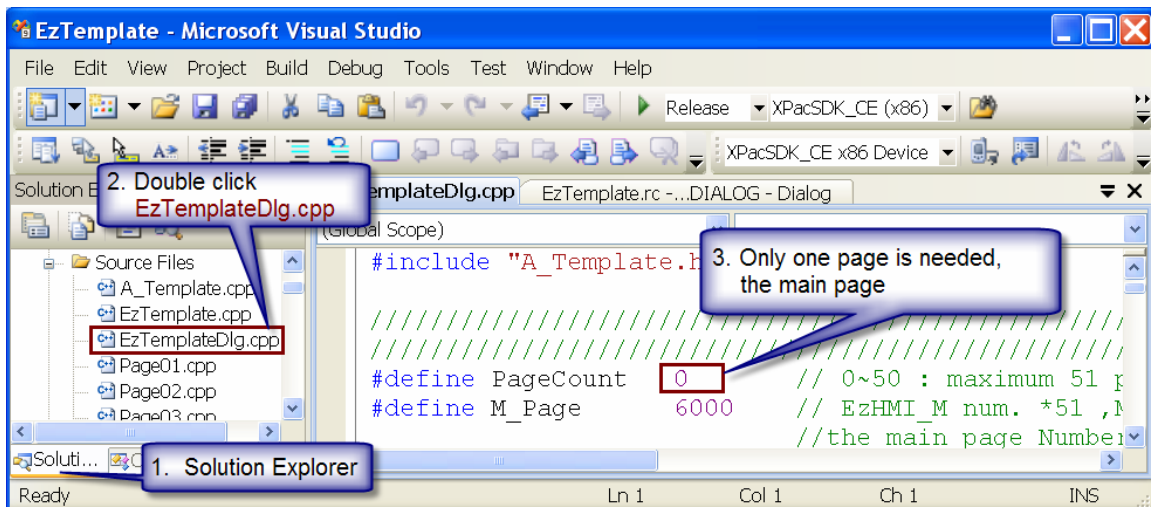
STEP 17: Link the button up event to the M register number 11. Every time when the button is released the M register 11 is set to true.
– Register number: Enter for “*Button(Up) → Mno(On)*” the number 11.

STEP 18: Close the property page by clicking “OK”

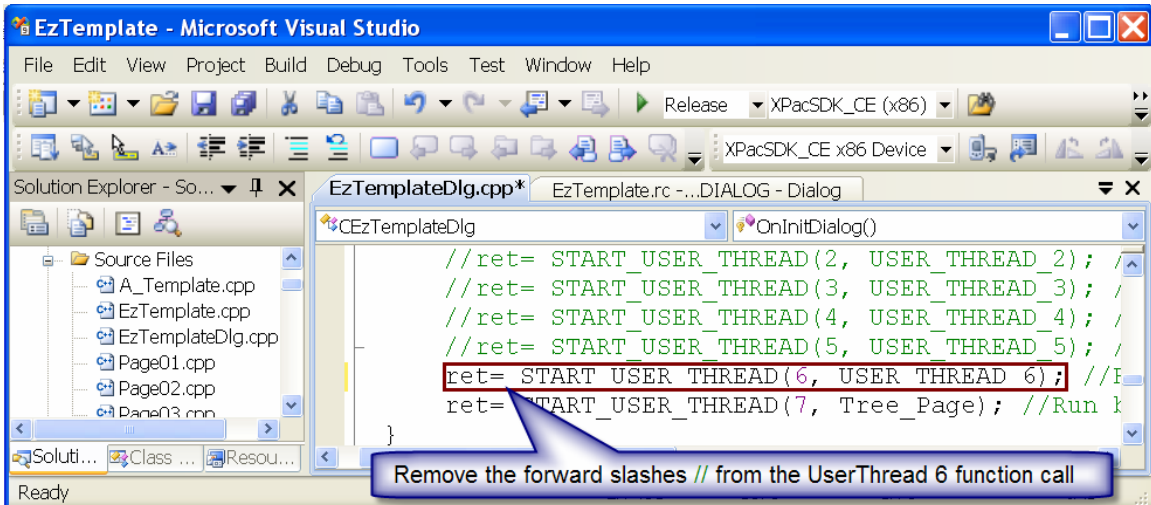


2.9.2 UserThread activation and code implementation

STEP 1: Only the main page is needed for this example therefore assign a 0 to the PageCount definition.

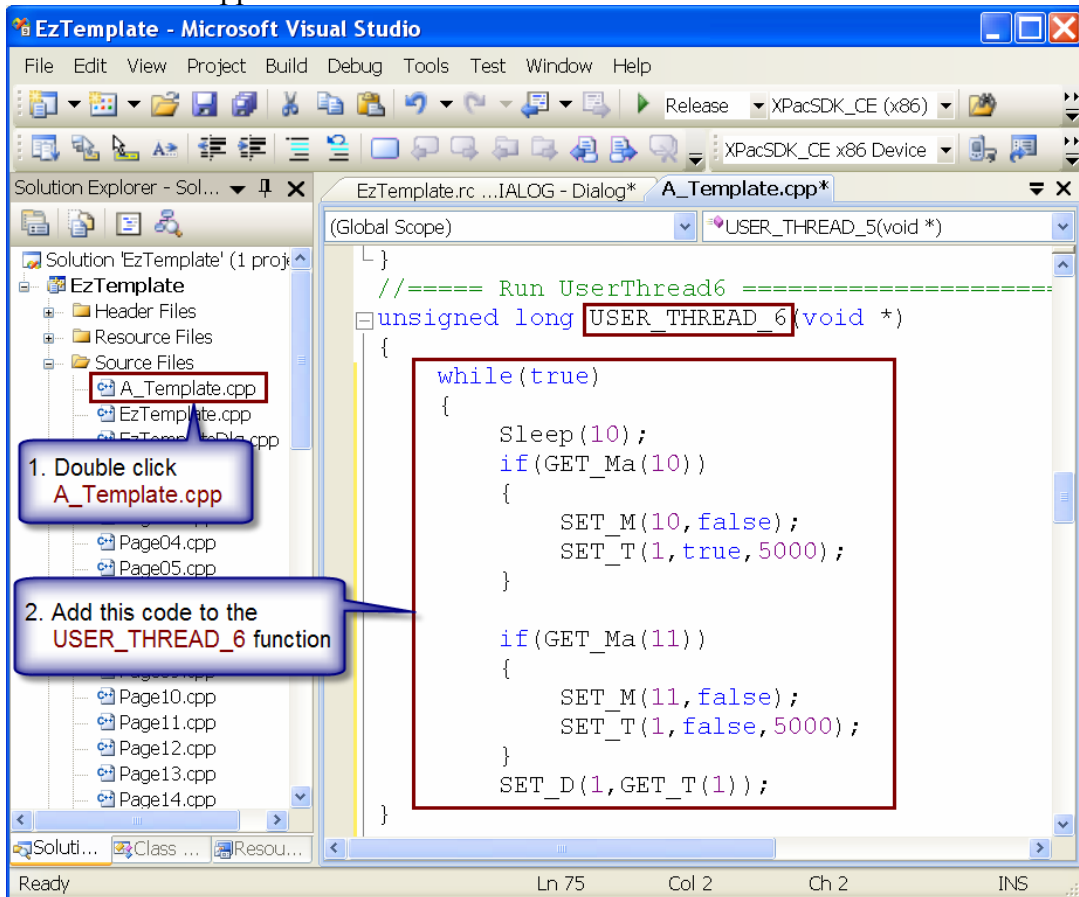


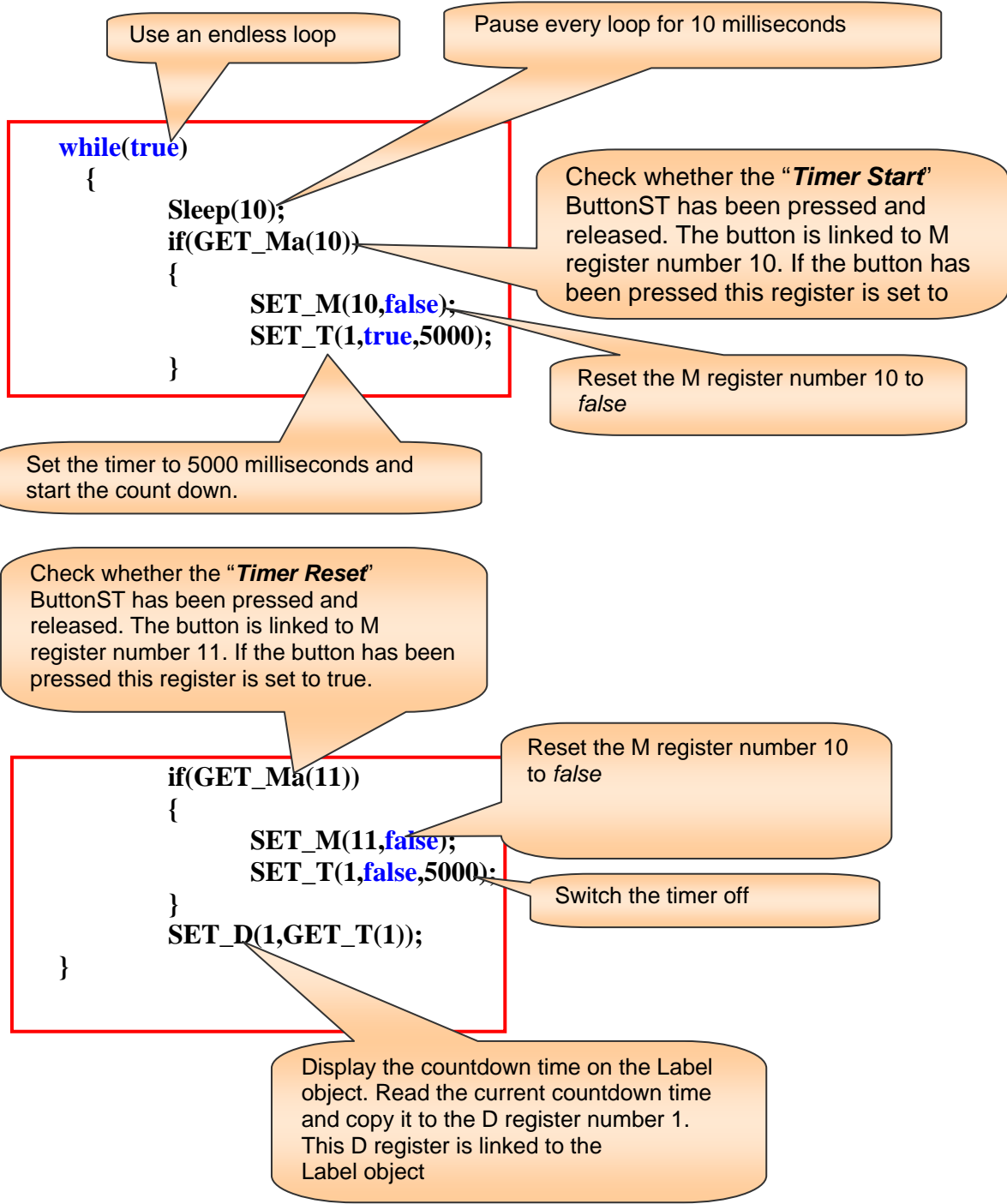
STEP 2: The User_Thread_6 will be used to handle the timer event. It is therefore necessary to remove the forward slashes from the USER_THREAD_6 function call.



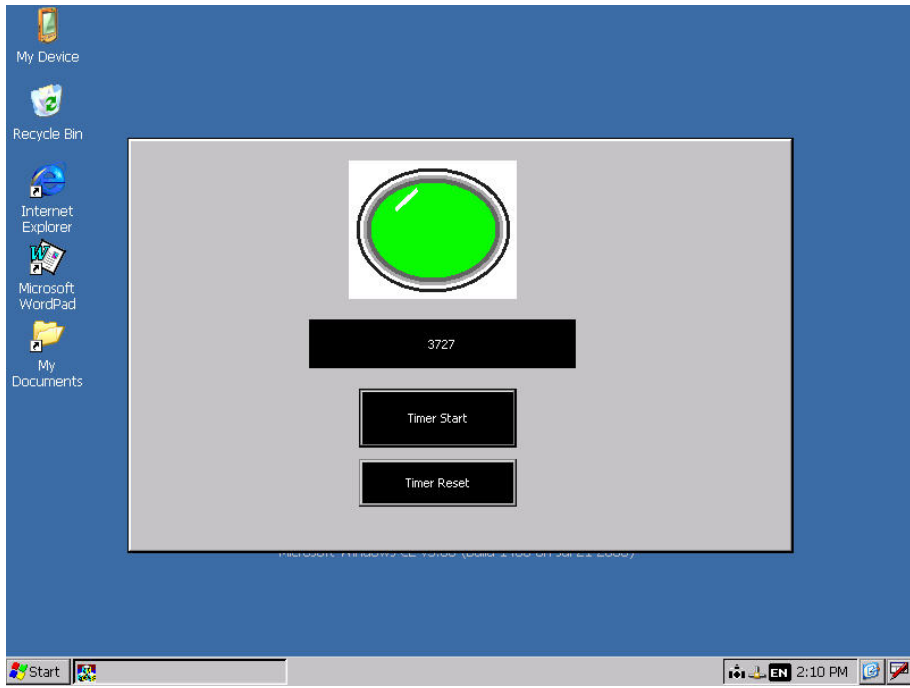
STEP 3: Add the following code to USER_THREAD_6 function

The indefinite loop checks every 10 millisecond whether the *“Timer Start”* button has been clicked. If the button has been clicked its M register is reset to false inside the loop and the time countdown is started. The Label object is updated in every loop with the current countdown time. The time count down is stopped when the *“Time Reset”* button is clicked and released.

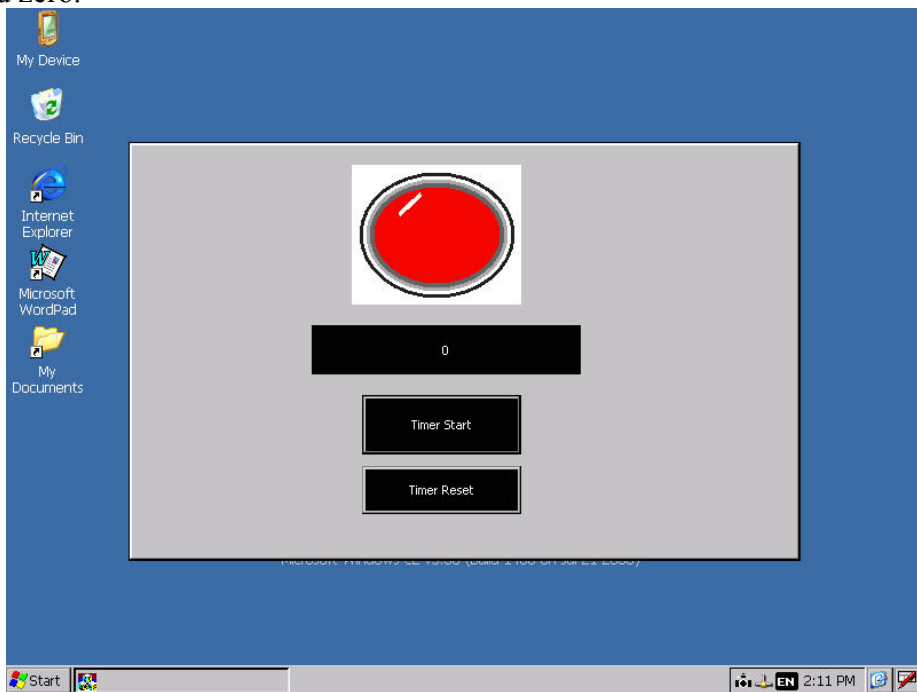




STEP 7: Compile the project and download the execution file as described in chapter 2.2.3. The following figure shows the user interface on the PAC. Clicking the “Timer Start” button start the countdown. The label displays the countdown time. Clicking the “**Time Reset**” button stops countdown and sets the countdown time to zero.



The LED which is linked to the time register changes its color when the countdown has reached zero.



2.10 Tutorial 10: Counter

The counter API provided by ICPDAS counts the number of falling edges of on/off (false/true) cycles. The counter is normally being used in conjunction with a digital input channel to count the number of status changes. The counter API is intended for modules which have no internal counter. The API has to be assigned an initial value from which to count backwards. Once the counter has reach zero the respective C register will be set to true and the counting procedure stops. Only by resetting the counter can a new counting process be started.

This example is designed to be device independent and therefore the counter API is not directly linked to an IO register but the on/off cycles are simulated by assigning the API alternating a true and false status. Two buttons are used, one for starting and counting the number of button clicks and the other for stopping and resetting the counter. A label displays the current count value and a LED changes its state as soon as the counter reaches the value zero.

2.10.1 EzHMI design and register settings

Open an EzTemplate project and rename

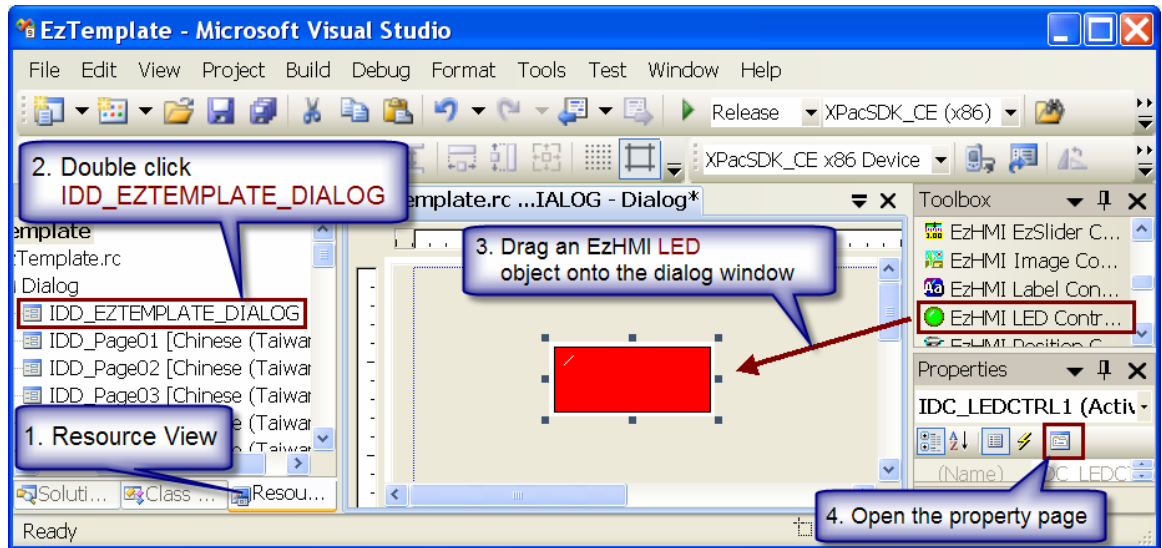
- the file “EzTemplate_800x600” to “*Counter*” and
- the execution file “EzTemplate.exe” to “*Counter.exe*”.

Chapter 2.2.1 describes this procedure in detail.

STEP 1: Open the IDD_EZTEMPLATE_DIALOG window

STEP 2: Drag an EzHMI LED object onto the main dialog window

STEP 3: Open the property pages: Right click the LED object and select “Properties”.
Click the “Property Pages” icon in the “Properties” window



Enter the following property settings:

STEP 4: Select a “*ROUND*” LED type and a “*RAISED*” border style

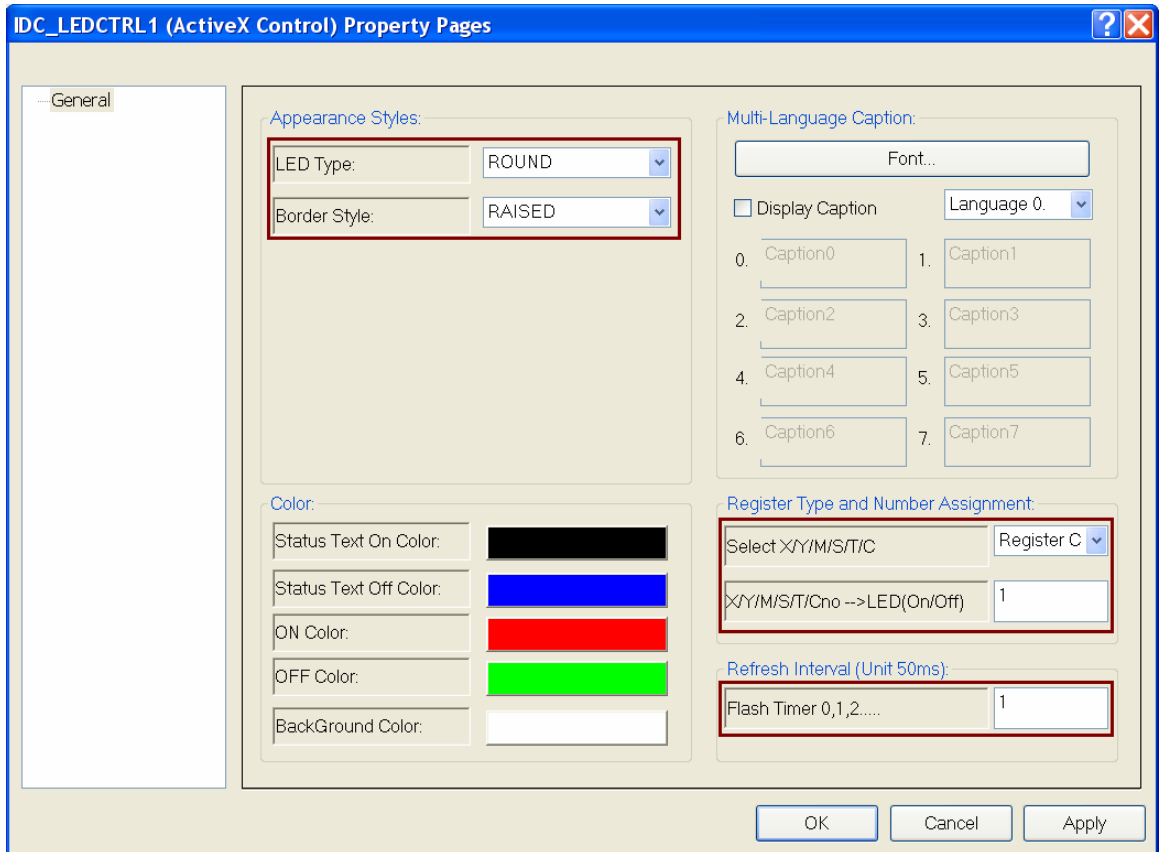
STEP 5: Link the LED object to the C (counter) register number 1.

- Register type: Select “*Register C*” for “*Select X/Y/M/S/T/C*”
- Register number: Enter for “*X/Y/M/S/T/C no →LED(On/Off)*” the register number *1*.

STEP 6: Set the LED object refresh rate to 50 milliseconds.

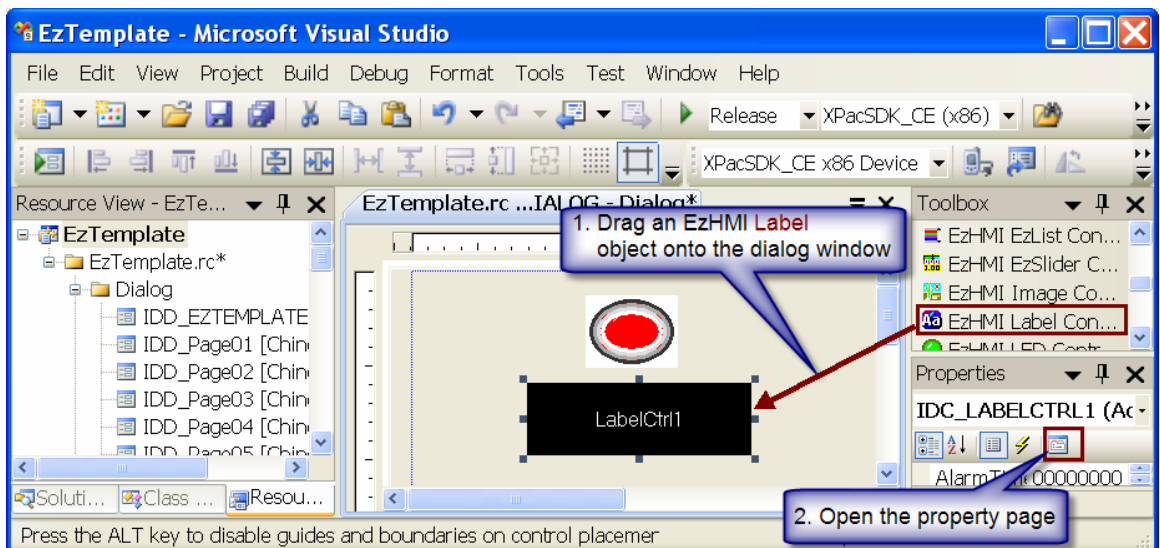
- Assign “Flash Timer 0,1,2...” the value *1*.

STEP 7: Close the property page by clicking “OK”



STEP 8: Add an EzHMI Label object to the main dialog window.

STEP 9: Open the property pages: Right click the Label object and select “Properties”.
Click the “Property Pages” icon in the “Properties” window.



Enter the following property settings:

STEP 10: Activate the “**MSG/AI/AO/D/F Enable**” check box. This tells the Label object that it is linked to a register.

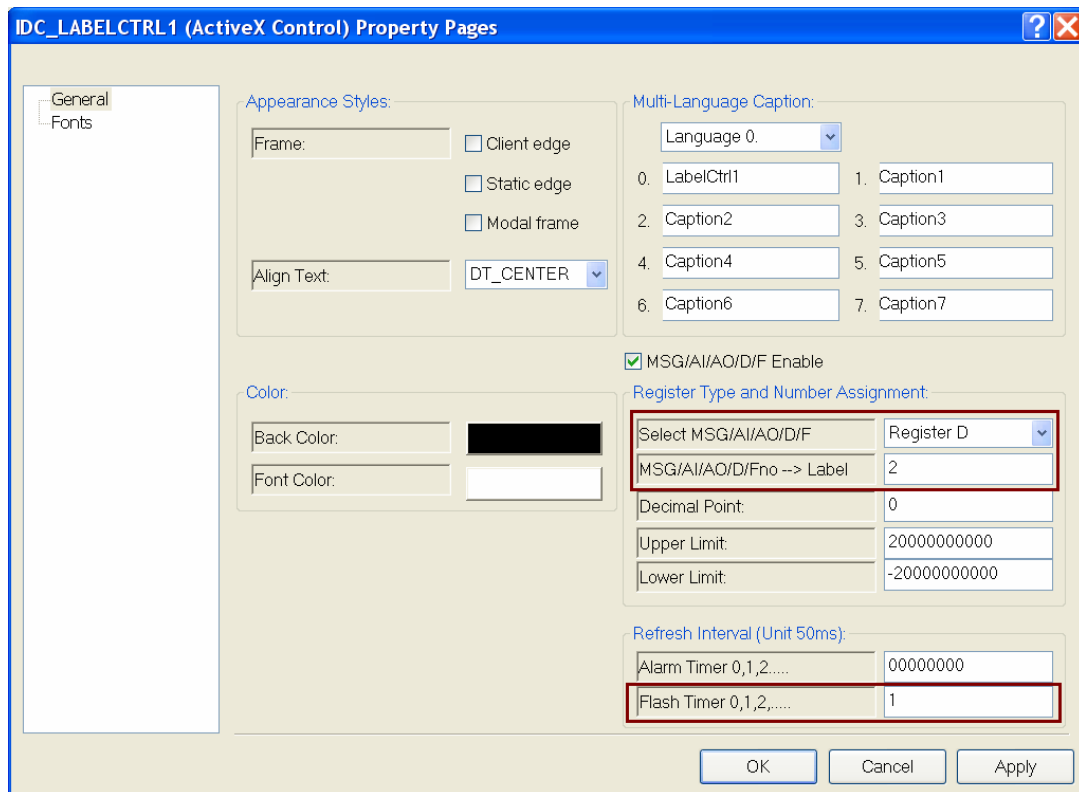
STEP 11: Link the Label object to the D register number 2.

- Register type: Select “**Register D**” for “**Select MSG/AI/AO/D/F**”
- Register number: Enter for “**MSG/AI/AO/D/Fno →Label**” the number **1**.

STEP 12: Set the Label object refresh rate to 50 milliseconds.

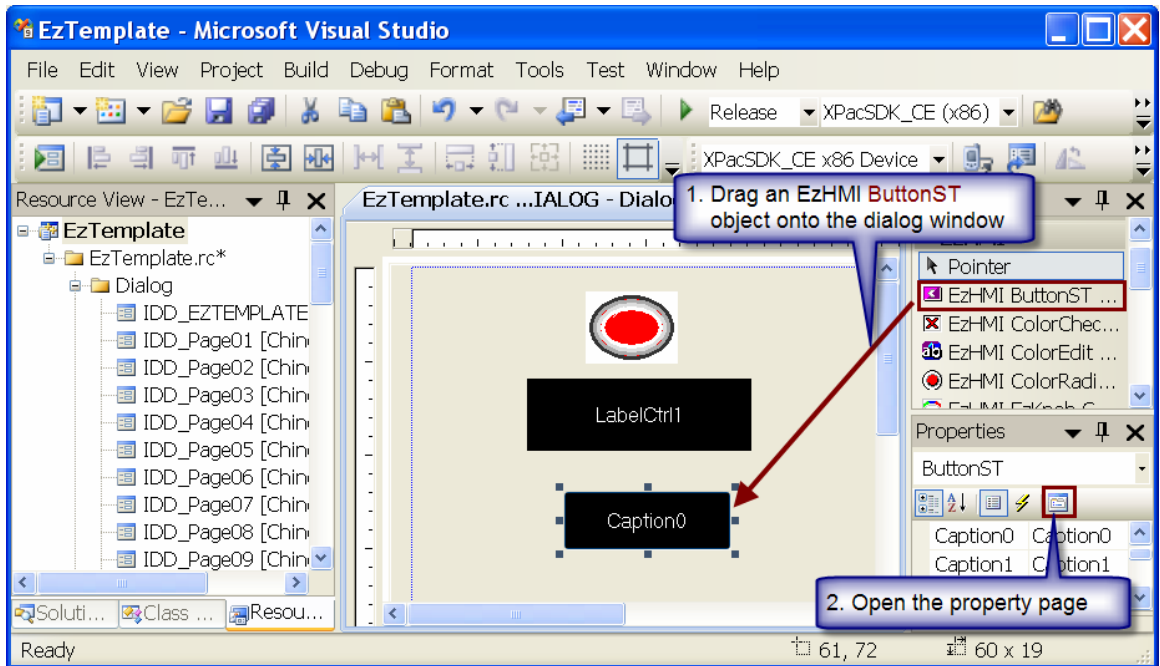
- Assign “Flash Timer 0,1,2...” the value 1.

STEP 13: Close the property page by clicking “OK”



STEP 14: Add an EzHMI ButtonST object to the main dialog window.

STEP 15: Open the property pages: Right click the ButtonST object and select “Properties”. Click the “Property Pages” icon in the “Properties” window.



Enter the following property settings:

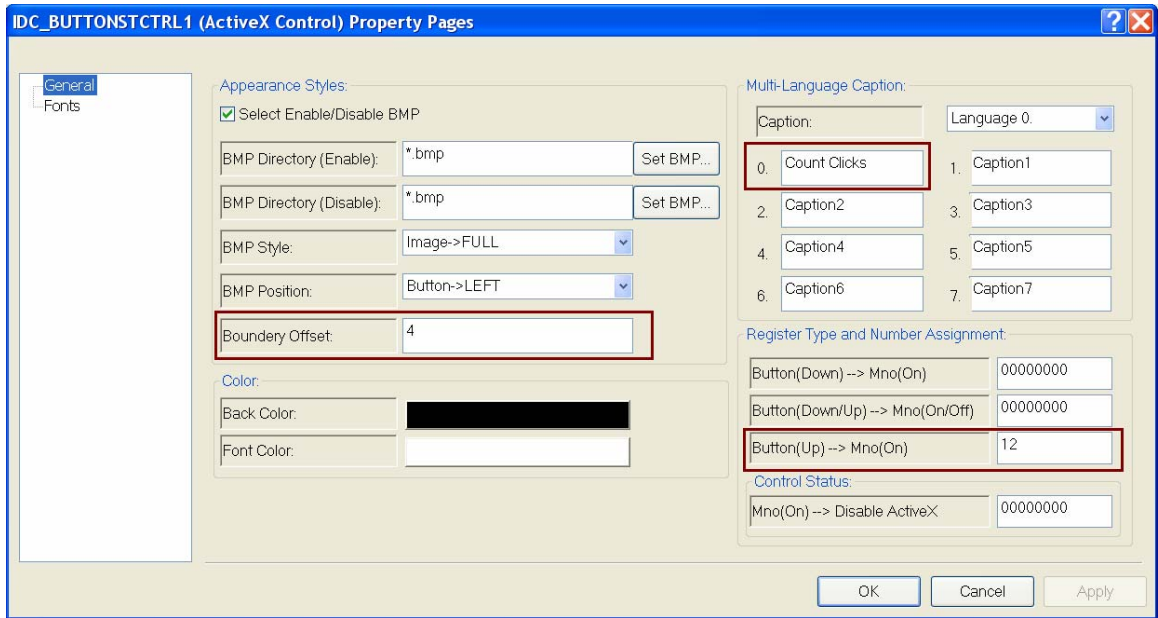
STEP 19: Change the button boundary to 4 pixels

STEP 20: Change the button label to “*Count Clicks*”.

STEP 21: Link the button up event to the M register number **12**. Every time when the button is released the M register 12 is set to true.

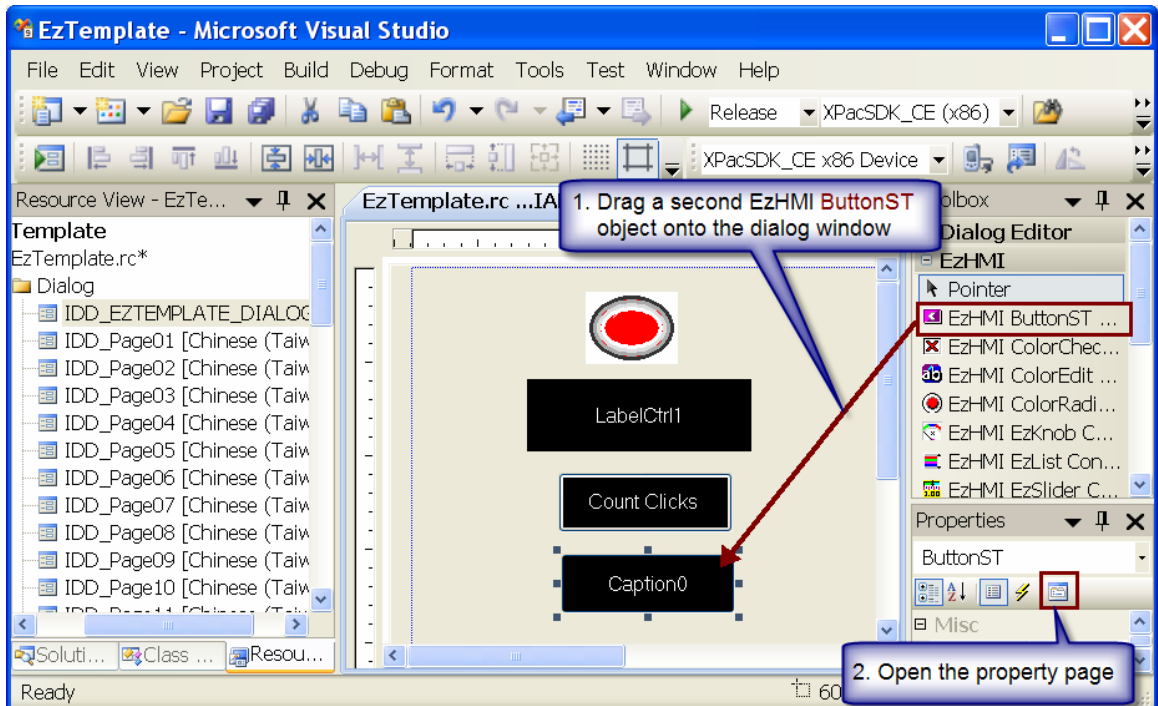
- Register number: Enter for “*Button(Up) →Mno(On)*” the number **12**.

STEP 22: Close the property page by clicking “OK”.



STEP 16: Add another EzHMI ButtonST object to the main dialog window.

STEP 17: Open the property pages: Right click the ButtonST object and select “Properties”. Click the “Property Pages” icon in the “Properties” window.



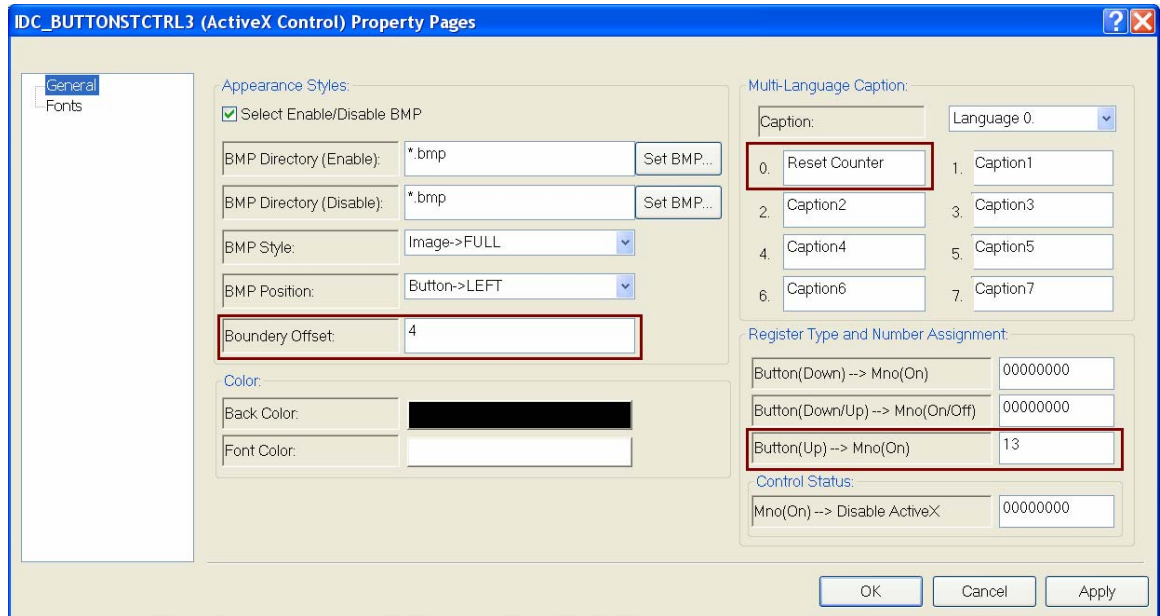
Enter the following property settings:

STEP 23: Change the label of the button to “Reset Counter”.

STEP 24: Link the button up event to the M register number 13. Every time when the button is released the M register 13 is set to true.

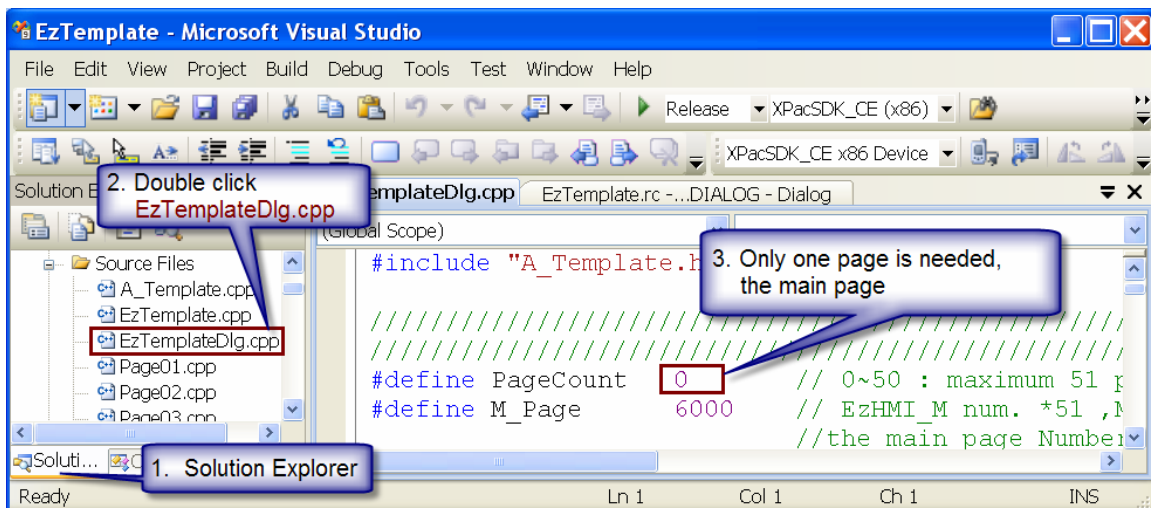
- Register number: Enter for “**Button(Up) → Mno(On)**” the number **13**.

STEP 25: Close the property page by clicking “OK”

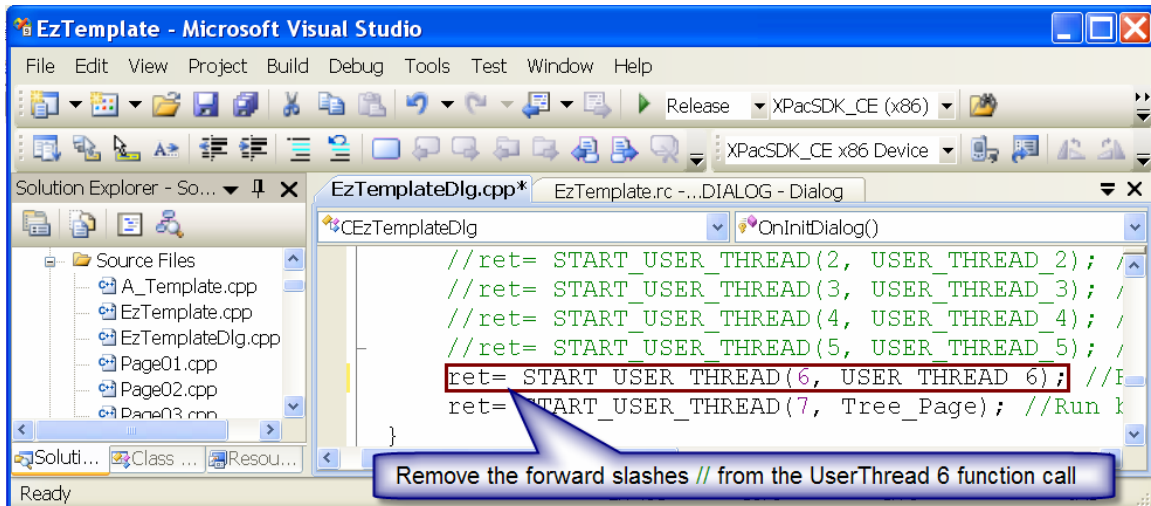


2.10.2 UserThread activation and code implementation

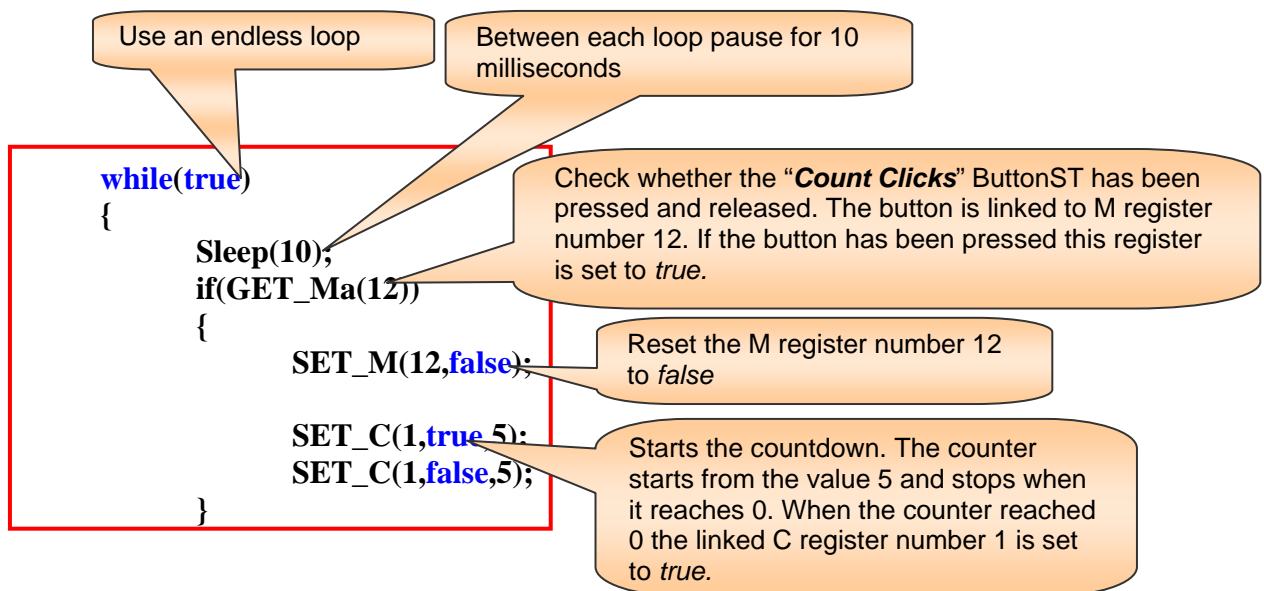
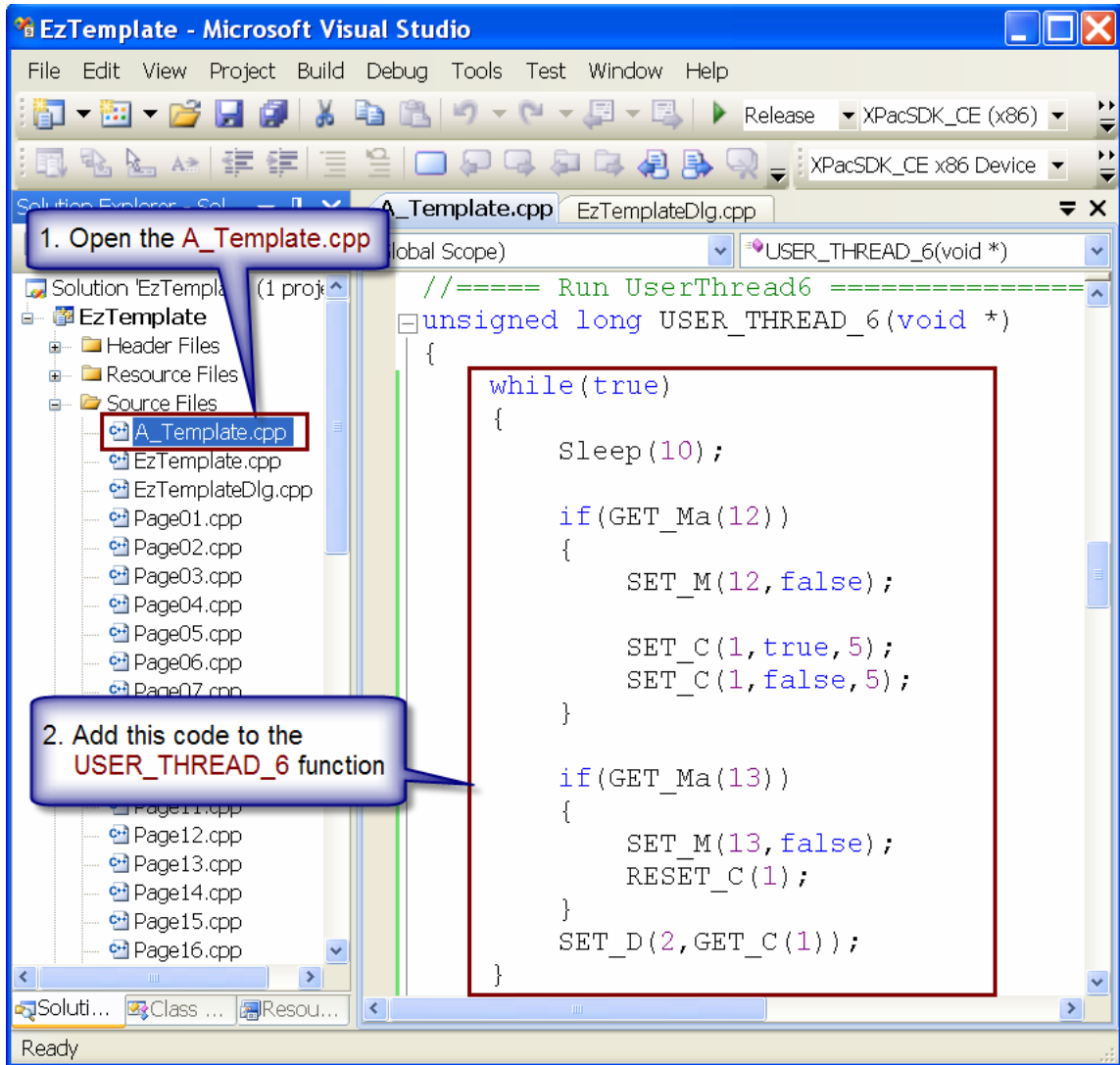
STEP 1: Only the main page is needed for this example therefore assign a zero to the PageCount definition.



STEP 2: The User_Thread_6 will be used to handle the counter event and to update the EzHMI relevant registers. Remove the forward slashes from the USER_THREAD_6 function activation.



STEP 3: Add the following code to USER_THREAD_6 function
The endless loop stops in every loop for 10 milliseconds before it continues to check whether the “*Count Clicks*” button has been clicked. If the button has been clicked its associated M register is reset to false inside the loop and one on/off cycle is generated. At the end of the loop the current counter value is copied to the D register number 2 and afterwards being displayed by the label which is directly linked to this register.
The second if block is being executed after the “*Reset Counter*” button has been pressed and released. This action stops and resets the counter.



The counter is only counting backwards when it detects a falling edge. Therefore the counter will only count down if it has detected a true and a false.

Check whether the “**Reset Counter**” ButtonST has been pressed and released. The button is linked to M register number 13. After the button has been pressed this register is set to *true*.

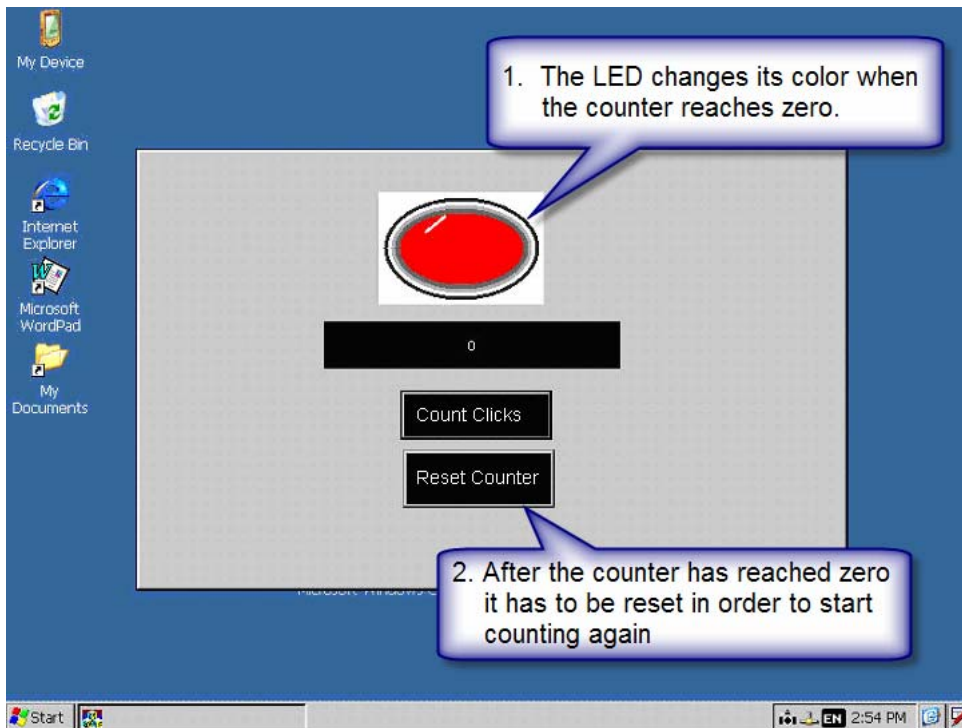
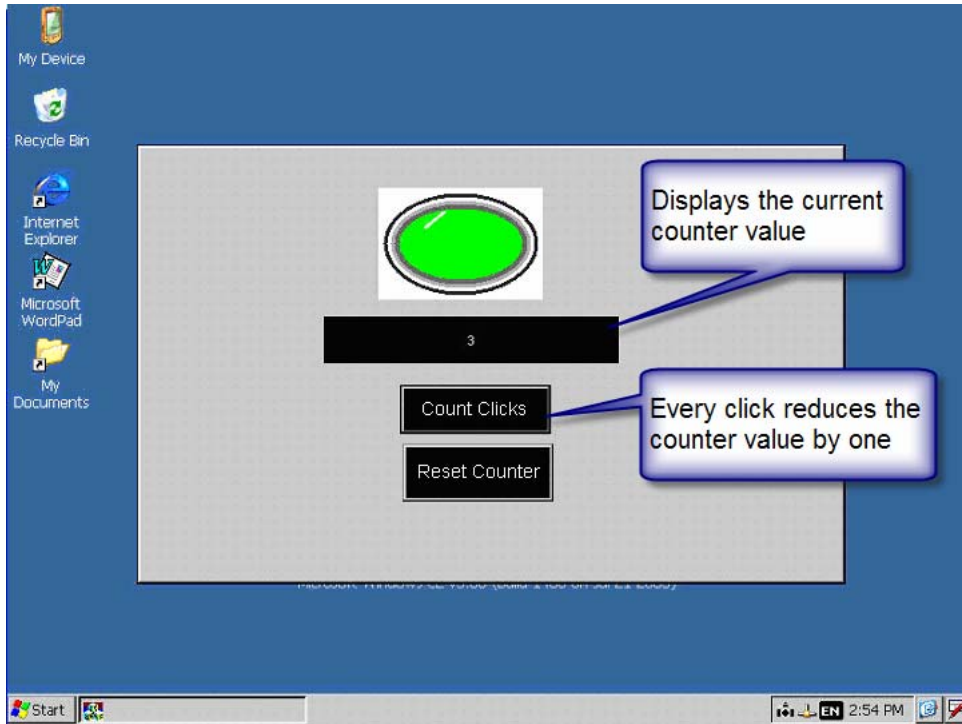
```
if(GET_Ma(13))  
{  
    SET_M(13,false);  
    RESET_C(1);  
}  
SET_D(2,GET_C(1));  
}
```

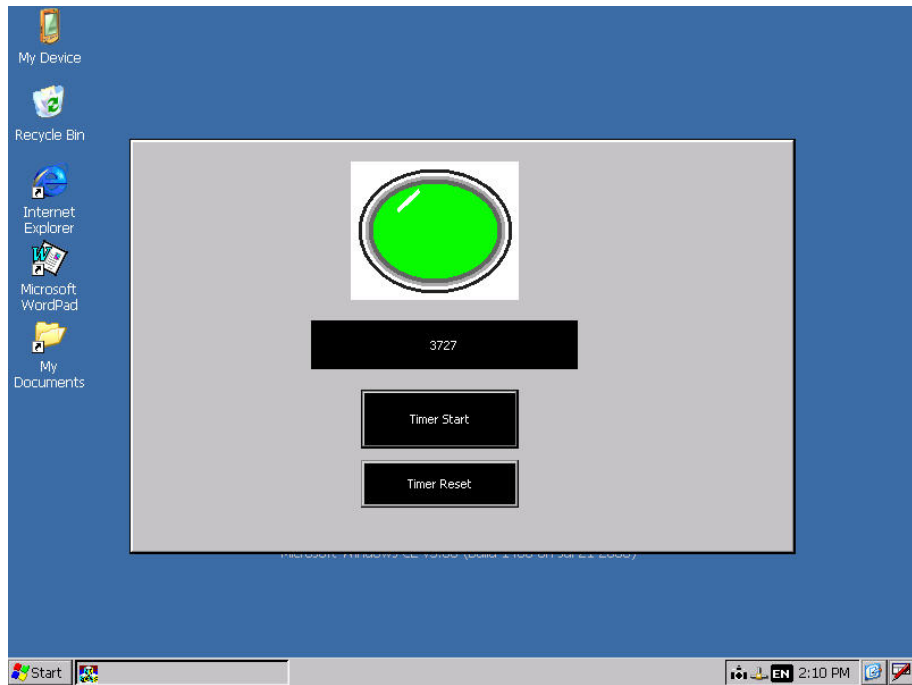
Reset the M register number 13 to *false*

Reset the counter

The current counter value is copied to D register number 2. The label object will display this value as it is linked to this register.

STEP 8: Compile the project and download the execution file as described in chapter 2.2.3. The following figure shows the user interface on the PAC. Every click on the “Count Clicks” button reduces the counter value by one. The label displays the countdown value. Clicking the “**Reset Counter**” button stops the countdown and sets the counter to zero.





The LED which is linked to the time register changes its color when the countdown has reached zero.

2.11 Tutorial 11: Advanced Encryption Standard (AES)

This example describes how the program developer can create interface for entering the registration code key.

2.11.1 EzHMI design and register settings

Open an EzTemplate project and rename

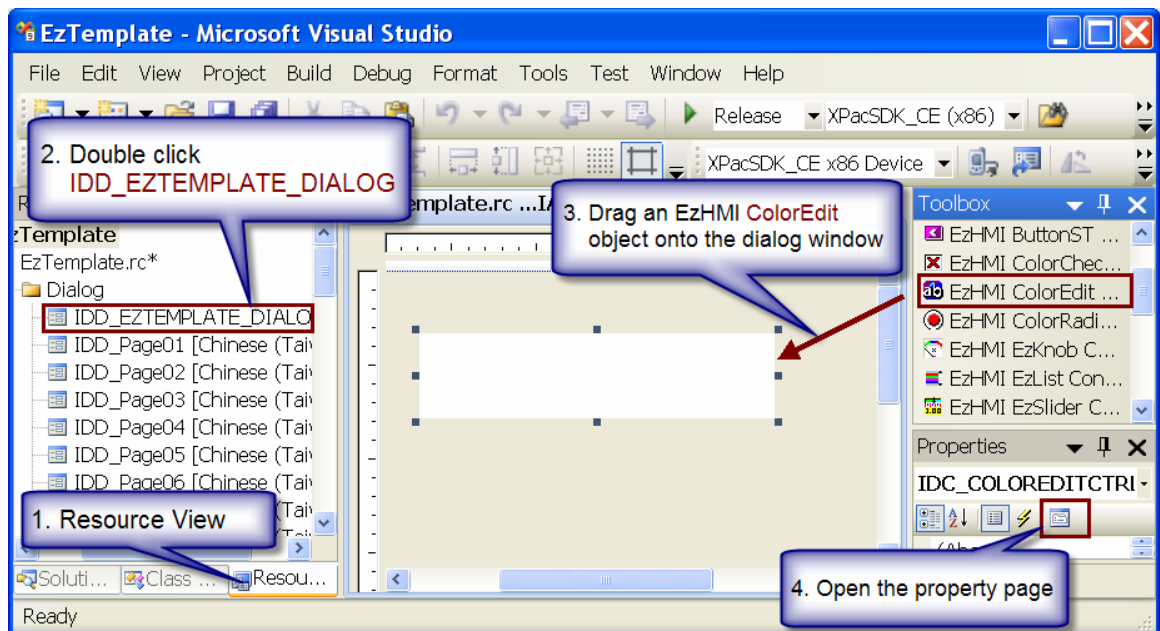
- the file “EzTemplate_800x600” to “**AES**” and
- the execution file “EzTemplate.exe” to “**AES.exe**”.

Chapter 2.2.1 describes this procedure in detail.

STEP 1: Open the IDD_EZTEMPLATE_DIALOG window

STEP 2: Drag an EzHMI ColorEdit object onto the main dialog window

STEP 3: Open the property pages: Right click the ColorEdit object and select “Properties”. Click the “Property Pages” icon in the “Properties” window



Do the following property settings:

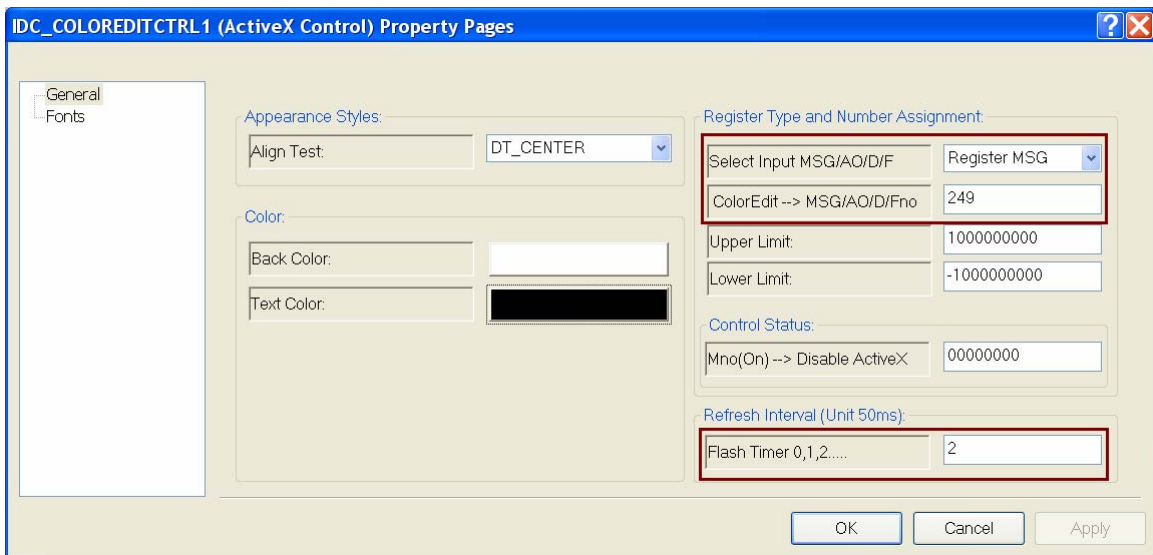
- STEP 4: Link the ColorEdit object to the MSG (message) register number 249.
- Register type: Select “**Register MSG**” for “**Select Input MSG/AO/D/F**”

- Register number: Enter for “*ColorEditX → MSG/AO/D/F no*” the register number **249**.
- The M register numbers 245 to 249 are reserved for passwords. This means when the ColorEdit object is linked to one of these registers the text strings store in this registers will not be visible in the ColorEdit but only stars “*****” are displayed. Text entered in the ColorEdit are also as a sequence stars.

STEP 5: Set the ColorEdit object refresh rate to 100 milliseconds.

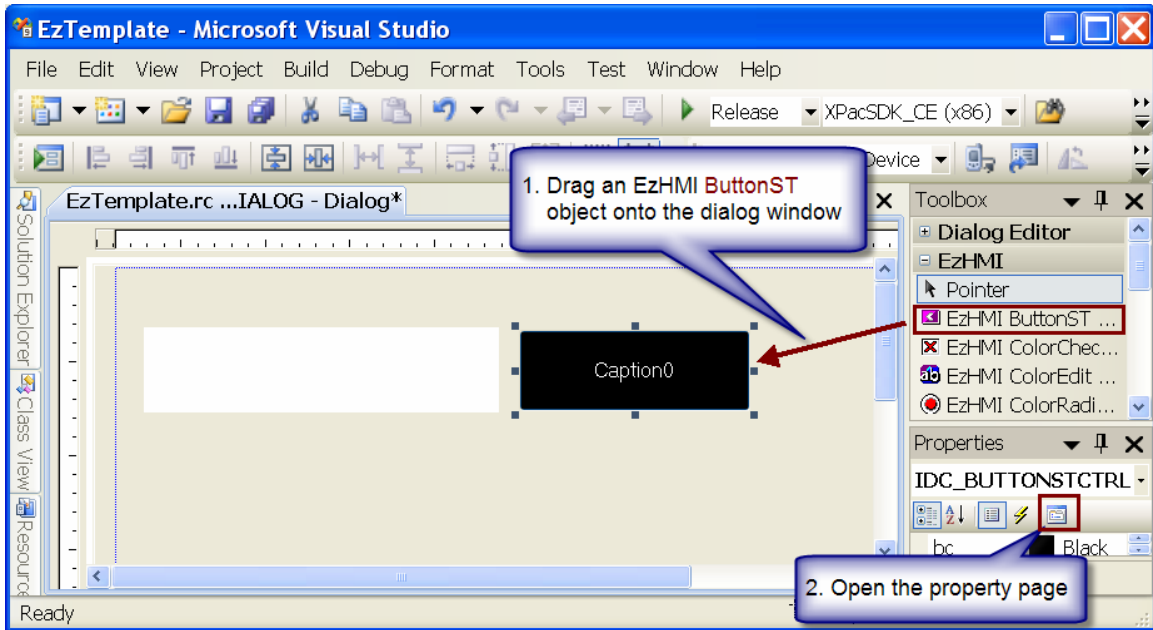
- Assign “*Flash Timer 0,1,2...*” the value **2**.

STEP 6: Close the property page by clicking “OK”



STEP 7: Drag an EzHMI ButtonST object onto the main dialog window

STEP 8: Open the property pages: Right click the ButtonST object and select “Properties”. Click the “Property Pages” icon in the “Properties” window



Enter the following property settings:

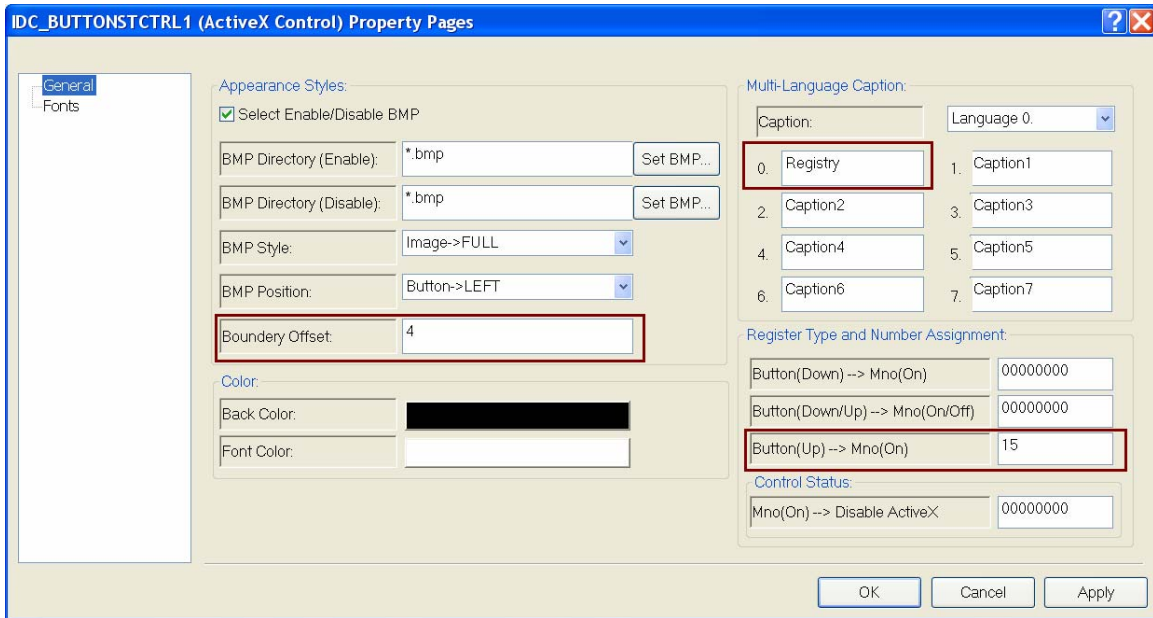
STEP 9: Change the button boundary to 4 pixels

STEP 10: Change the button caption to “**Registry**”.

STEP 11: Link the button up event to the M register number **15**. Every time when the button is released the M register 15 is set to true.

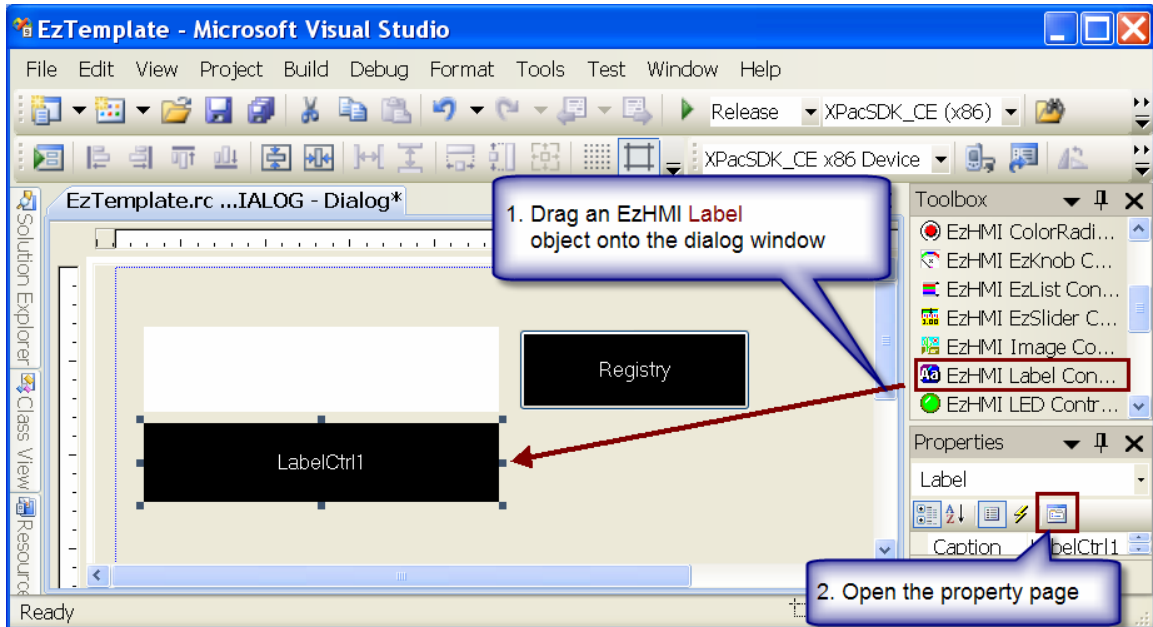
– Register number: Enter for “**Button(Up) →Mno(On)**” the number **15**.

STEP 12: Close the property page by clicking “OK”.



STEP 13: Drag an EzHMI ButtonST object onto the main dialog window

STEP 14: Open the property pages: Right click the ButtonST object and select “Properties”. Click the “Property Pages” icon in the “Properties” window



Enter the following property settings:

STEP 15: Activate the “MSG/AI/AO/D/F Enable” check box. This tells the Label object that it is linked to a register.

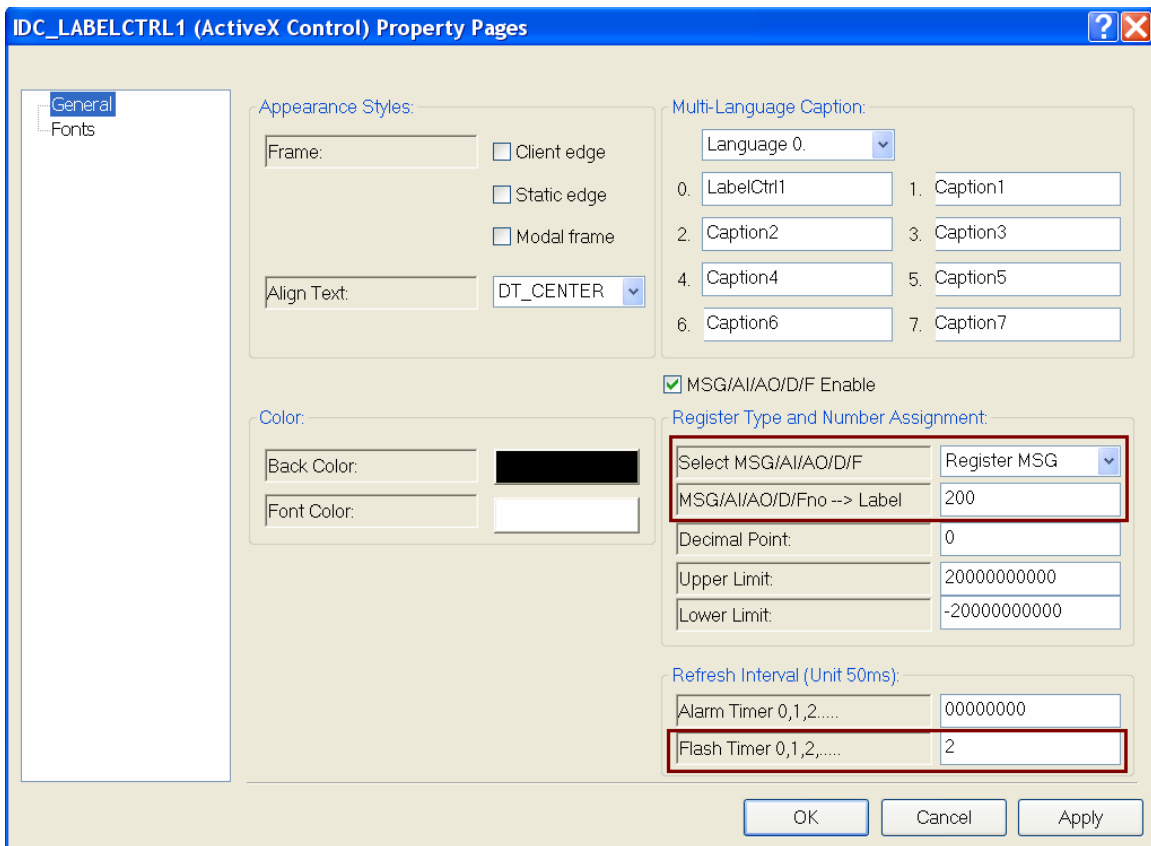
STEP 16: Link the Label object to the MSG register number 200.

- Register type: Select “**Register MSG**” for “**Select MSG/AI/AO/D/F**”
- Register number: Enter for “**MSG/AI/AO/D/Fno →Label**” the number **200**.

STEP 17: Set the Label object refresh rate to 100 milliseconds.

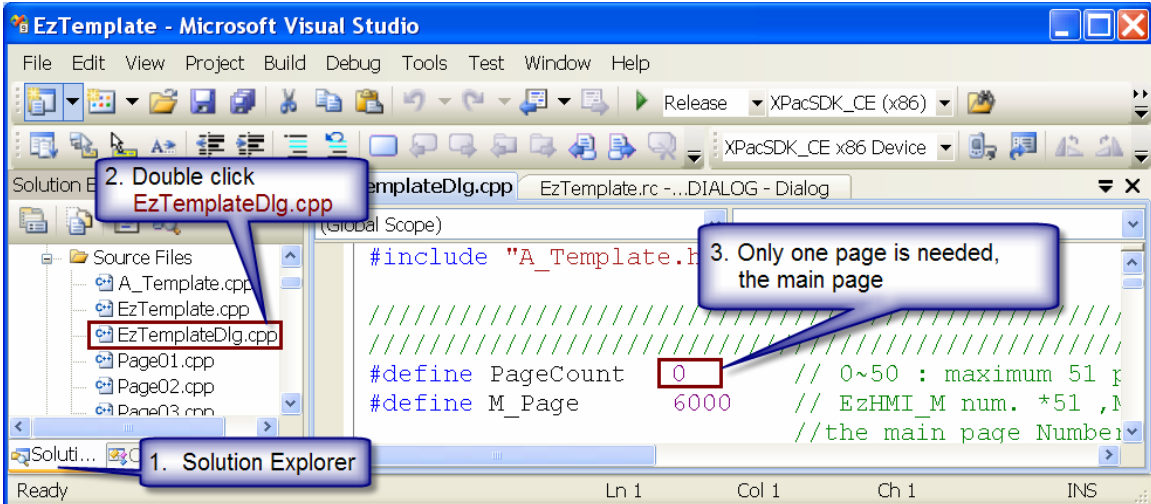
- Assign “Flash Timer 0,1,2...” the value 2.

STEP 18: Close the property page by clicking “OK”

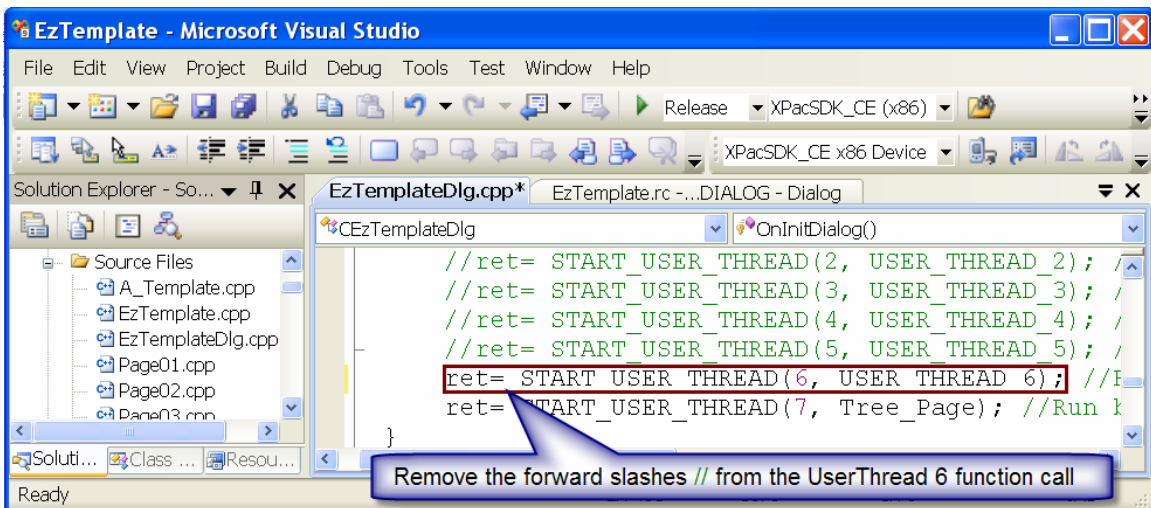


2.11.2 UserThread activation and code implementation

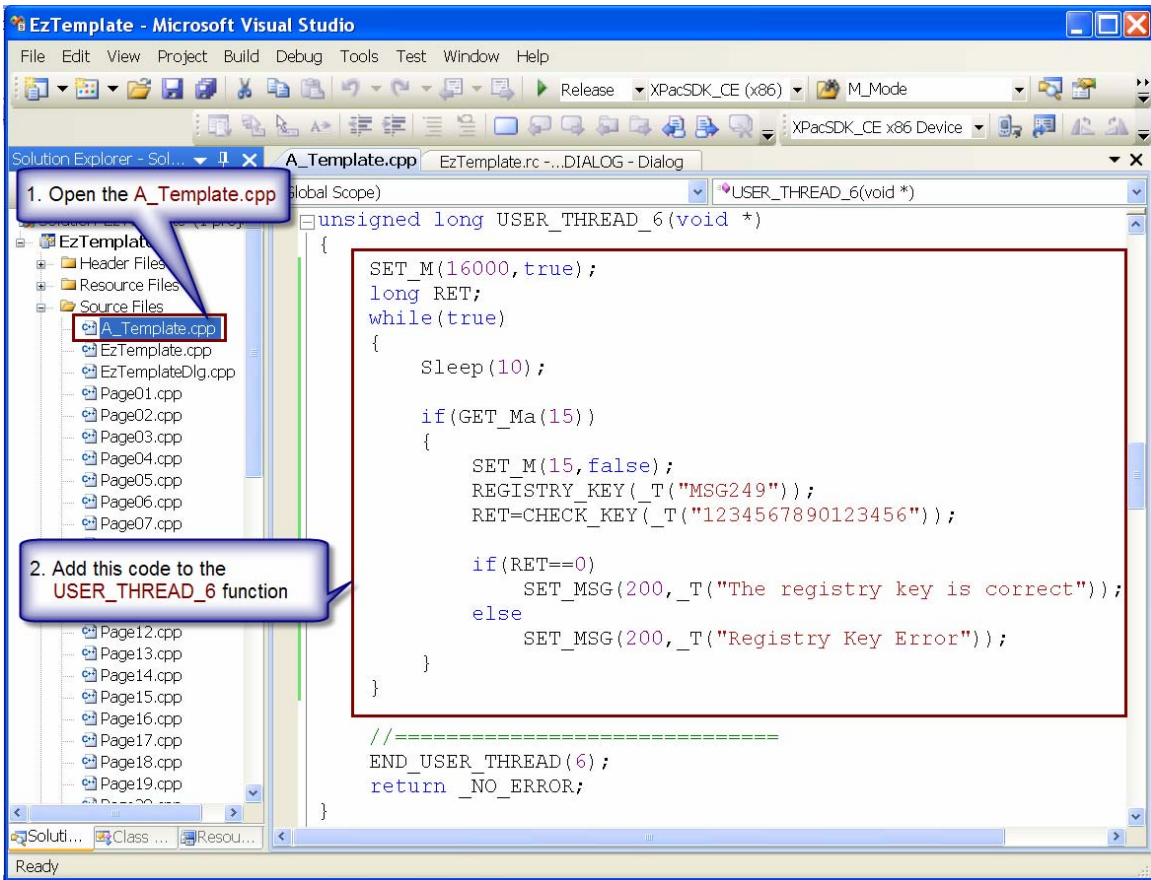
STEP 1: Only the main page is needed for this example therefore assign a zero to the PageCount definition.



STEP 2: The User_Thread_6 will be used to register the product. Remove the forward slashes from the USER_THREAD_6 function activation.



STEP 3: Add the following code to USER_THREAD_6 function



```

SET_M(16000,true);
long RET;

while(true)
{
    Sleep(10);

    if(GET_Ma(15))
    {
        SET_M(15,false);
        REGISTRY_KEY(_T("MSG249"));
        RET=CHECK_KEY(_T("1234567890123456"));
    }

    //=====
    END_USER_THREAD(6);
    return _NO_ERROR;
}

```

Activate the virtual keyboard by setting the M register number 16000 to true. When the ColorEdit object receives the focus the virtual keyboard automatically pops up.

Use an endless loop

Between each loop pause for 10 milliseconds

If the "Registry" ButtonST has been pressed and released its associated M register number 15 is set to true.

Reset the M register number 15 to false

This function uses the registration key entered in the ColorEdit box which is linked to the MSG register 249.

Uses the registration key to calculate the developer product key and compares this key with the key directly assigned to the function as a parameter. If both are identical the function returns 0.

If both keys are equal:
Assign the MSG register 200 the text "Valid registry key". The label object linked to this register will display this message.

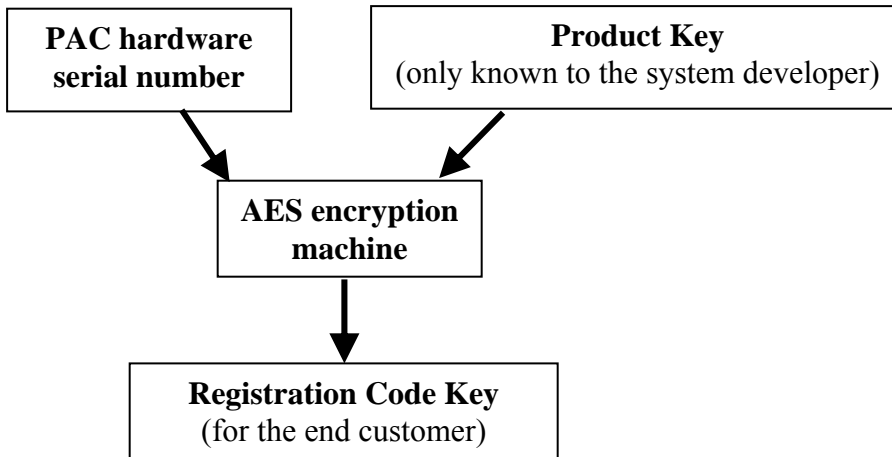
```
if(RET==0)
    SET_MSG(200,_T("Valid registry key"));
else
    SET_MSG(200,_T("Invalid registry key"));
}
```

If the keys do not match return an error message:
The error message "Invalid registry key" will be written to MSG register 200. The label object linked to this register will display this message.

2.11.3 Registry key generation

Before the example can be downloaded to the PAC the product key and the registration code key has to be specified by using the EzConfig utility.

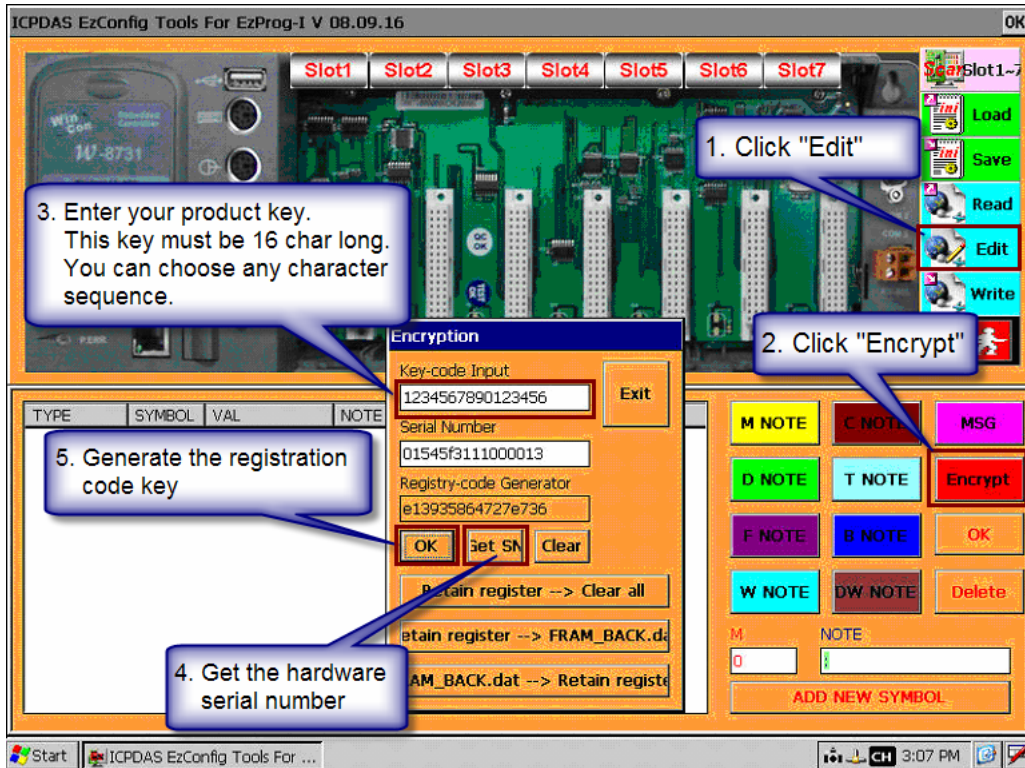
To generate a registration key for the end user the program developer has to specify a random 16 character product key. The encryption engine generates by using the hardware serial number of the PAC and the product key a registration key. The product key is only known to the system developer and should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



The **CHECK_KEY** API requires the programmer to enter real product key as a function parameter and then it generates from the registration code key entered by the end 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.

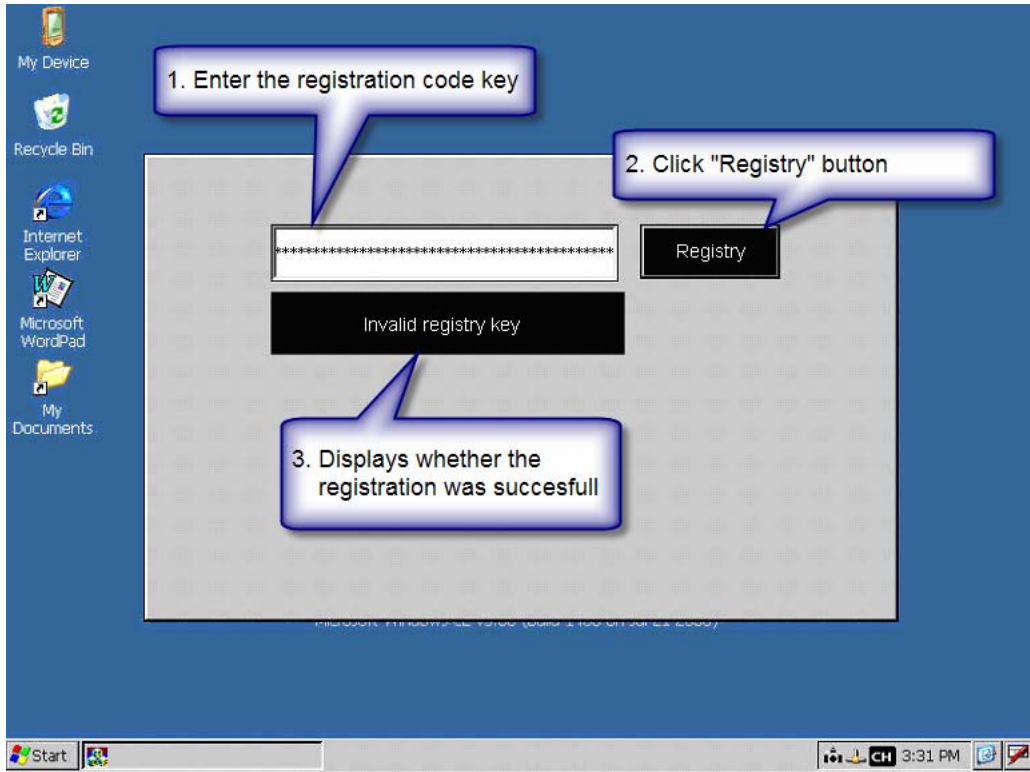
Steps describing the setting of the product key and registration code key:

- STEP 1: Click the “Edit” button
- STEP 2: Click the “Encrypt” button
- STEP 3: Enter your product key. This key must be 16 char 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”.
- STEP 6: Close the EzConfig utility.



Compile the project and download the execution file to the PAC as described in chapter 2.2.3. The following figure shows the user interface on the PAC.

Enter the registration code key in the ColorEdit box and click the “Registry” button. If the key has been entered incorrectly the message “Invalid registry key” appears in the label box.



2.12 Tutorial 12: Using Multilanguage message files

In the chapter 2.4 (Tutorial 4: Multi-language) the multi-language support of the EzHMI objects is being discussed. This chapter demonstrates the application of multi-language messages. EzProg-I supports at the maximum eight languages and provides for each language a separate text file. The messages used for the program have to be added together with a unique message identifier to the text files. Messages next the same identifier should present the same content in the corresponding language across the multi-language files. The text files themselves can be created by using Notepad.

2.12.1 EzHMI design and register settings

Open an EzTemplate project and rename

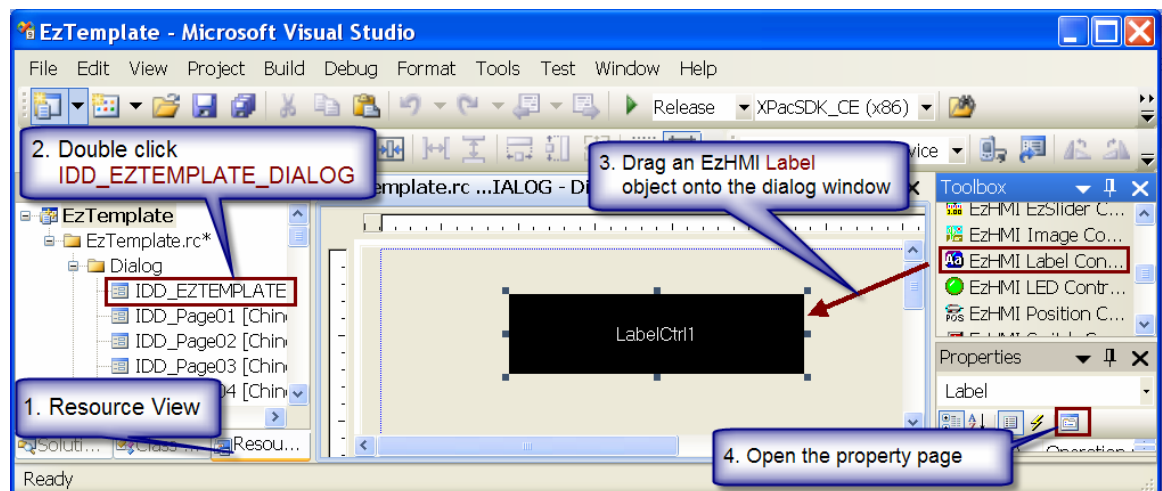
- the file “EzTemplate_800x600” to “**MSGF**” and
- the execution file “EzTemplate.exe” to “**MSGF.exe**”.

Chapter 2.2.1 describes this procedure in detail.

STEP 1: Open the `IDD_EZTEMPLATE_DIALOG` window.

STEP 2: Drag an EzHMI Label object onto the main dialog window.

STEP 3: Open the property pages: Right click the Label object and select “Properties”.
Click the “Property Pages” icon in the “Properties” window.



Do the following property settings:

STEP 4: Check the “**MSG/AI/AO/D/Fno Enable**”.

STEP 5: Link the Label object to the MSG (message) register number 201.

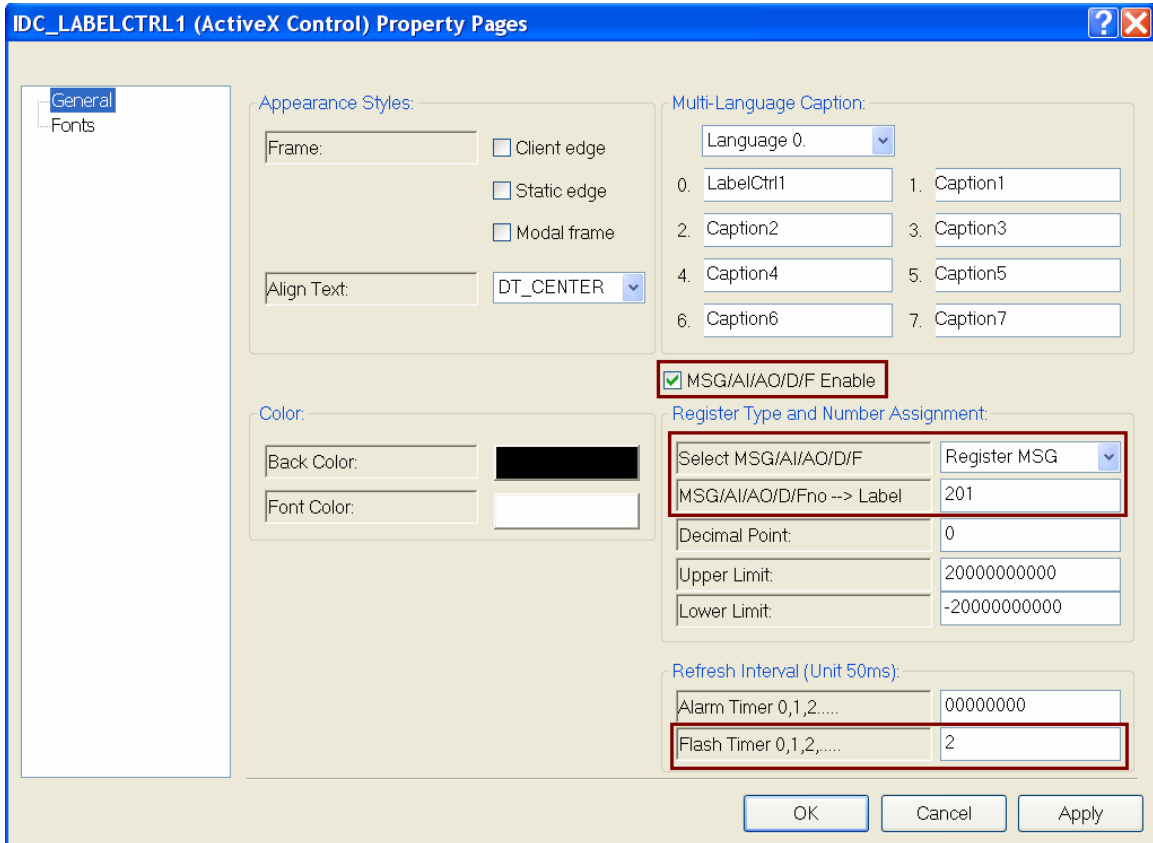
- Register type: Select “**Register MSG**” for “**Select Input MSG/AI/AO/D/F**”.

- Register number: Enter for “*ColorEditX → MSG/AI/AO/D/Fno*” the register number **201**.

STEP 6: Set the **Label** object refresh rate to 100 milliseconds.

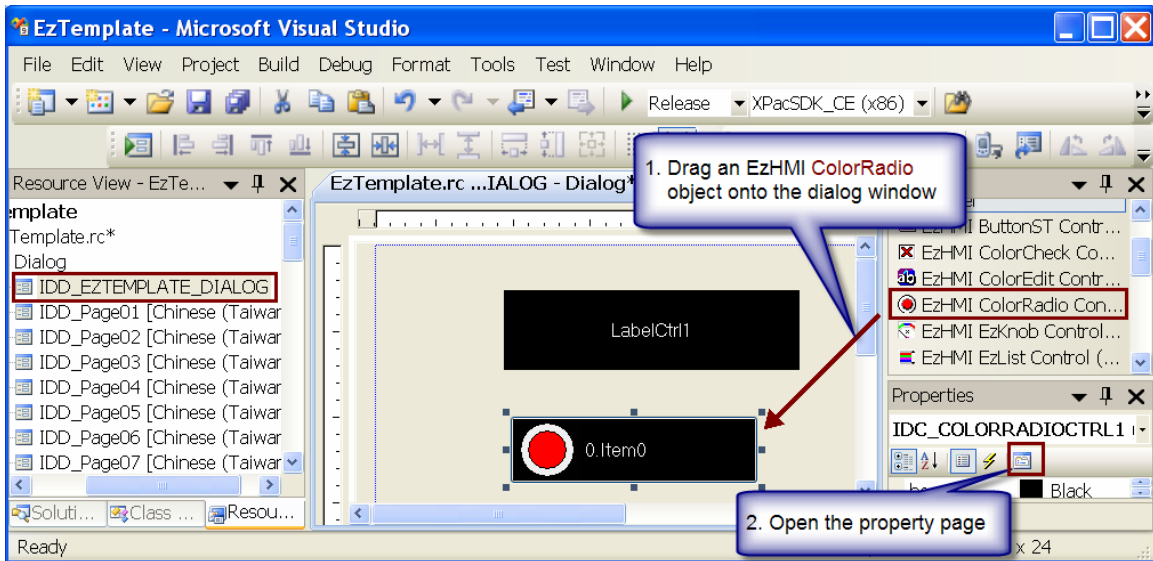
- Assign “*Flash Timer 0,1,2...*” the value **2**.

STEP 7: Close the property page by clicking “OK”.



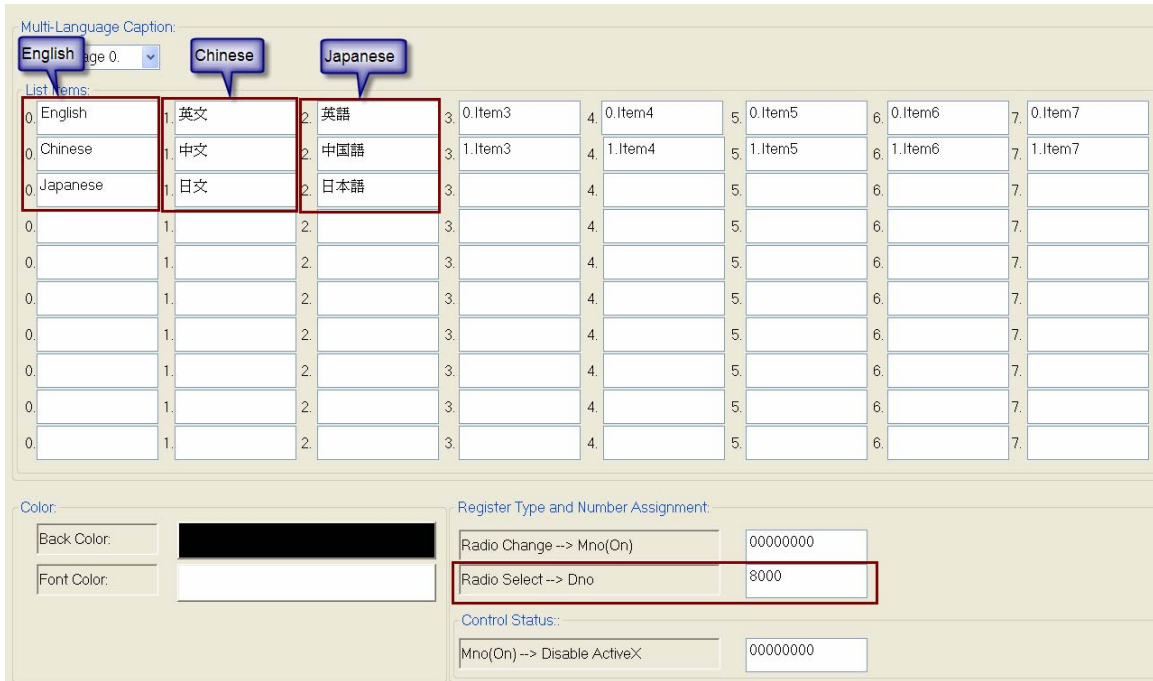
STEP 8: Drag an EzHMI ColorRadio object onto the main dialog window.

STEP 9: Open the property pages: Right click the ColorRadio object and select “Properties”. Click the “Property Pages” icon in the “Properties” window.



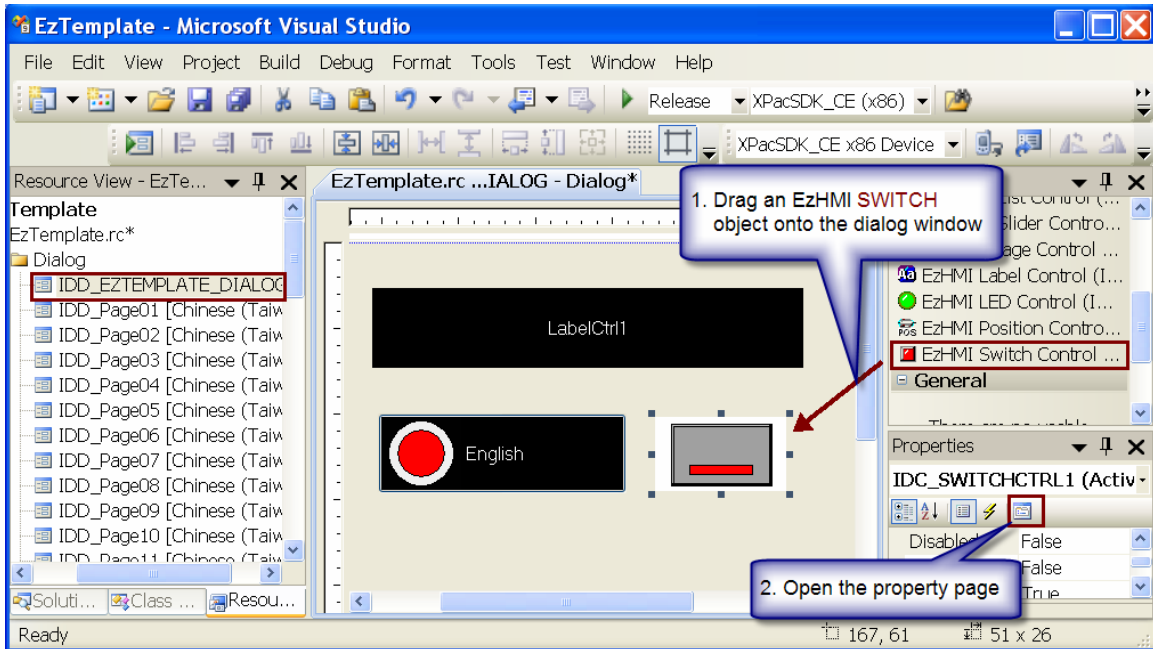
Do the following property settings:

- STEP 10: Provide the user with an option list of the three languages: English, Chinese and Japanese.
 The user can select one language for the user interface. If the interface language is set to English the three language options are displayed in English. If the user changes the language setting to Chinese the three language options are displayed in Chinese.
- STEP 11: Link the ColorRadio object to the D register number 8000.
 – Register number: Enter for “**Radio Select → Dno**” the register number **8000**.
 The D register 8000 is reserved for language setting. A maximum of eight languages are supported by the system. The first language in the list is represented by the value 0 in the D 8000 register and the eighth language by the value 7.
- STEP 12: Close the property page by clicking “OK”



STEP 13: Drag an EzHMI SWITCH object onto the main dialog window.

STEP 14: Open the property pages: Right click the SWITCH object and select “Properties”. Click the “Property Pages” icon in the “Properties” window.



Do the following property settings:

STEP 15: Select “SWITCH TOGGLE” as switch type

STEP 16: Check the “*Y/Mno Enable*”.

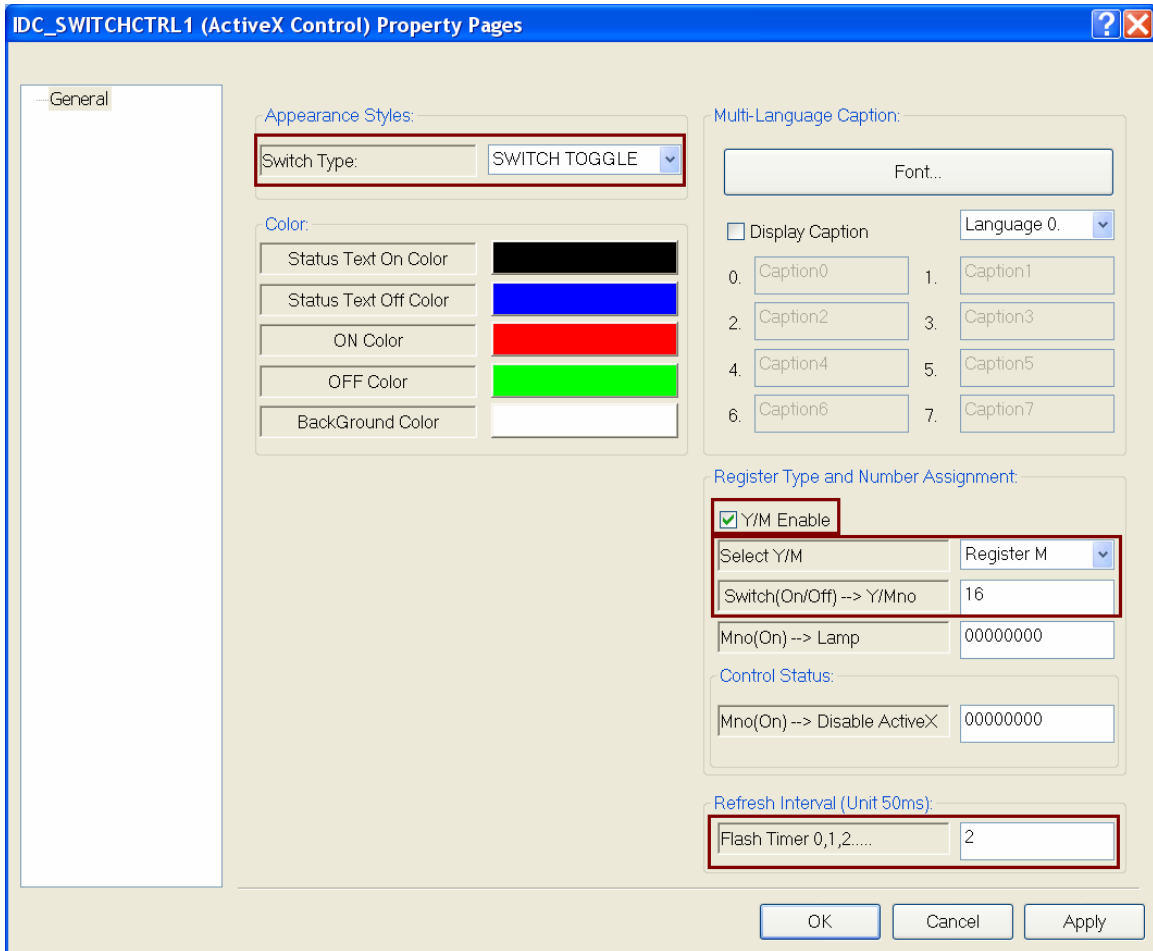
STEP 17: Link the SWITCH object to the M register number 16.

- Register type: Select “*Register M*” for “*Select Y/M*”.
- Register number: Enter for “*Switch(On/Off) → Y/Mno*” the register number *16*.

STEP 18: Set the *SWITCH* object refresh rate to 100 milliseconds.

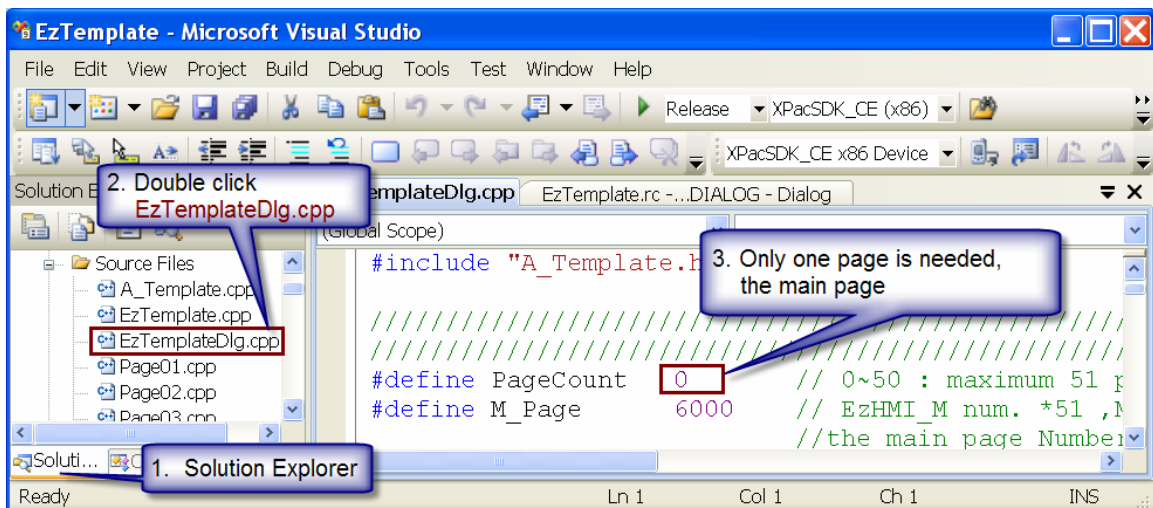
- Assign “*Flash Timer 0,1,2...*” the value *2*.

STEP 19: Close the property page by clicking “OK”.

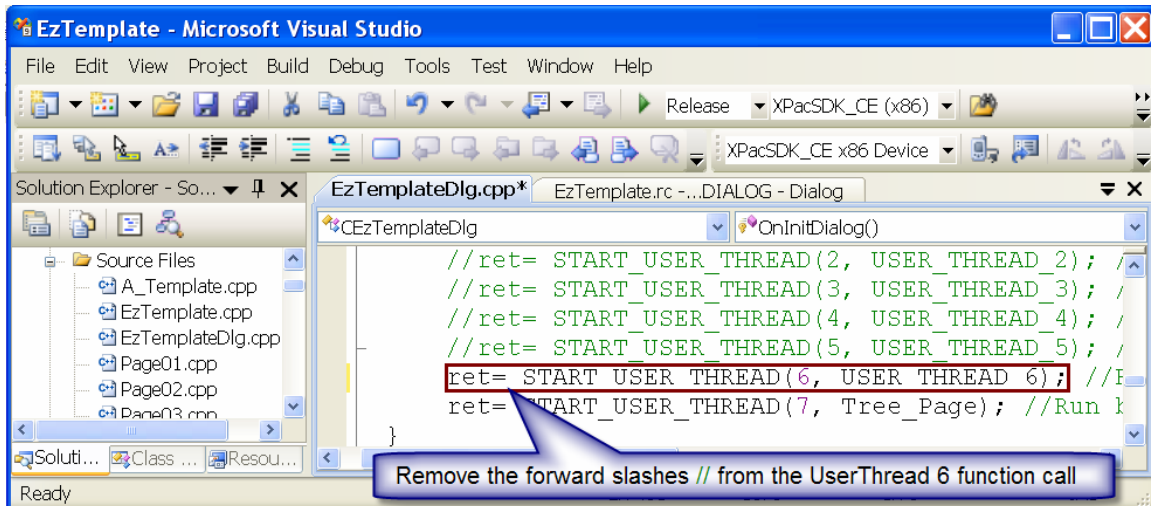


2.12.2 UserThread activation and code implementation

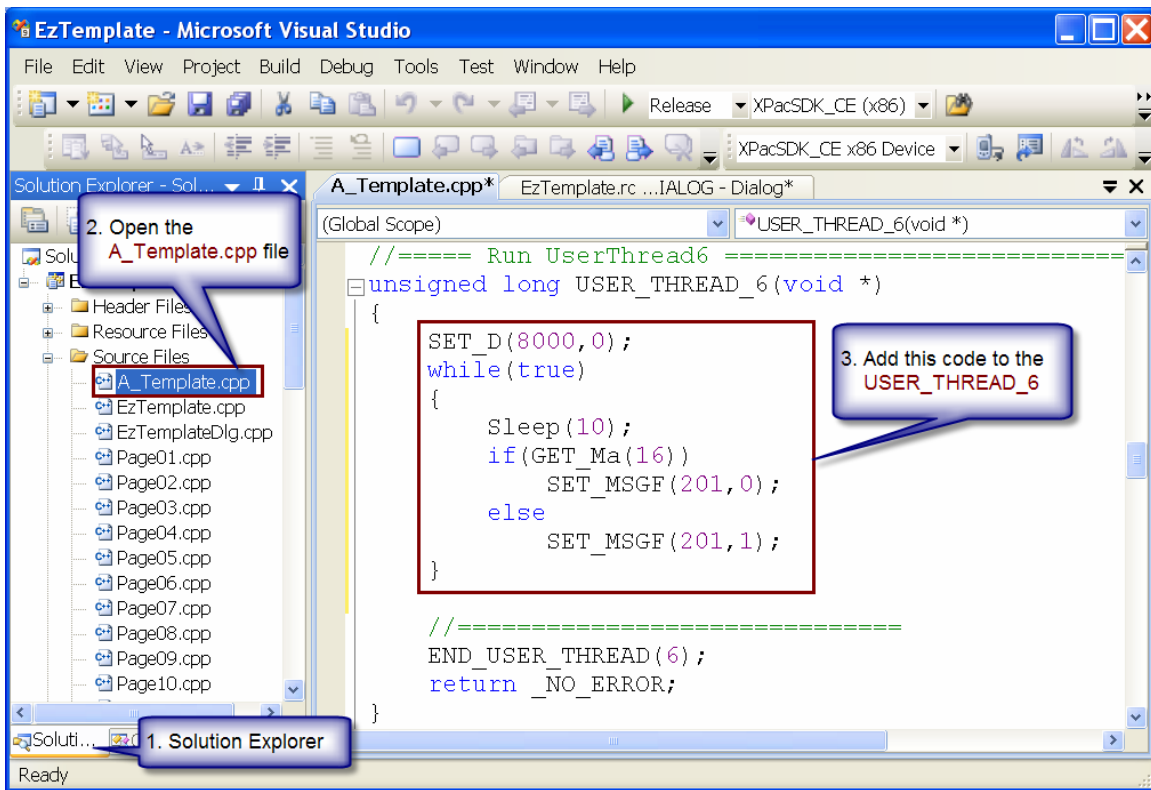
STEP 1: Only the main page is needed for this example therefore assign a zero to the PageCount definition.

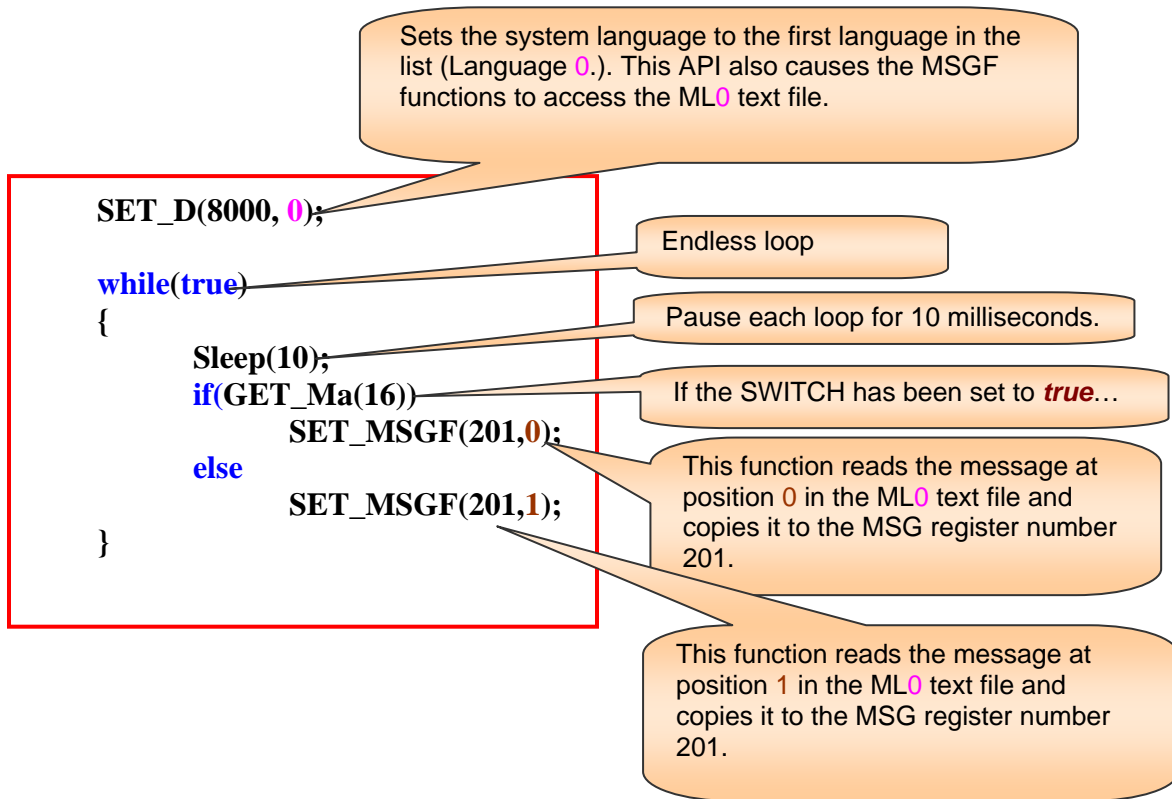


STEP 2: The User_Thread_6 will be used to handle the multi-language setting.
Remove the forward slashes from the USER_THREAD_6 function activation.



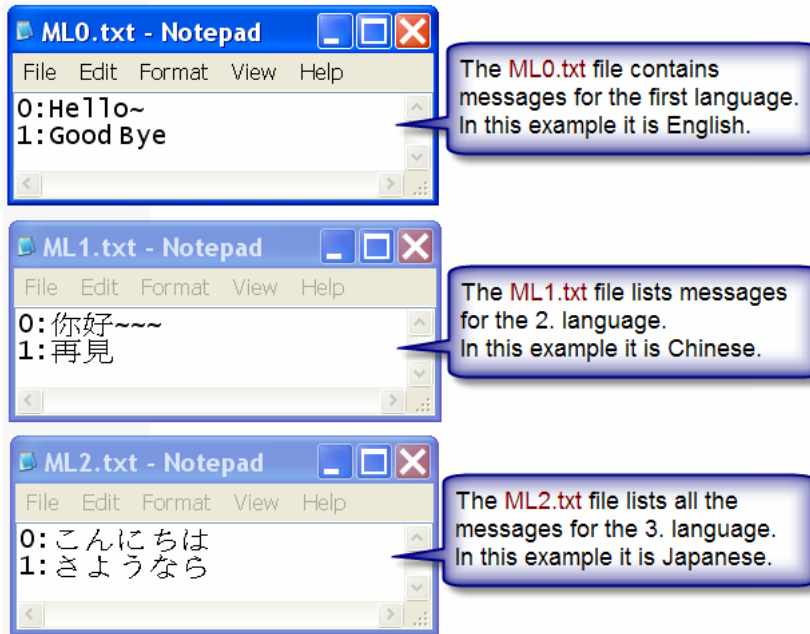
STEP 3: Add the following code to USER_THREAD_6 function



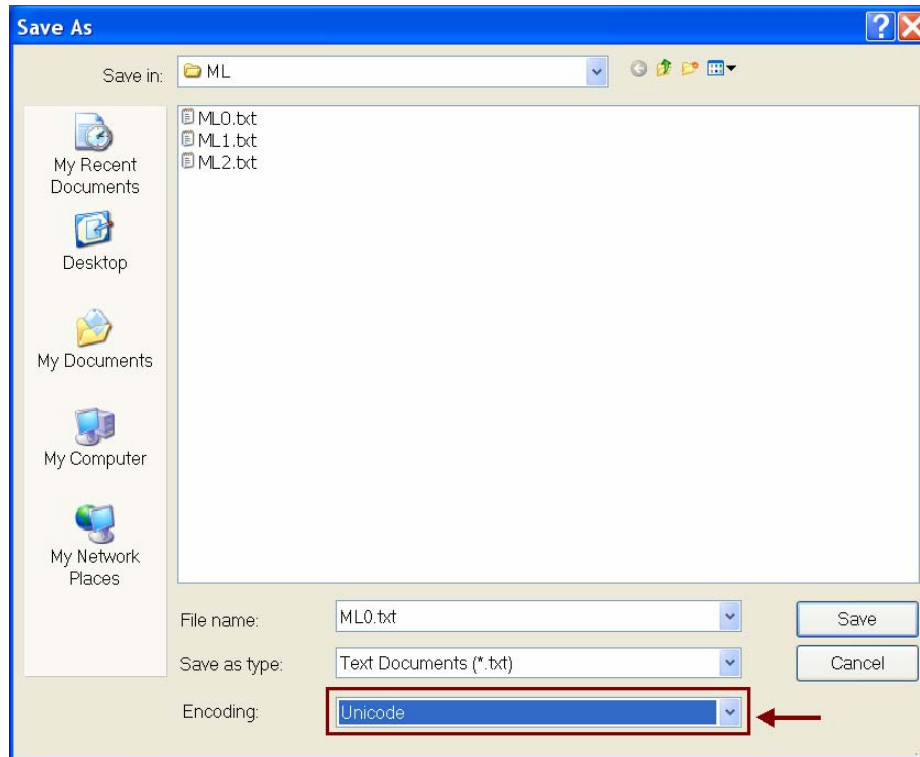


2.12.3 Create multi-language message files

STEP 1: Create for every language supported by your program a message text file on the PC. Identical messages in different languages should be assigned to the same message number. For example the text “Hello~” and the Chinese (“你好~~”) and Japanese (“こんにちは”) equivalent has been assigned to the same message number “0:”.



STEP 2: Make sure that these files are saved as Unicode text.

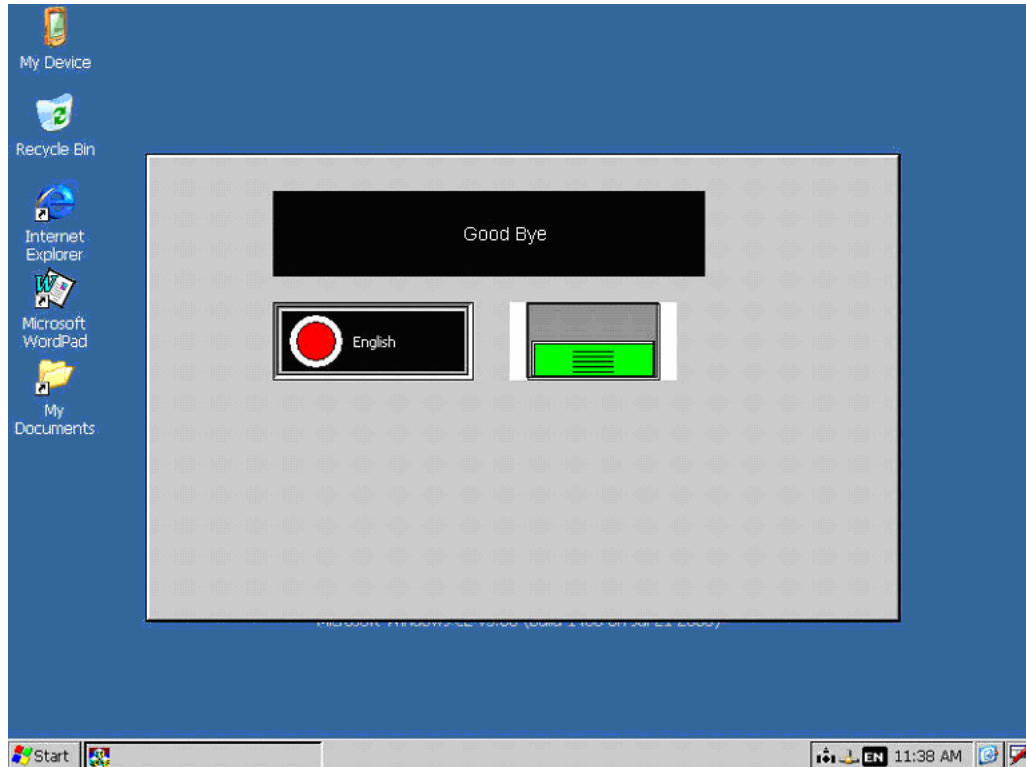


STEP 3: Download these text files to the following directory on the PAC:

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

Compile the project and download the execution file to the PAC as described in chapter 2.2.3. The following figure shows the user interface on the PAC.

When the SWITCH is set to ON the message “Hello~” and when switched off the message “Good Bye” appears in the Label box. A different language can be selected by clicking on the ColorRadio button. The message in the Label box appears now in the corresponding language.



2.13 Tutorial 13: Text fonts

The chapter demonstrates how to change the

- text color
- text type
- text size

of EzHMI objects.

If a font type is needed which is not provided by WinCE 6 or Windows XP, you first have to get hold of the font file and proceed as follows:





STEP 1: Copy the required font file (.ttf) to the “*C:\WINDOWS\Fonts*” directory of the PC (Windows XP)

STEP 2: Copy the same font file (.ttf) to the following directory of the PAC:

- for WinCon: “*\CompactFlash\ICPDAS\MAInit*”
- for MPac: “*\System_Disk\ICPDAS\MAInit*”

STEP 3: Reboot the PAC device

In this example the following fonts types are being used:

 wt034.ttf
 ps_24704(Bodoni_poster).ttf
 ps_24516(Apple_Chancery).ttf
 mvboli.ttf

These files are in the *C:\ICPDAS\EzProg-I\Tutorial\EzTemplate_FONT* directory.

2.13.1 EzHMI design and register settings

Open an EzTemplate project and rename

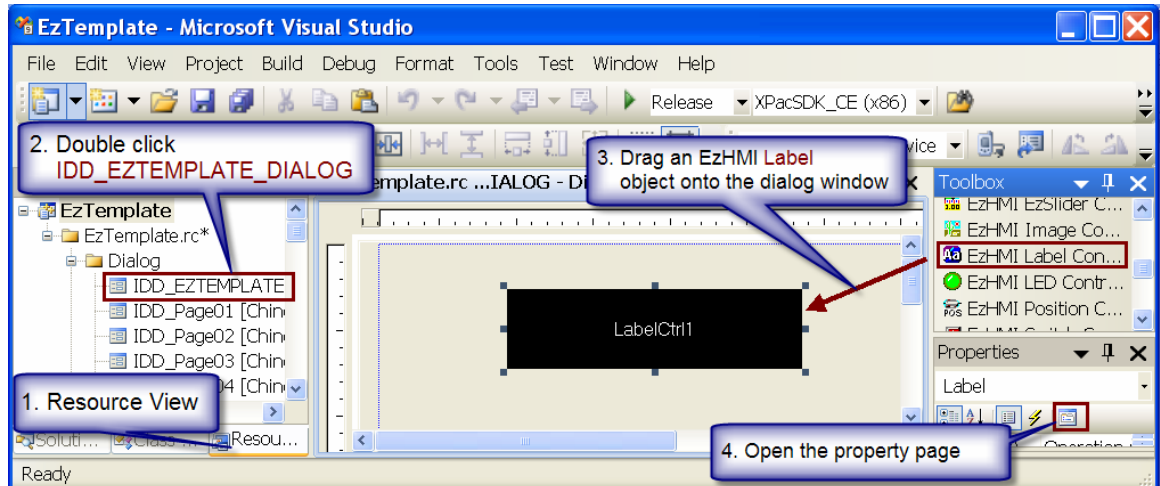
- the file “EzTemplate_800x600” to “*EzTemplate_FONT*” and
- the execution file “EzTemplate.exe” to “*EzTemplate_FONT.exe*”.

Chapter 2.2.1 describes this procedure in detail.

STEP 1: Open the IDD_EZTEMPLATE_DIALOG window.

STEP 2: Drag an EzHMI Label object onto the main dialog window.

STEP 3: Open the property pages: Right click the Label object and select “Properties”. Click the “Property Pages” icon in the “Properties” window.



Do the following property settings:

STEP 4: Change the text color (“Font Color”) to pink.

STEP 5: Check the “*MSG/AI/AO/D/F Enable*” checkbox.

STEP 6: Link the Label object to the MSG (message) register number 1.

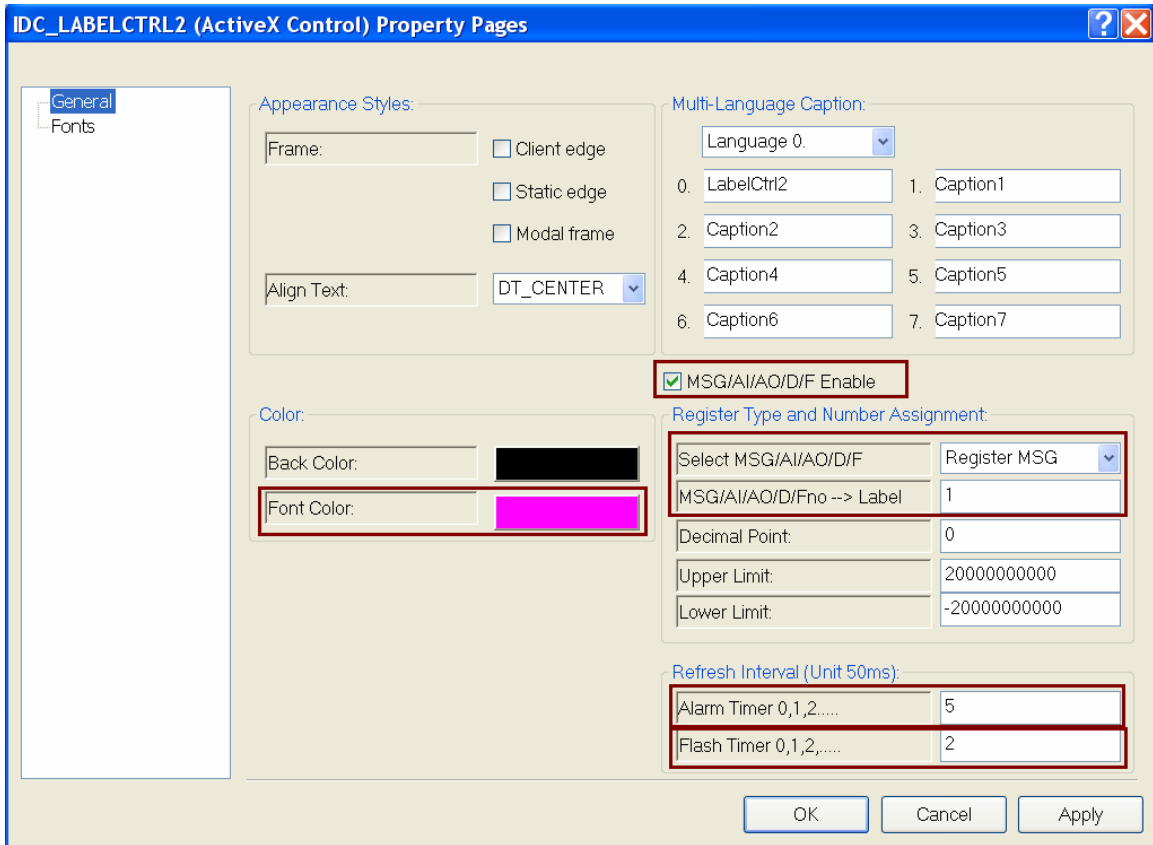
- Register type: Select “*Register MSG*” for “*Select Input MSG/AI/AO/D/F*”.
- Register number: Enter for “*ColorEditX → MSG/AI/AO/D/Fno*” the register number *1*.

STEP 7: Set the text flashing time to 250 milliseconds. A label linked to a MSG register will only flash if the message stored in the register is starts with a “#” character.

- Assign “*Alarm Timer 0,1,2...*” the value 5.

STEP 8: Set the *Label* object refresh rate to 100 milliseconds.

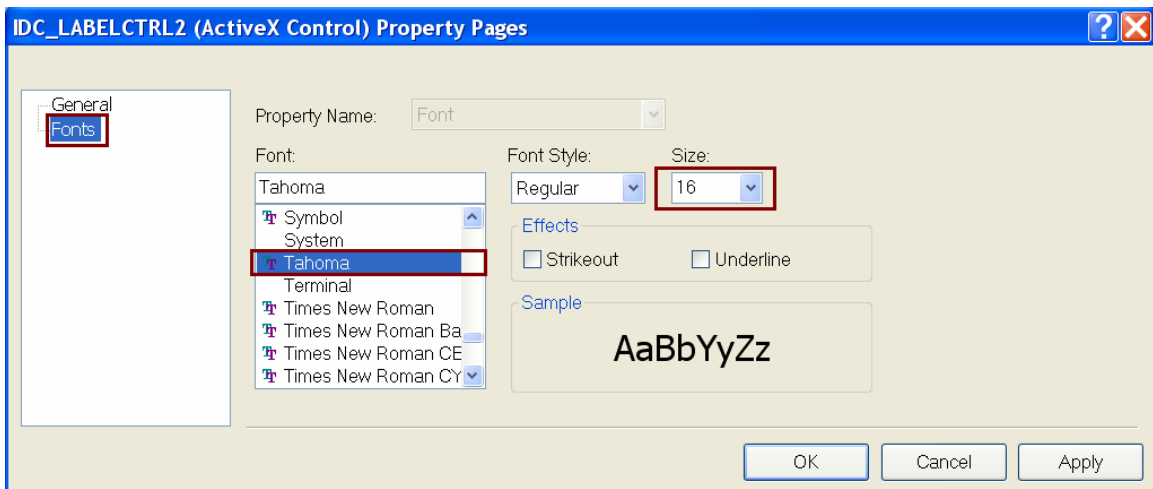
- Assign “*Flash Timer 0,1,2...*” the value 2.



STEP 9: Click the “Fonts” tab.

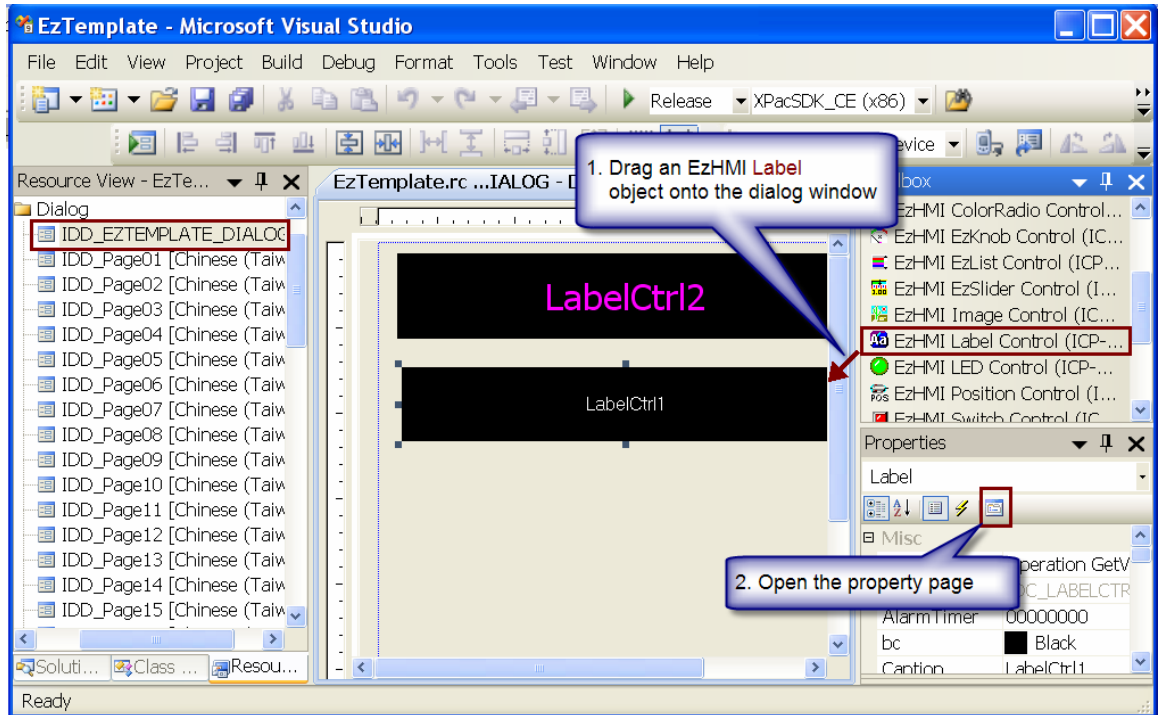
STEP 10: Select “Tahoma” as font style and set the text size to 16

STEP 11: Close the property page by clicking “OK”.



STEP 12: Drag another EzHMI Label object onto the main dialog window.

STEP 13: Open the property pages: Right click the Label object and select “Properties”.
Click the “Property Pages” icon in the “Properties” window.



Do the following property settings:

STEP 14: Align the text to the right (Align Text: DT_RIGHT)

STEP 15: Change the text color (“Font Color”) to lightgreen.

STEP 16: Enter “1234” in the first caption text box (0.).

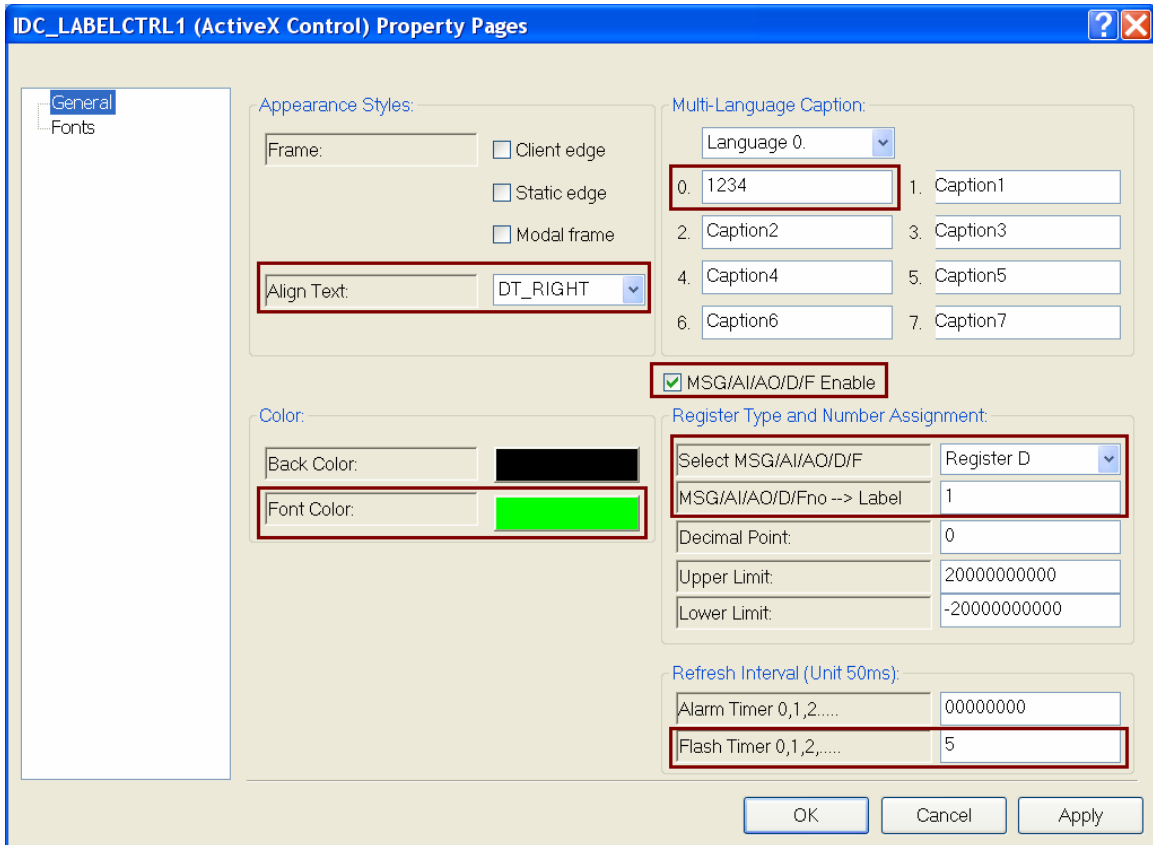
STEP 17: Check the “*MSG/AI/AO/D/F Enable*” checkbox.

STEP 18: Link the Label object to the MSG (message) register number 1.

- Register type: Select “*Register MSG*” for “*Select Input MSG/AI/AO/D/F*”.
- Register number: Enter for “*ColorEditX → MSG/AI/AO/D/Fno*” the register number *1*.

STEP 19: Set the *Label* object refresh rate to 250 milliseconds.

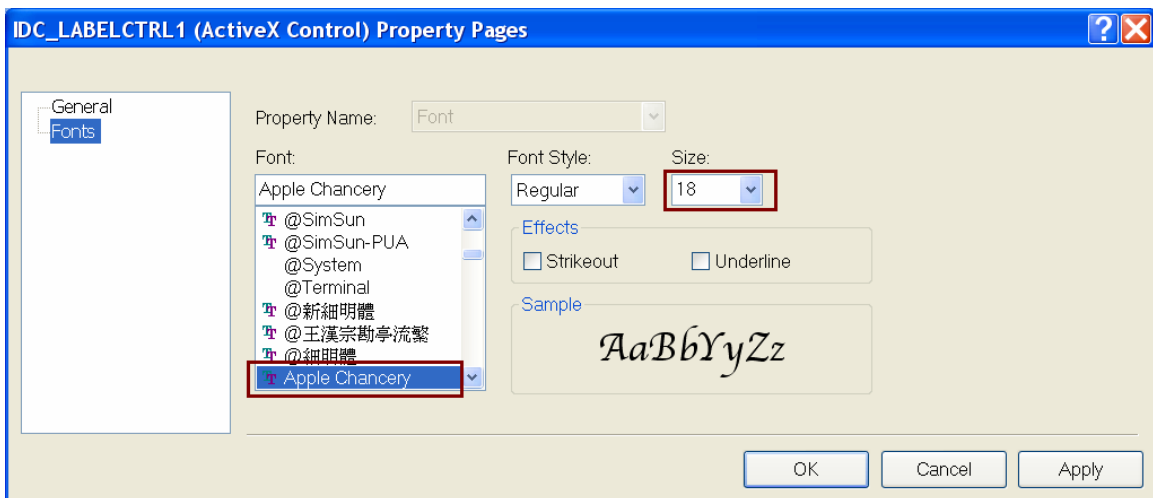
- Assign “*Flash Timer 0,1,2...*” the value *5*.

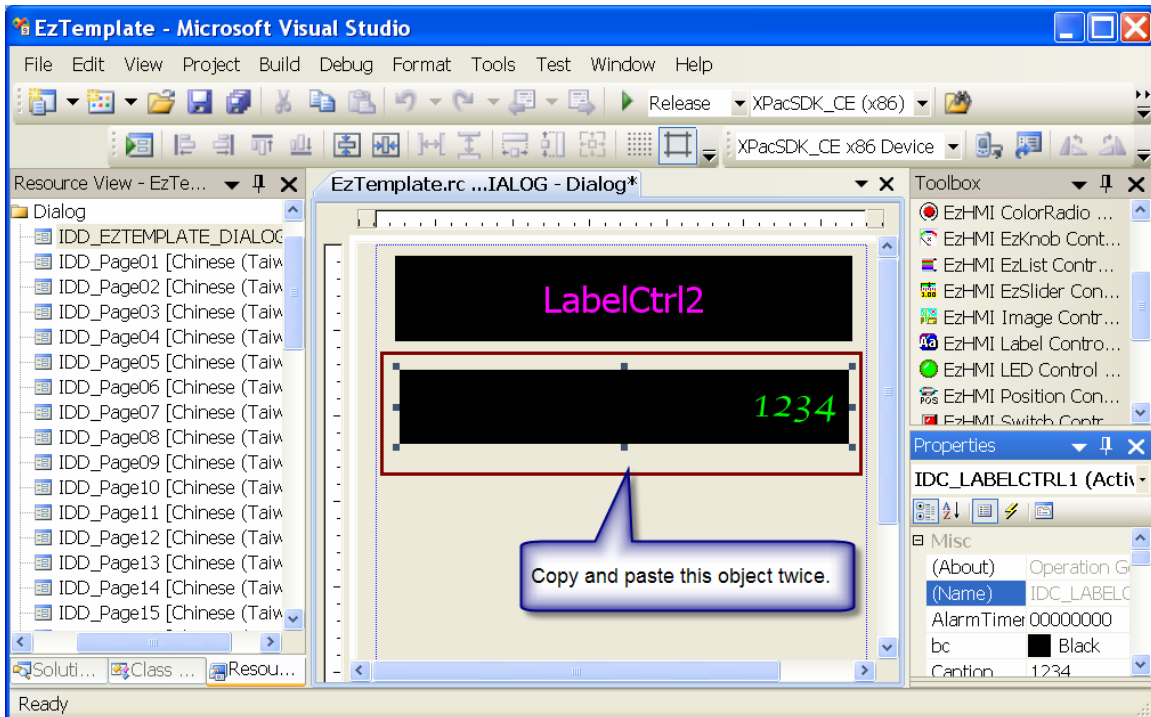


STEP 20: Open the “Fonts” property page.

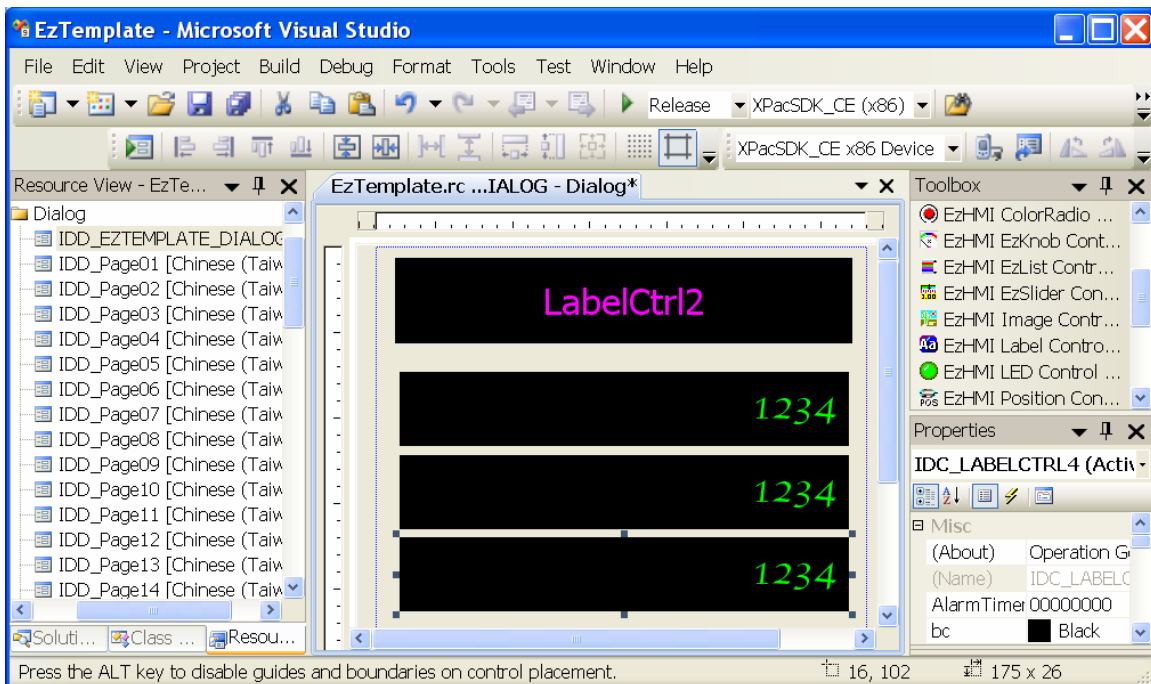
STEP 21: Select “*Apple Chancery*” as font style and set the text size to **18**.

STEP 22: Close the property page by clicking “OK”.

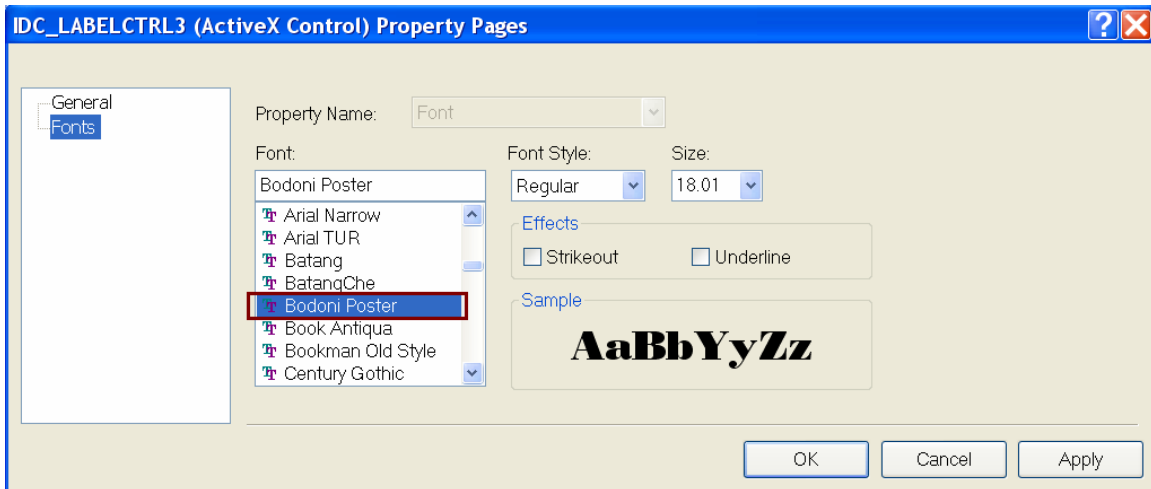




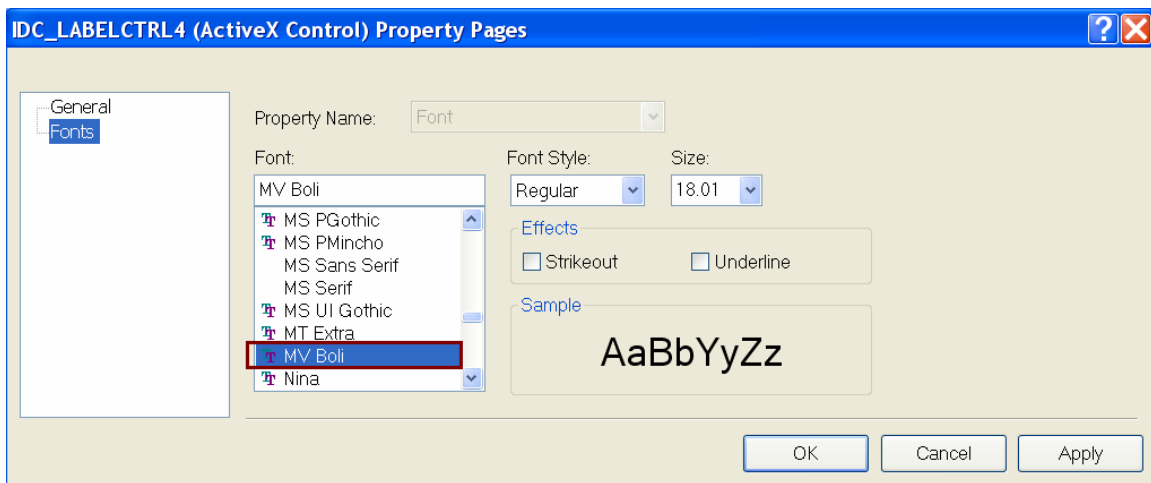
STEP 23: Copy the second label object and paste it twice and move the object to the position shown on the next figure:



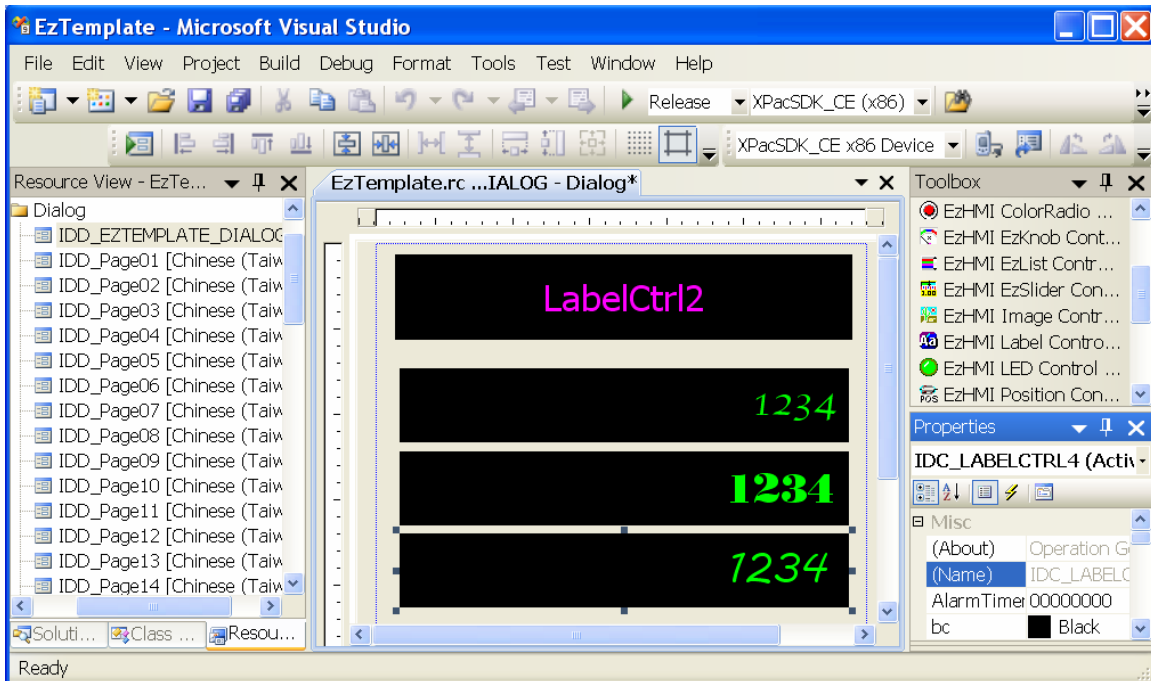
STEP 24: Open the “Fonts” property page of the first pasted object and select “*Bedoni Poster*” as font style. Click “OK” to close the window.



STEP 25: Open the “Fonts” property page of the second pasted object and select “*MV Boli*” as font style. Click “OK” to close the window.

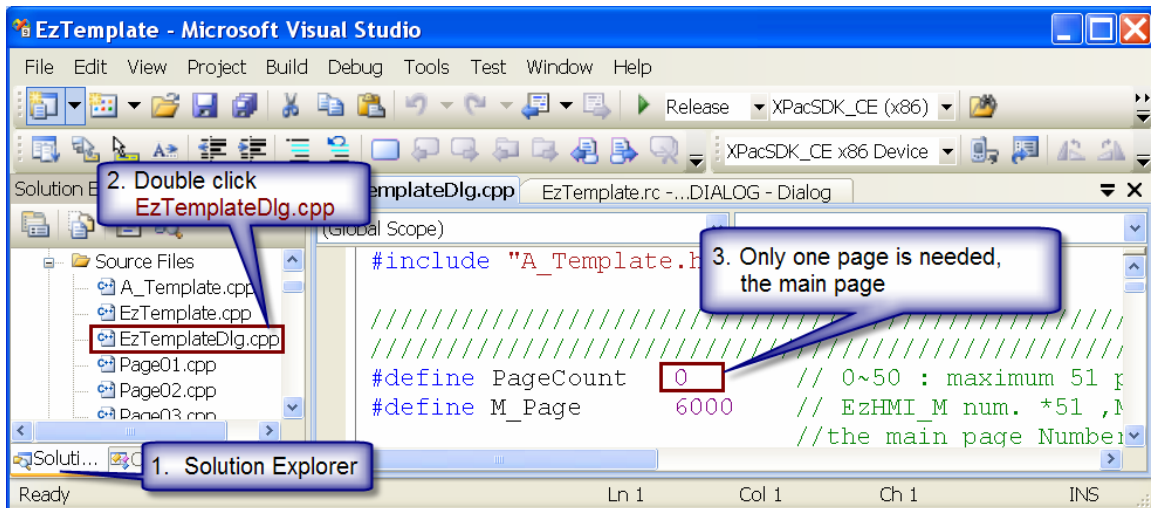


After the property setting has been done the label objects should have the following appearances:

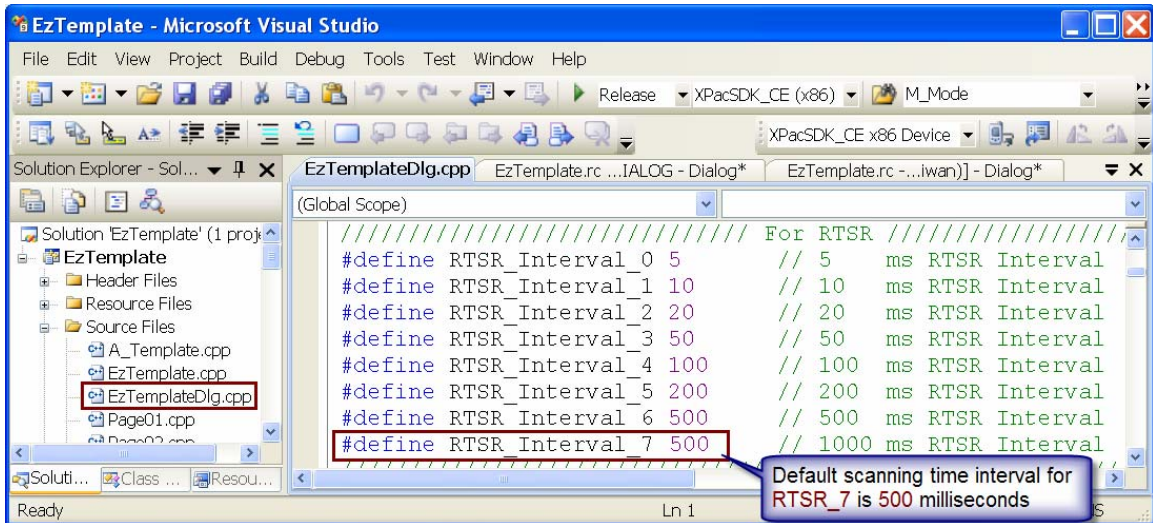


2.13.2 RTSR activation and code implementation

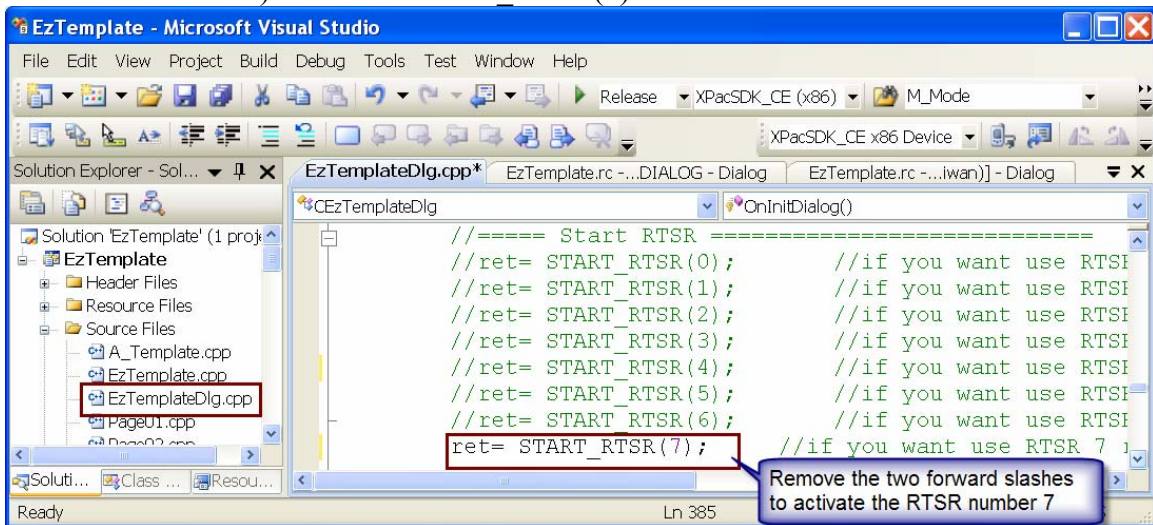
STEP 1: For this example only the main page is needed therefore assign a 0 to the PageCount definition.



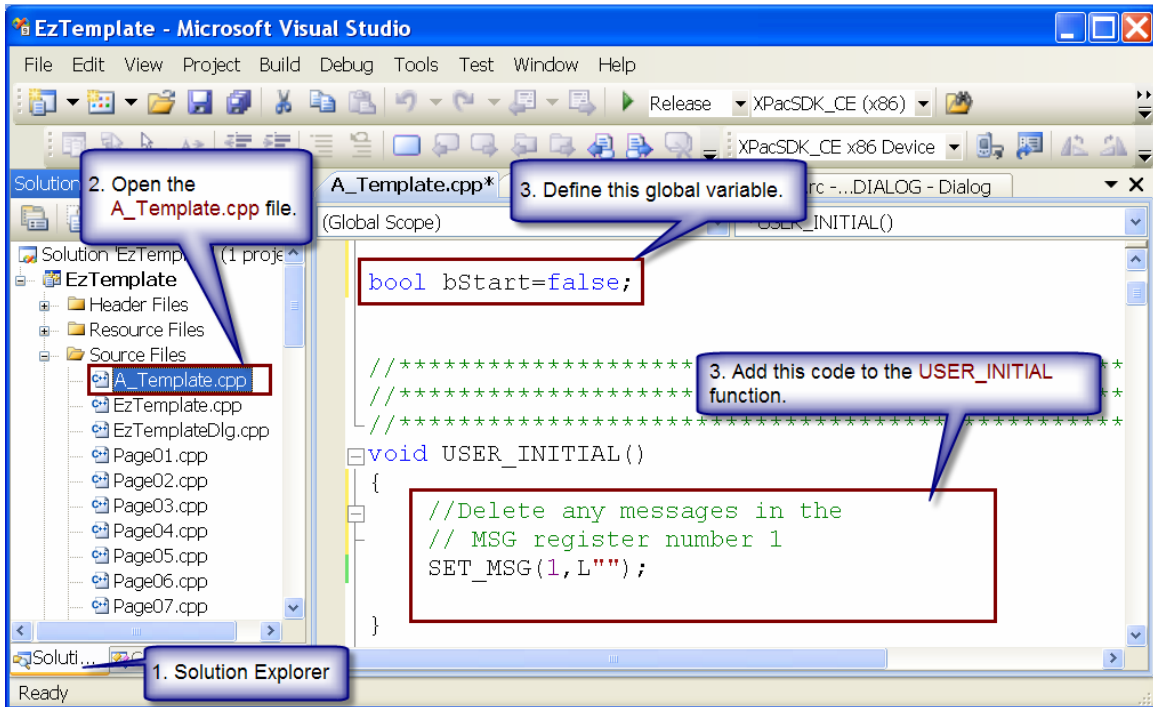
STEP 2: Set the RTSR scanning time interval. In this example the RSTR number 7 with the lowest priority level will be used. Change the scanning time interval to 500 milliseconds.



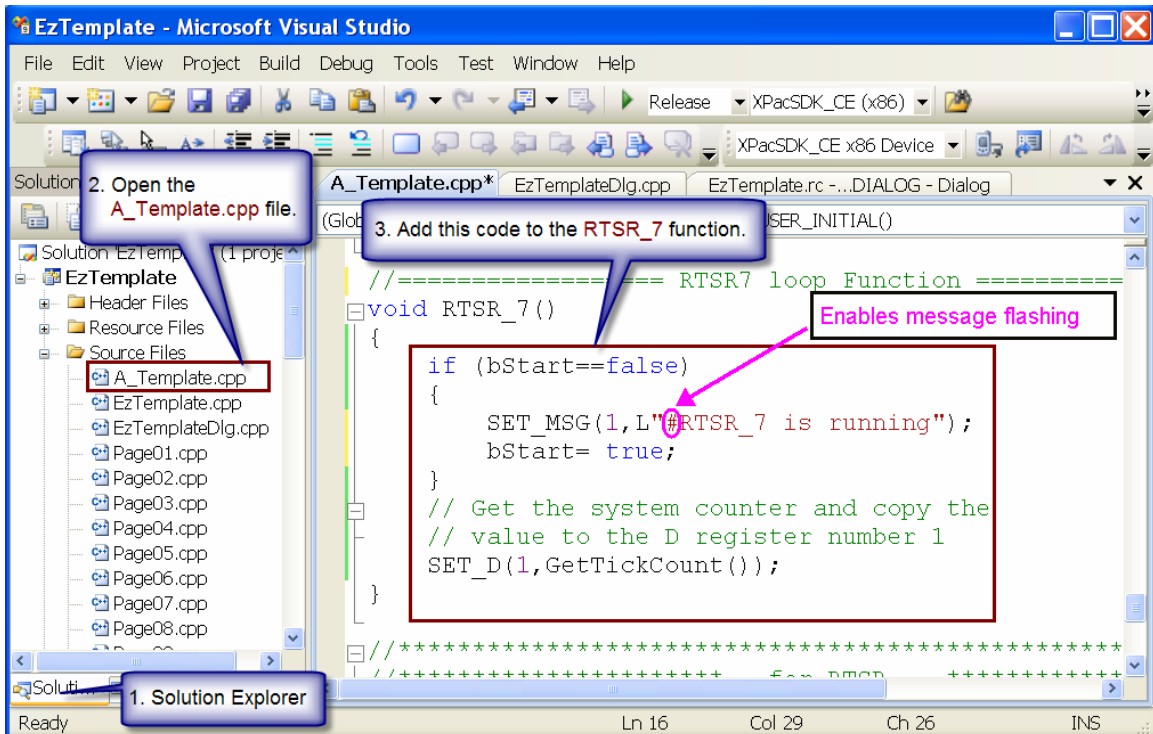
STEP 3: Activate the RTSR number 7 by removing the c++ comment (two forward slashes) from the START_RTZR(7) function call.



STEP 1: Define a global variable (`bStart=false`) for determining whether the RTSR loop has already been called. In the initialize function empty the MSG register number 1.

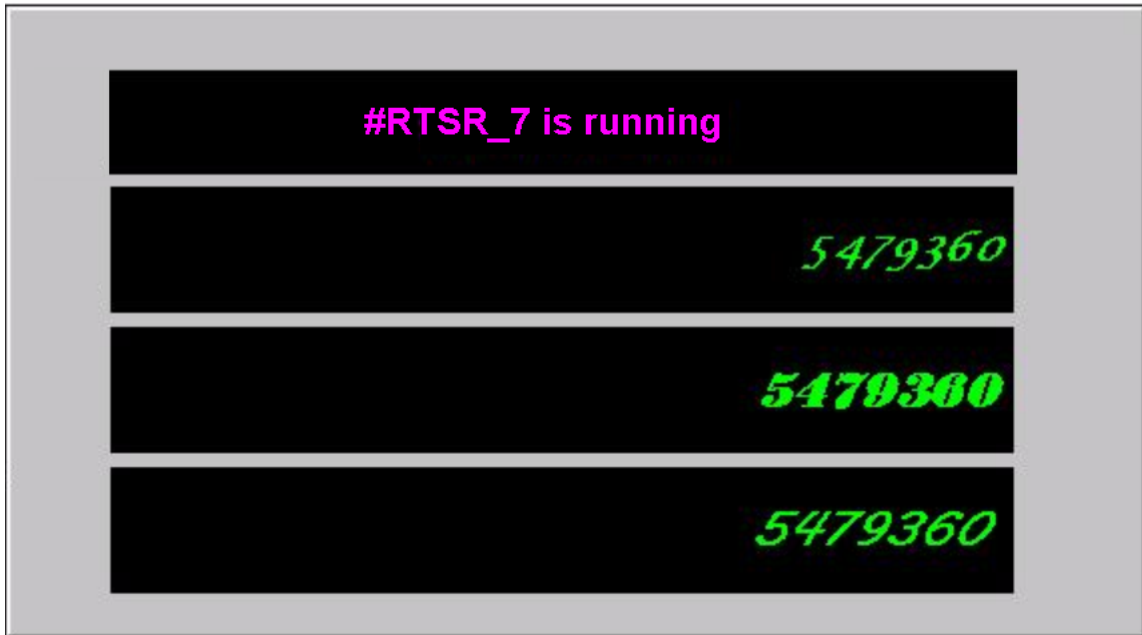


STEP 2: Add the following code to **RTSR_7** function. When the **RTSR** is being called the first time the “if” block writes the message “**#RTSR_7 is running**” to the **MSG** register number 1. The **GetTickCount** function returns the number of milliseconds that have elapsed since the system was started.



Compile the project and download the execution file to the PAC as described in chapter 2.2.3. The following figure shows the user interface on the PAC.

The first label displays the text “#RTSR_7 is running” and is flashing every 250 milliseconds and the other three labels are updated with the new count tick value at the RTSR scan rate of 500 milliseconds.



2.14 Tutorial 14: Steps

2.14.1 EzHMI design and register settings

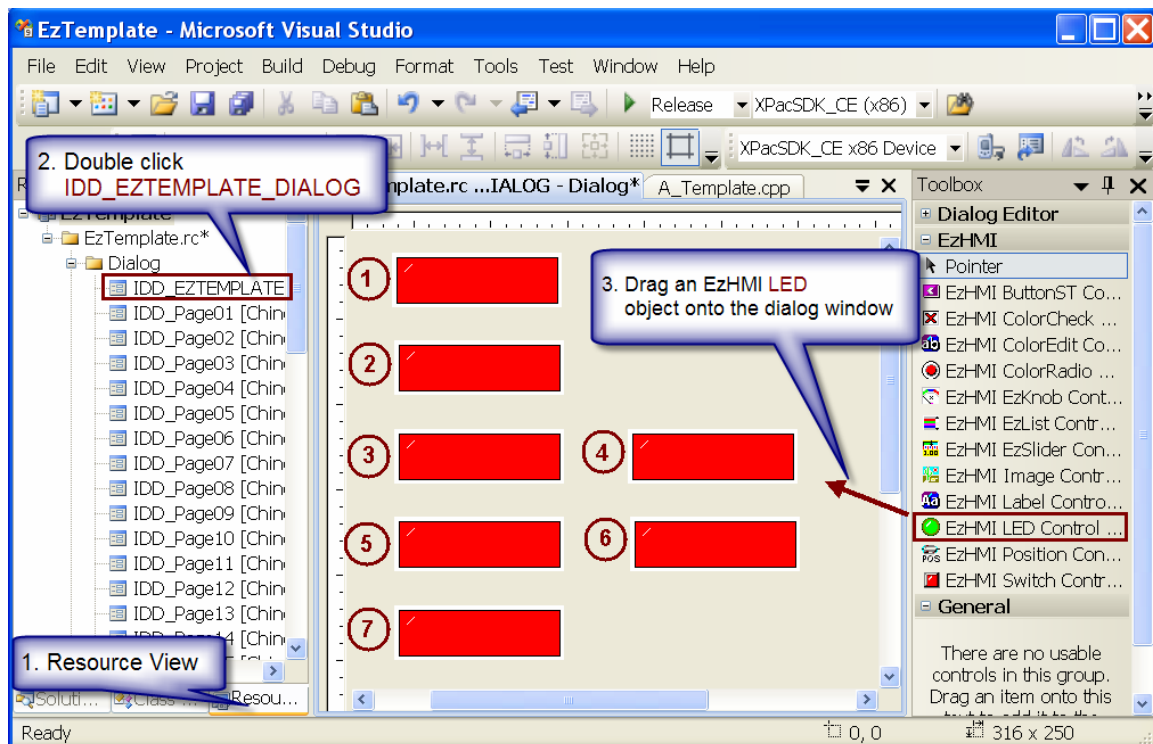
Open an EzTemplate project and rename

- the file “EzTemplate_800x600” to “*Step*” and
- the execution file “EzTemplate.exe” to “*Step.exe*”.

Chapter 2.2.1 describes this procedure in detail.

STEP 1: Open the IDD_EZTEMPLATE_DIALOG window.

STEP 2: Drag seven EzHMI LED objects (1 to 7) onto the main dialog window as shown on the next figure.



STEP 3: Open the property pages: Right click the LED object and select “Properties”. Click the “Property Pages” icon in the “Properties” window.

For each LED object do the following property settings:

STEP 4: Enable the “*Display Caption*” checkbox.

STEP 5: Enter for each LED the following caption name in the first text box (number 0.):

- LED 1 : M20
- LED 2 : S100
- LED 3 : M21(S200)
- LED 4 : M22(S200)
- LED 5 : S101
- LED 6 : S102
- LED 7 : S300

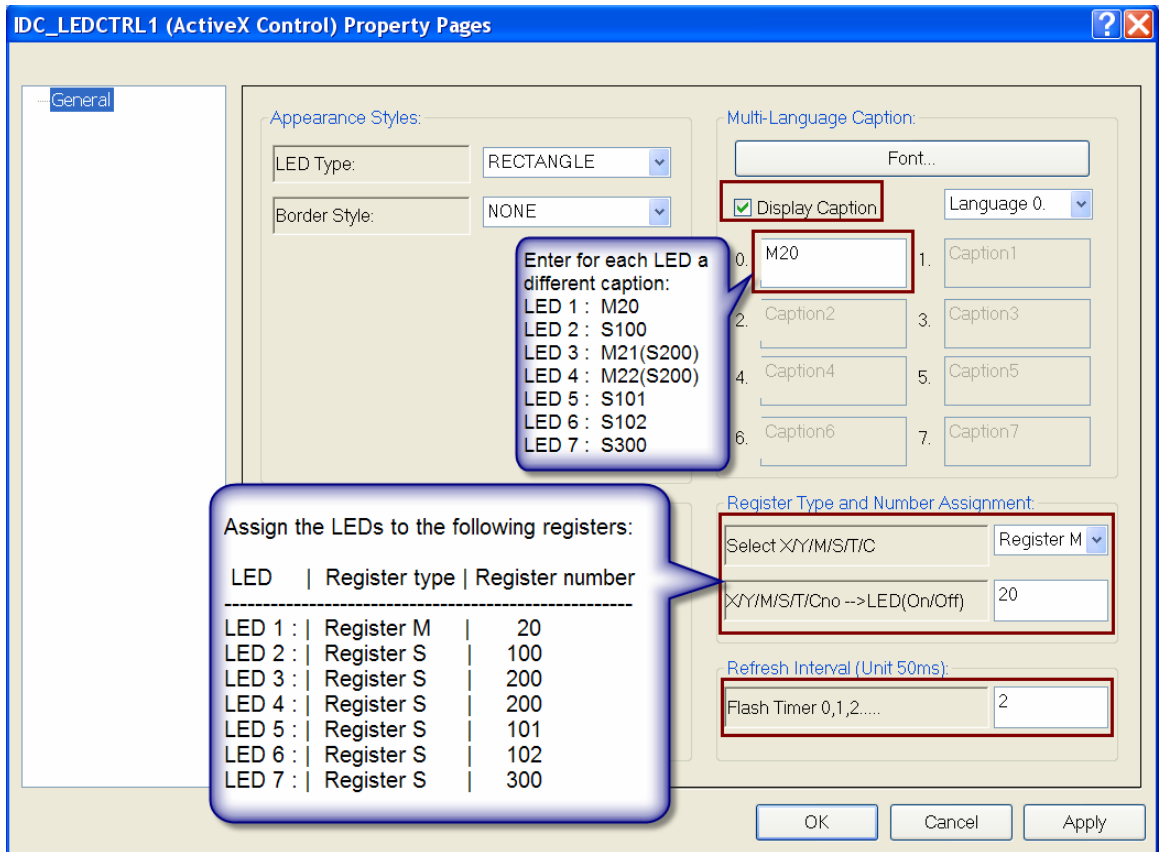
STEP 6: Link each LED object to the following register type and number:

LED	Register type	Register number
LED 1 :	Register M	20
LED 2 :	Register S	100
LED 3 :	Register S	200
LED 4 :	Register S	200
LED 5 :	Register S	101
LED 6 :	Register S	102
LED 7 :	Register S	300

STEP 7: Set the **LED** objects refresh rate to 100 milliseconds.

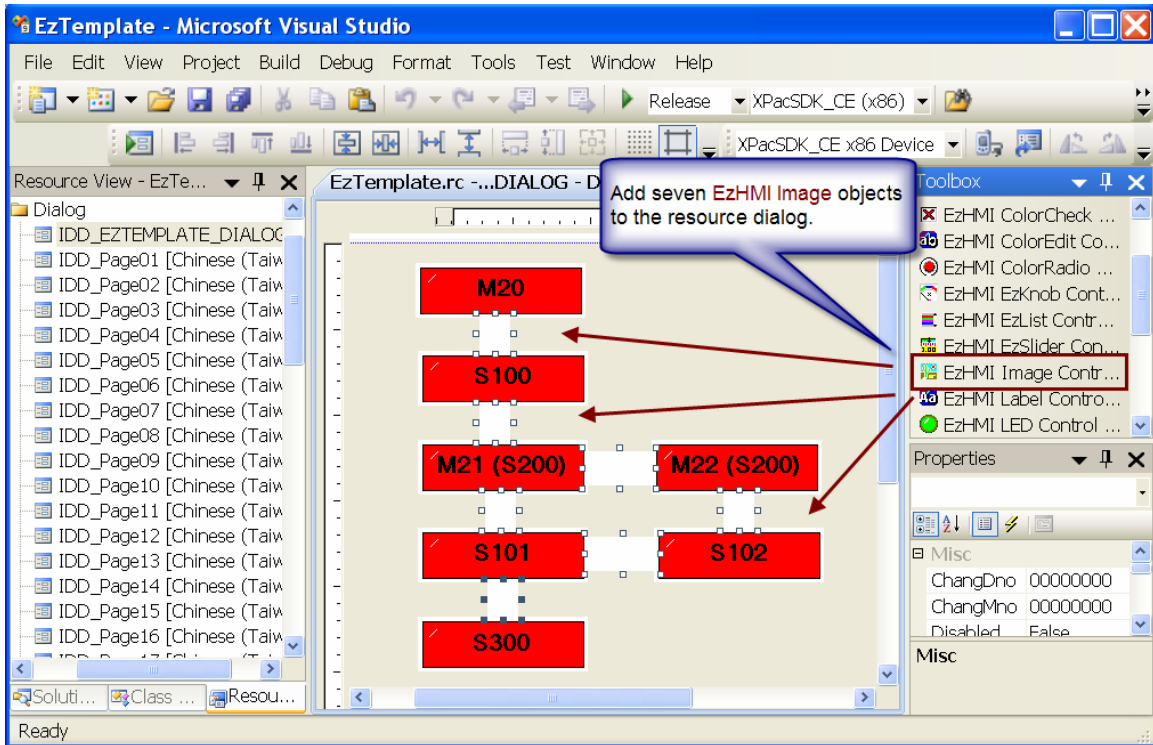
- Assign "**Flash Timer 0,1,2...**" the value **2**.

STEP 8: Close each property page by clicking "OK".



Connect the LED objects with a line:

STEP 9: Drag seven EzHMI Image objects (L1 to L7) onto the main dialog window as shown on the next figure.



STEP 10: Open the property pages: Right click the Image object and select “Properties”. Click the “Property Pages” icon in the “Properties” window.

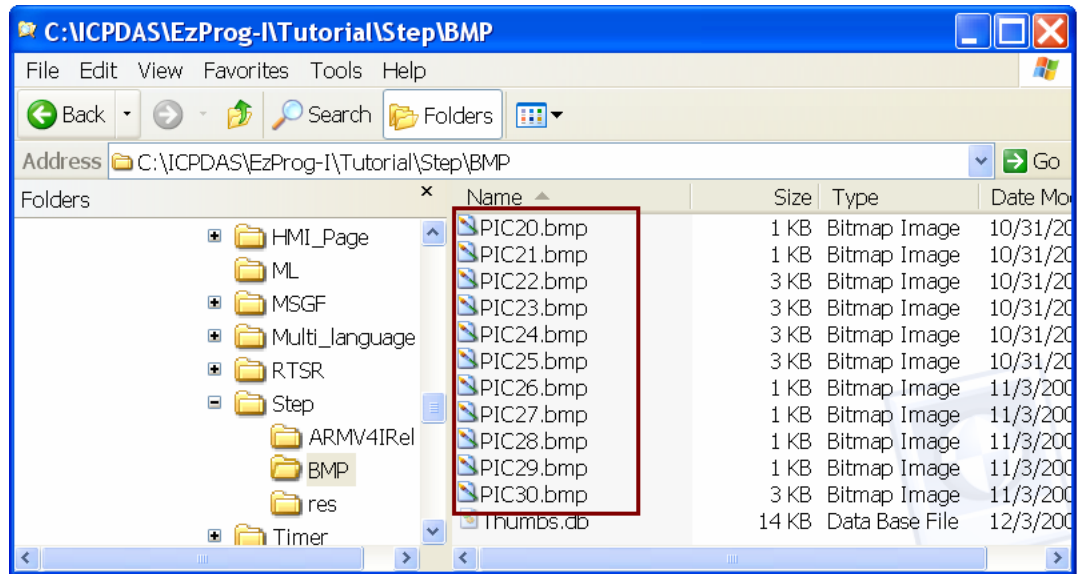
For each Image object do the following property settings:

STEP 11: Load the following bitmaps to the Image objects:

Image	Bitmap
L1:	C:\ICPDAS\EzProg-I\Tutorial\Step\BMP\PIC20.bmp
L2:	C:\ICPDAS\EzProg-I\Tutorial\Step\BMP\PIC20.bmp
L3:	C:\ICPDAS\EzProg-I\Tutorial\Step\BMP\PIC22.bmp
L4:	C:\ICPDAS\EzProg-I\Tutorial\Step\BMP\PIC20.bmp
L5:	C:\ICPDAS\EzProg-I\Tutorial\Step\BMP\PIC20.bmp
L6:	C:\ICPDAS\EzProg-I\Tutorial\Step\BMP\PIC20.bmp
L7:	C:\ICPDAS\EzProg-I\Tutorial\Step\BMP\PIC24.bmp

STEP 12: Download all the bitmaps in the *C:\ICPDAS\EzProg-I\Tutorial\Step\BMP* directory to the following directory of the PAC:

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



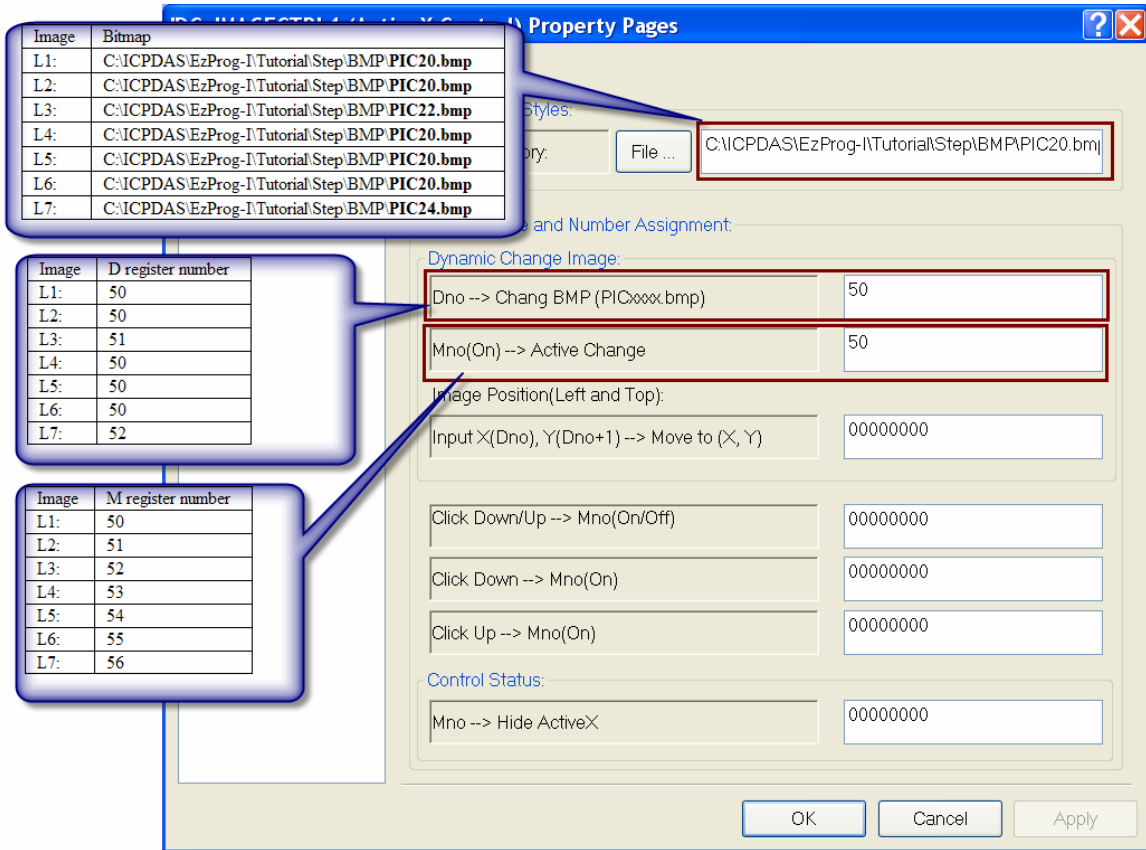
STEP 13: Enter for each Image the following register number for “*Dno* → *Change BMP(PICxxx.bmp)*”.

Image	D register number
L1:	50
L2:	50
L3:	51
L4:	50
L5:	50
L6:	50
L7:	52

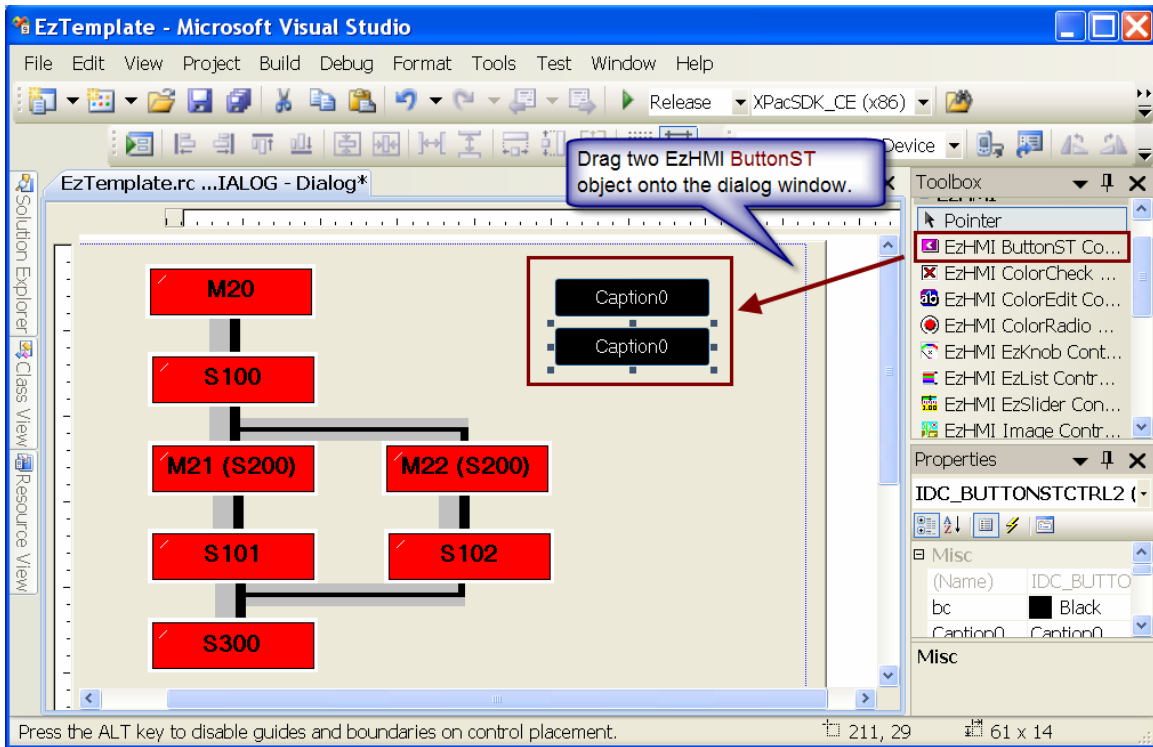
STEP 14: Assign “*Mno(On)* → *Active Change*” a M register number. If this register changes its value from false to true the Image object loads and displays the bitmap indicated by the D register value. The Image control sets the M flag automatically back to false after updating the image.

Image	M register number
L1:	50
L2:	51
L3:	52
L4:	53
L5:	54
L6:	55
L7:	56

STEP 15: Close each property page by clicking “OK”.



STEP 16: Drag two EzHMI ButtonST objects onto the main dialog window as shown on the next figure.

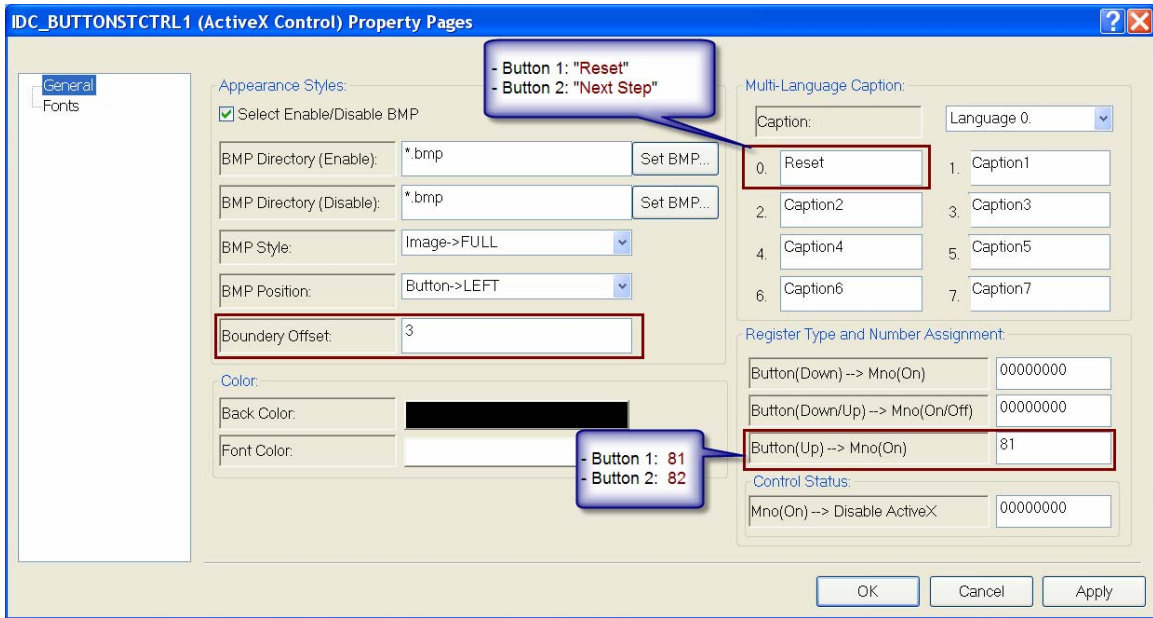


STEP 17: Open the property pages: Right click the ButtonST object and select “Properties”. Click the “Property Pages” icon in the “Properties” window.

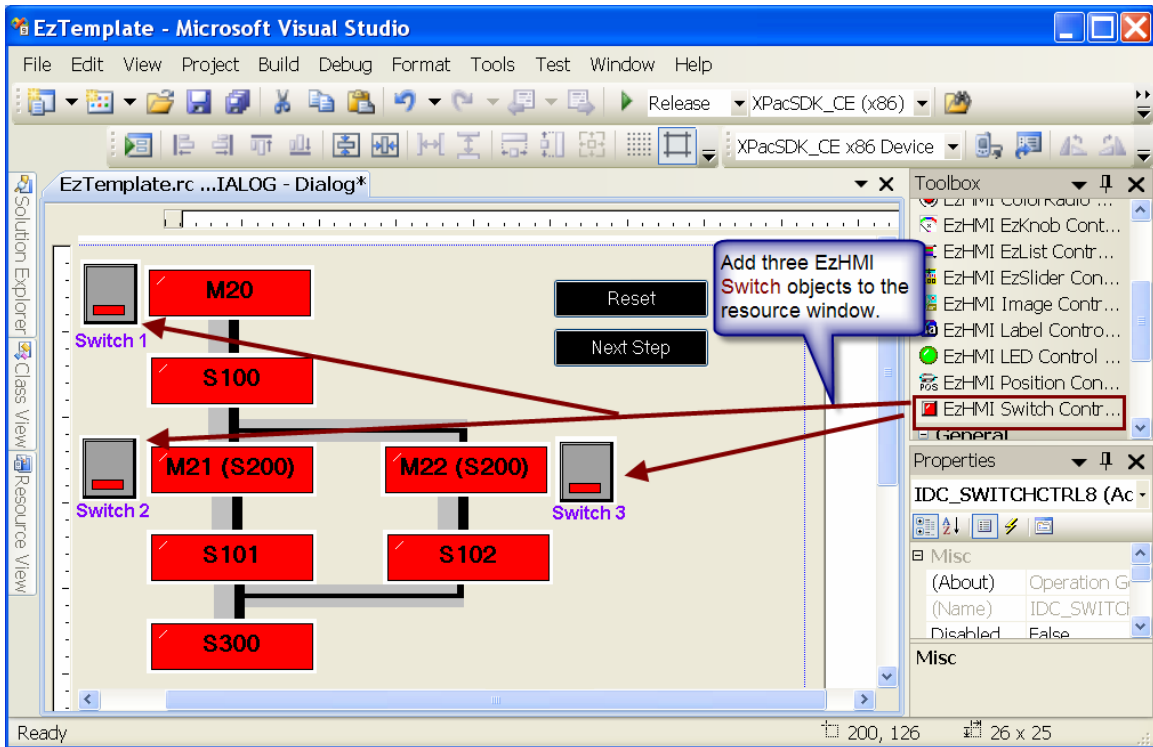
STEP 18: Set the “*Boundary Offset*” to 3 pixels.

STEP 19: Set the caption of the first button to “Reset” and of the second button “Next Step”.

STEP 20: Assign the first button the M register number 81 and the second button M register number 82 for “*Button(Up) →Mno(On)*” property.



STEP 21: Drag three EzHMI Switch objects (Switch 1 to Switch 3) onto the main dialog window as shown on the next figure.



STEP 22: Open the property pages: Right click the ButtonST object and select “Properties”. Click the “Property Pages” icon in the “Properties” window.

For all three switches do the following property settings:

STEP 23: Set the switch type to “**SWITCH TOGGLE**”.

STEP 24: Select yellow as the text color for the on state (“**Status Text On Color**”).

STEP 25: Enable the “**Display Caption**”.

STEP 26: Assign the switches the following captions (in text box 0.):

Switch	Caption
Switch 1	M20
Switch 2	M21
Switch 3	M22

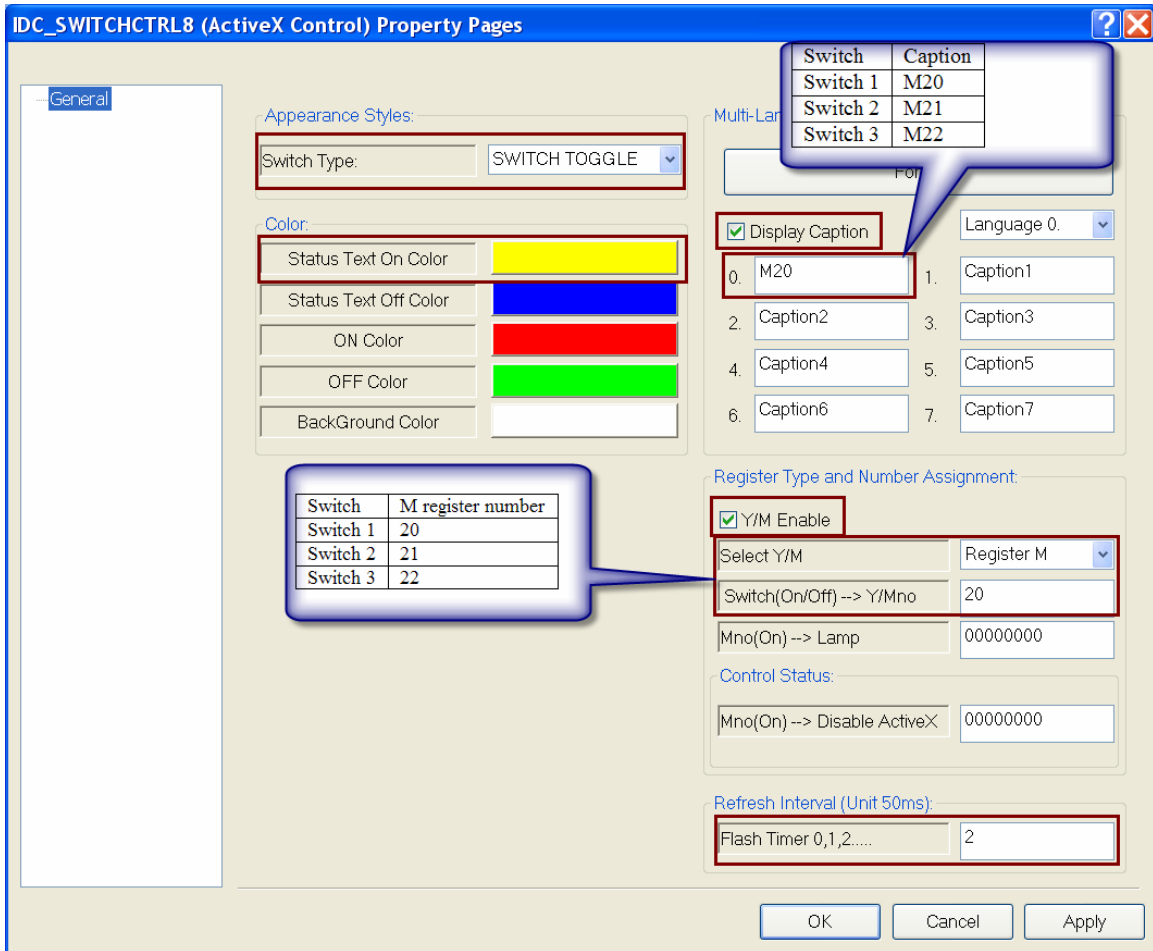
STEP 27: Check the “**Y/M Enable**”

STEP 28: Link the switches to the following M register numbers:

Switch	M register number
Switch 1	20
Switch 2	21
Switch 3	22

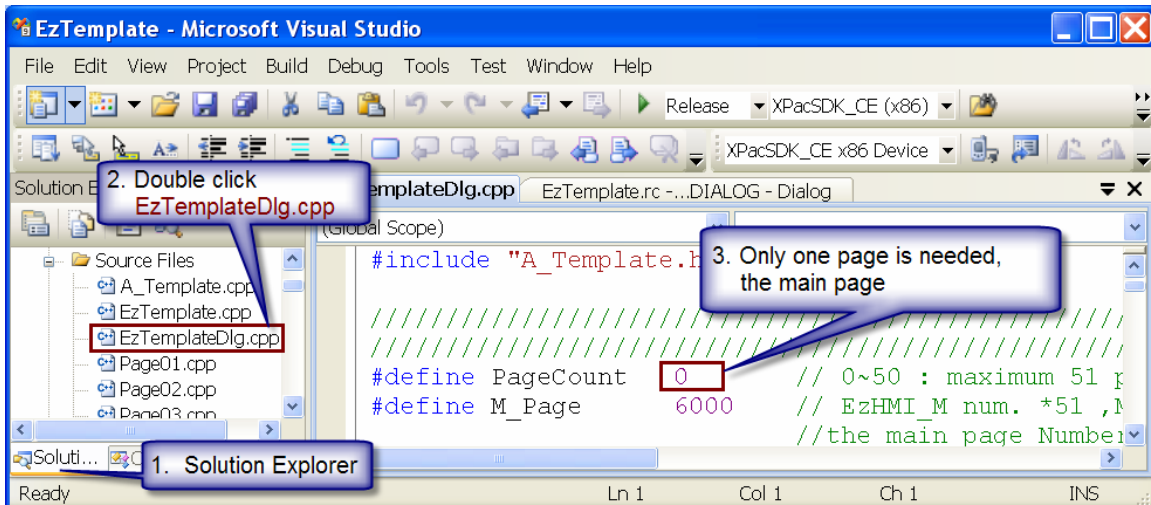
STEP 29: Set the **Switch** objects refresh rate to 100 milliseconds.

- Assign “**Flash Timer 0,1,2...**” the value 2.

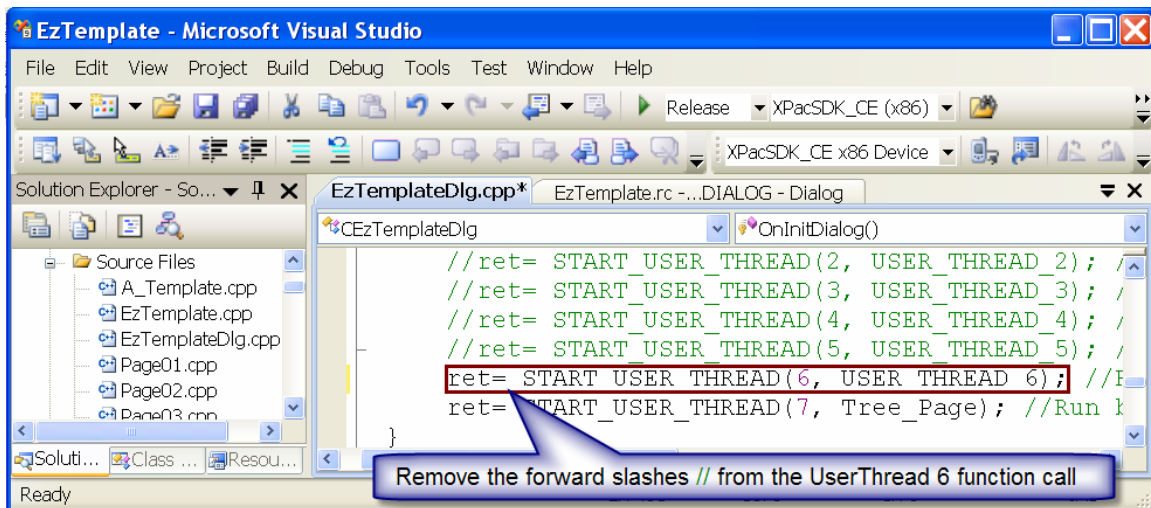


2.14.2 UserThread activation and code implementation

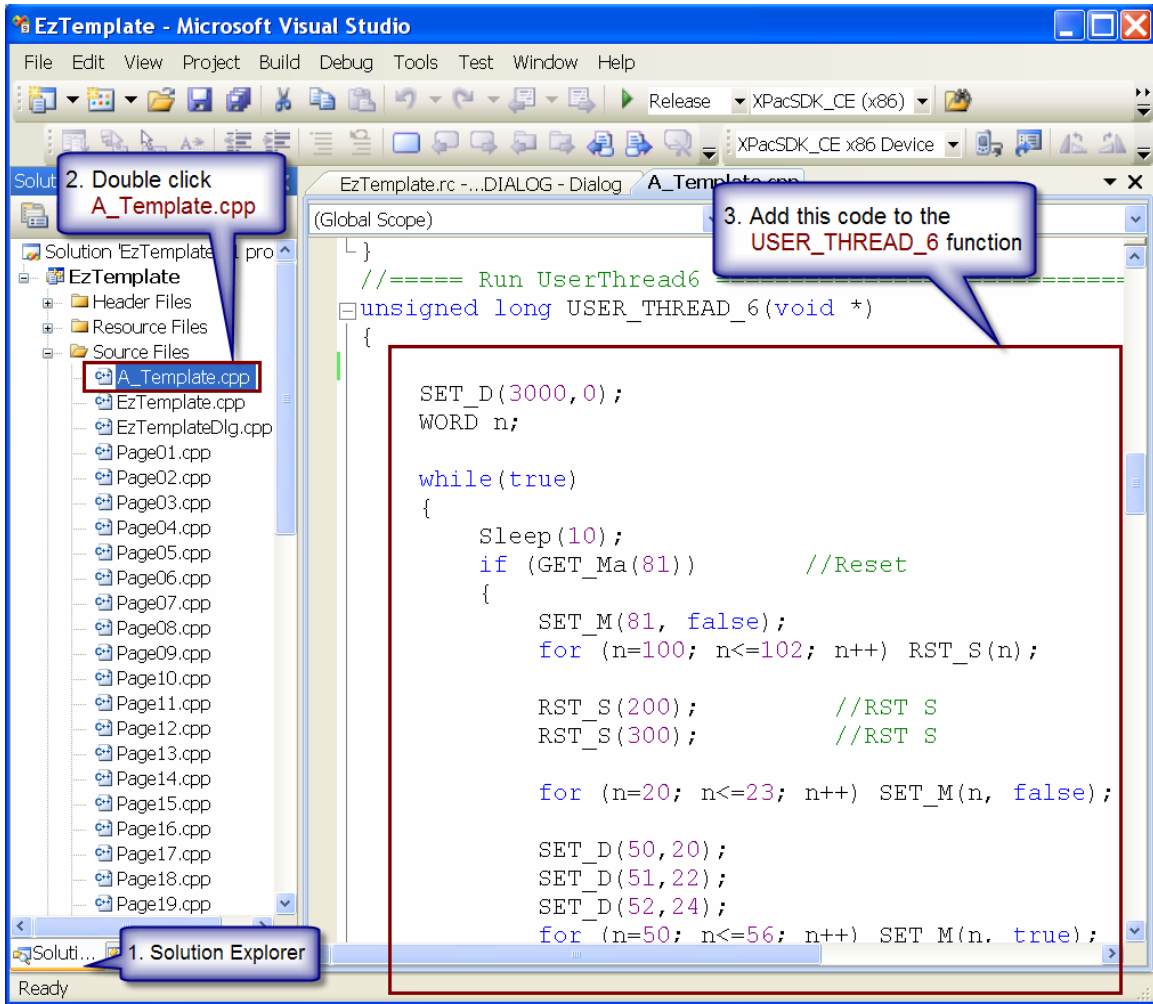
STEP 1: Only the main page is needed for this example therefore assign a zero to the PageCount definition.



STEP 2: The User_Thread_6 will be used to handle the multi-language setting. Remove the forward slashes from the USER_THREAD_6 function activation.



STEP 3: Add the following code to USER_THREAD_6 function



D3000 record Step value

```
while(true);  
SET_D(3000,0);  
while(true)  
{
```

Use a endless loop

Pause every loop for 10 milliseconds

```
Sleep(10);
```

```
if (GET_Ma(81))  
{
```

If the "Reset" button has been pressed and **released**

```
SET_M(81, false);
```

Reset the M register of the "Reset" button to *false*.

```
for (n=100; n<=102; n++) RST_S(n);
```

Reset the S register numbers 100 to 102 to

```
RST_S(200);
```

Reset the S register number 200 to *false*.
(Button "M21 (S200)" will be set to *false*.)

```
RST_S(300);
```

Reset the S register number 300 to *false*.
(Button "M21 (S200)" will be set to

```
for (n=20; n<=23; n++) SET_M(n, false);
```

Set the M register numbers 20 to 22 to *false*
→ The three switch buttons are set to off.

```
SET_D(50,20);
```

Link the dynamic Image to PIC20.bmp

```
SET_D(51,22);
```

Link the dynamic Image to PIC22.bmp

```
SET_D(52,24);
```

Link the dynamic Image to PIC24.bmp

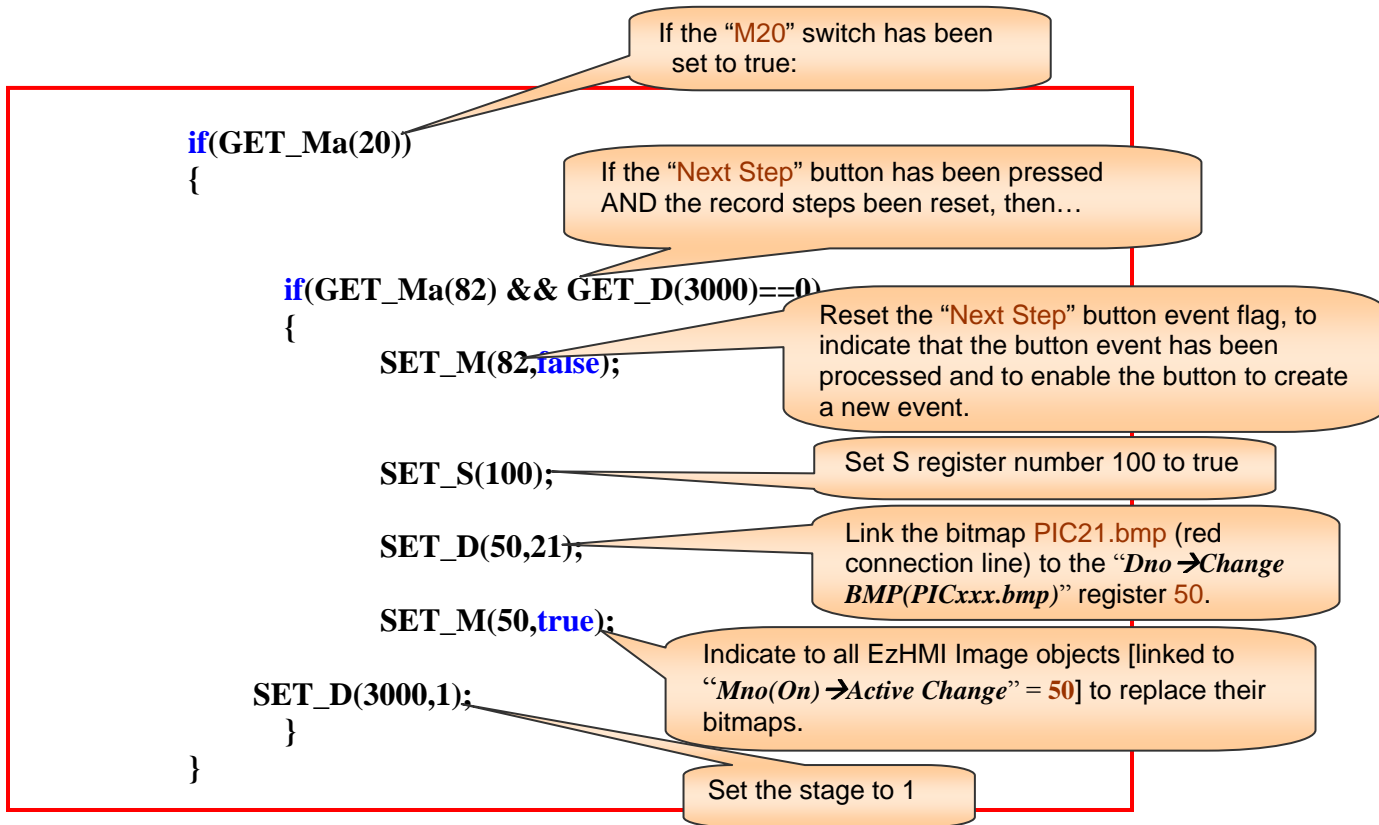
```
for (n=50; n<=56; n++) SET_M(n, true);
```

```
SET_D(3000,0);
```

Inform the EzHMI Image object to display new the dynamic images

```
}
```

Reset the record value to zero



If the S register number 100 is true
(the "S100" LED is on):

```
if(GET_S(100))  
{
```

If the "Next Step" button has been pressed
AND the record steps is set to 1

```
if(GET_Ma(82) && GET_D(3000)==1)  
{
```

```
SET_M(82,false);
```

Indicate that the "Next Step" button event has
been processed and reset the event flag.

```
SET_S(200);
```

Set the S register number 200 to true

```
SET_D(51,23);
```

Link the bitmap *PIC23.bmp* (red
connection line) to the "*Dno* → *Change
BMP(PICxxx.bmp)*" register 51.

```
SET_M(51,true);
```

Indicate to the EzHMI Image object [linked to
"*Mno(On)* → *Active Change*" = 51] to replace its
bitmap.

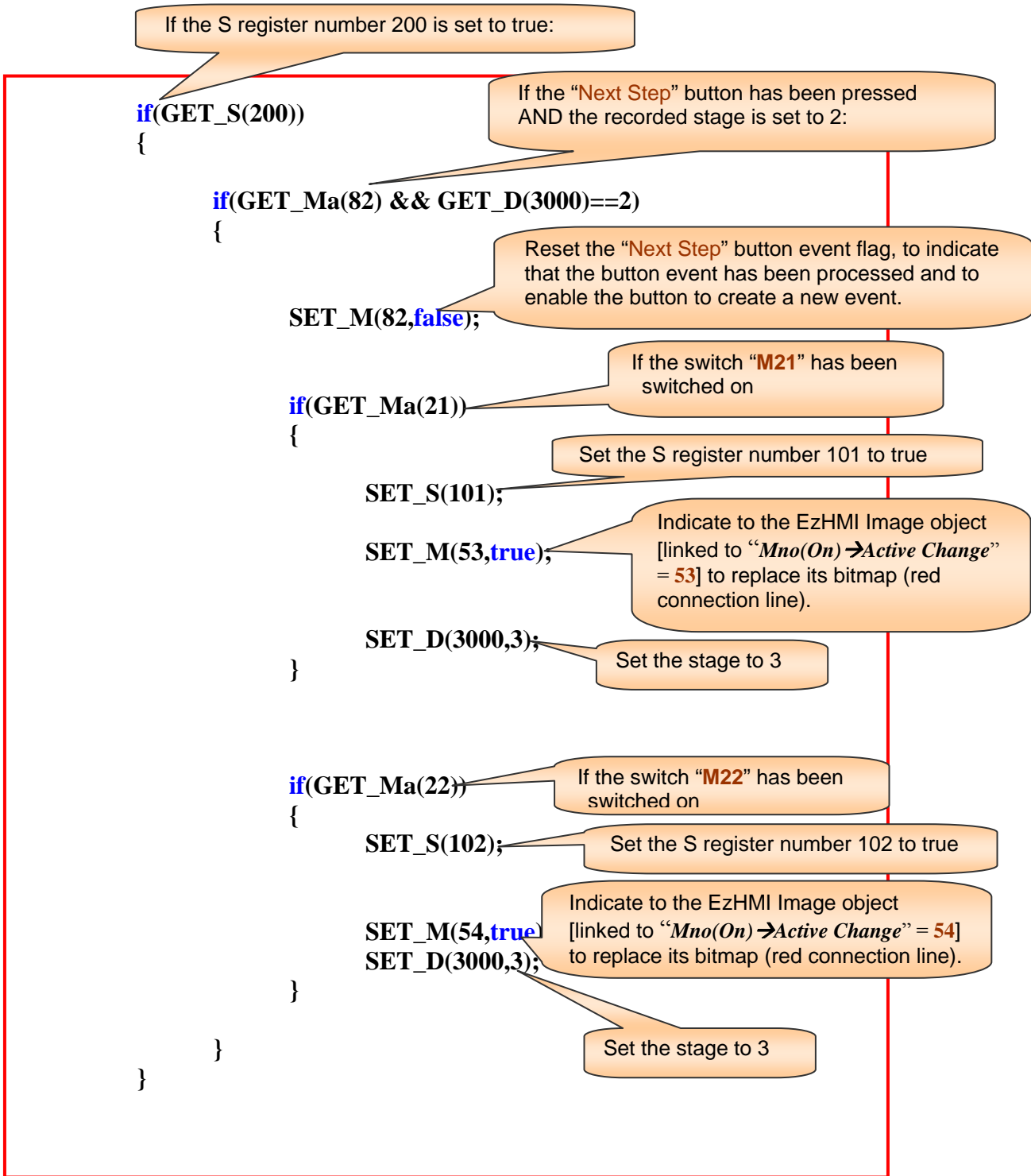
```
SET_M(52,true);
```

Indicate the EzHMI Image objects to
replace its bitmap.

```
SET_D(3000,2);
```

Set the stage to 2

```
}
```



If the S register number 101 is set to true:

```
if(GET_S(101))  
{
```

If the "Next Step" button has been pressed
AND the recorded stage is set to 3:

```
if(GET_Ma(82) && GET_D(300)==3)
```

```
{
```

Reset the "Next Step" button event flag, to indicate that
the button event has been processed and to enable the
button to create a new event.

```
SET_M(82,false);
```

Set the S register number 300 to true.

```
SET_S(300);
```

```
SET_M(55,true);
```

Indicate to the EzHMI Image object [linked to
"Mno(On) →Active Change" = 55] to replace its
bitmap (red connection line).

```
SET_D(3000,4);
```

Set the stage to 4

```
}
```

```
}
```

```
if(GET_S(102))
```

If the S register number 102 is set to true:

```
{
```

If the "Next Step" button has been pressed AND the
recorded stage is set to 3:

```
if(GET_Ma(82) && GET_D(3000)==3)
```

```
{
```

Reset the "Next Step" button event flag, to indicate that
the button event has been processed and to enable the
button to create a new event.

```
SET_M(82,false);
```

Set the S register number 300 to true.

```
SET_S(300);
```

```
SET_D(50,28);
```

Link the bitmap PIC28.bmp (red
connection line) to the "Dno →Change
BMP(PICxxx.bmp)" register 50.

```
SET_M(55,true);
```

Indicate to the EzHMI Image object
[linked to "Mno(On) →Active
Change" = 55] to replace its bitmap
(red connection line).

```
SET_D(52,25);
```

Link the bitmap PIC25.bmp (red
connection line) to the "Dno →Change
BMP(PICxxx.bmp)" register 52.

```
SET_M(56,true);
```

Indicate to the EzHMI Image object
[linked to "Mno(On) →Active
Change" = 56] to replace its bitmap
(red connection line).

```
SET_D(3000,4);
```

Set the stage to 4

```
}
```



```
}  
  
}
```

Make sure that all the bitmaps are downloaded to the following directory of the PAC:

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

Compile the project and download the execution file to the PAC as described in chapter 2.2.3. The following figure shows the user interface on the PAC.

