Develop your own c-function and c-function blocks in the ISaGRAF WinCE PAC?

- **Description of the Application**

  The following ISaGRAF WinCE PAC support user to develop his own c-function and c-function blocks to work together with the ISaGRAF SoftLogic.

VP-2xW7/2xW6, VP-4137:	1.53 or later
WP-5147/WP-5146:	1.07 or later
WinPAC-8xx7:	1.61 or later
XP-8xx7-Atom-CE6:	1.02 or later
XP-8xx7-CE6:	1.41 or later

If your PAC 's ISaGRAF driver version is older than above version, please visit http://www.icpdas.com/en/download/show.php?num=368&nation=US&kind1=&model=&kw=isagraf to download the new driver and then follow the PDF file inside the ZIP file to update it.

- **Description of the Files**

  This paper “FAQ167.pdf” (and “FAQ167_demo.zip”) can be downloaded from the following web site https://www.icpdas.com/en/faq/index.php?kind=280#751 > FAQ-167. The “FAQ167_demo.zip” includes “FAQ167.pdf” and some EVC++ 4.0 project inside for building a DLL which contains your own c-function and c-function blocks.

  The standard ISaGRAF driver for ICP DAS WinCE PAC doesn't include the following “User_c” DLL. When you create your own c-function and c-function blocks to work with the ISaGRAF logic, you create the following DLL file. Please copy them to the WinCE PAC and reset the PAC once. Then the ISaGRAF driver will load that DLL to support your c-function and c-function blocks.

  DLL file name for each PAC:

XP-8xx7-CE6	\System_disk\ISaGRAF\User_c_xpce6.dll
WP-8xx7	\System_disk\ISaGRAF\User_c_wp8.dll
WP-5147	\Micro_SD\ISaGRAF\User_c_wp5.dll
VP-25W7 / VP-4137	\System_disk\ISaGRAF\User_c_vp.dll
XP-8xx7-Atom-CE6	\System_disk\ISaGRAF\User_c_xpce6_atom.dll

1. How to develop your own c-function and c-function blocks?

The “FAQ167_demo_chinese.zip” includes DLL files which were created for users to develop their own C-function and C-function block. Users need to put the DLL file to the same folder as the ISaGRAF driver in the WinCE PAC (refer to the previous page), or using the following project to compile and build a DLL file.

For example, this paper uses a XP-8347-CE6 as an example. Users need to put the “User_c_xpce6.dll” file to the path “\System_disk\ISaGRAF\” and then reboot the PAC.

1.1. Select the goals for your project

The “FAQ167_demo_chinese.zip” includes the following projects that can be used to compile and build a DLL file. (Refer to Section 4.3 - Step 2).

For example, this paper uses a XP-8347-CE6 as an example to describe the EVC++ project (refer to Chapter 4). Users need to copy this “User_c_xpce6” folder to any location on your PC. This folder contains an EVC++ project to build the User_c DLL. Your PC must install the Microsoft Embedded Visual C++ 4.0 before you can compile this project.

- **The EVC++ 4.0 Projects**
  1. User_c_vp: VP-2xW7/2xW6, VP-4137
  2. User_c_wp5: WP-5147/WP-5146
  3. User_c_wp8: WinPAC-8xx7
  4. User_c_xpce6: XP-8xx7-CE6
  5. User_c_xpce6_atom: XP-8xx7-Atom-CE6

- **The VS2008 Projects**
  1. User_c_vp: VP-2xW7/2xW6, VP-4137
  2. User_c_wp5: WP-5147/WP-5146
  3. User_c_wp8: WinPAC-8xx7
  4. User_c_xpce6: XP-8xx7-CE6
  5. User_c_xpce6_atom: XP-8xx7-Atom-CE6

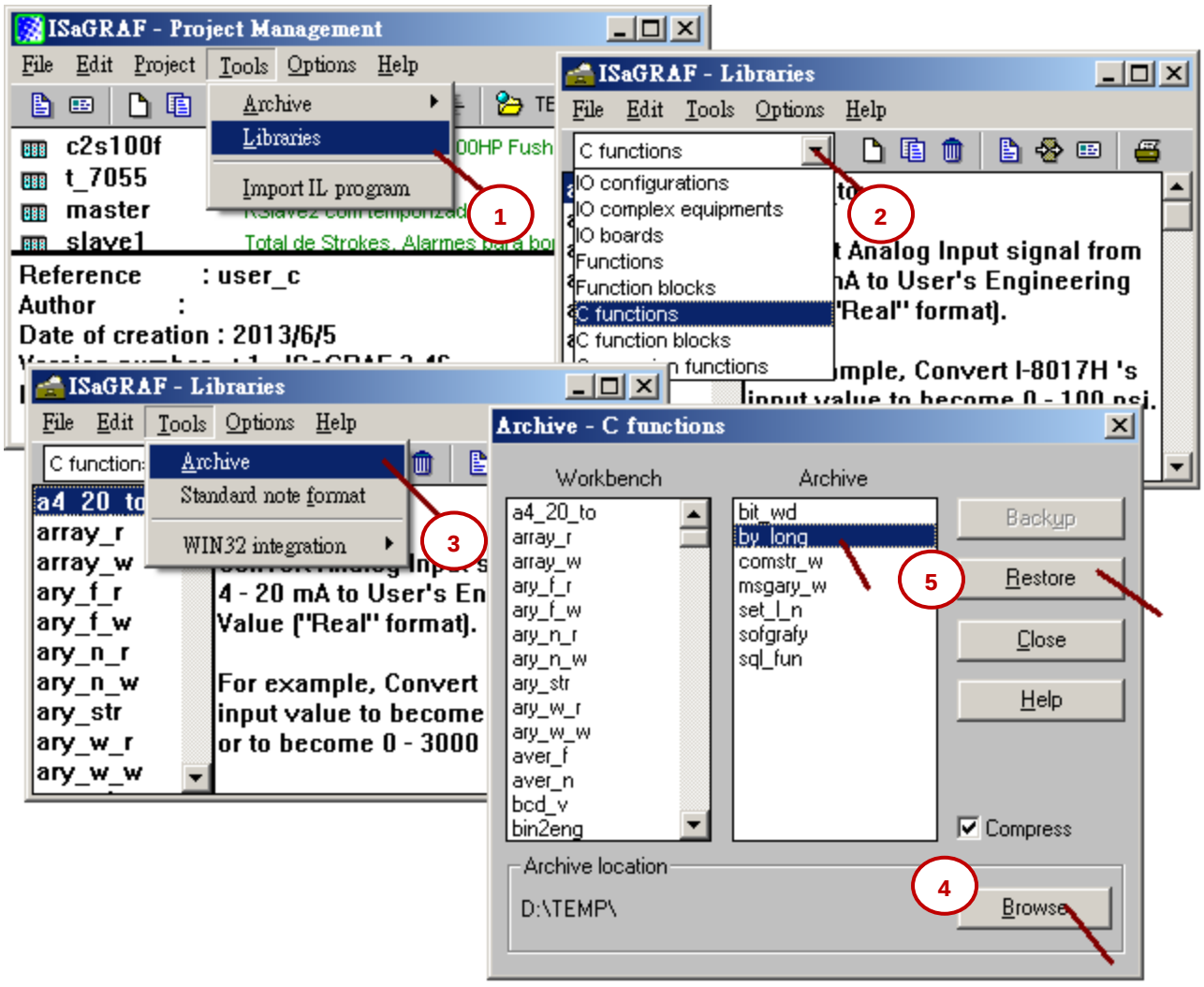
Moreover, this “FAQ-167” also provides the accomplished ISaGRAF demo programs.
(i.e., **C-function file** - “by_long” ; **C-function block file** - “long_by” ; **ISaGRAF Project** - “user_c.pia”). Refer to the Section 1.2 to restore these files. Then, users can review these finished files or follow the instructions in the Chapter 3 to create C-function and C-function Block.

1.2. Restore the ISaGRAF demo programs

This paper uses a XP-8347-CE6 as an example. Users can follow step-by-step instructions in the Chapter 3 to create C-function and C-function block files. Or, you can also restore the following completed files to your PC / ISaGRAF (ISaGRAF Ver 3.46 or 3.55) for cross-referencing.

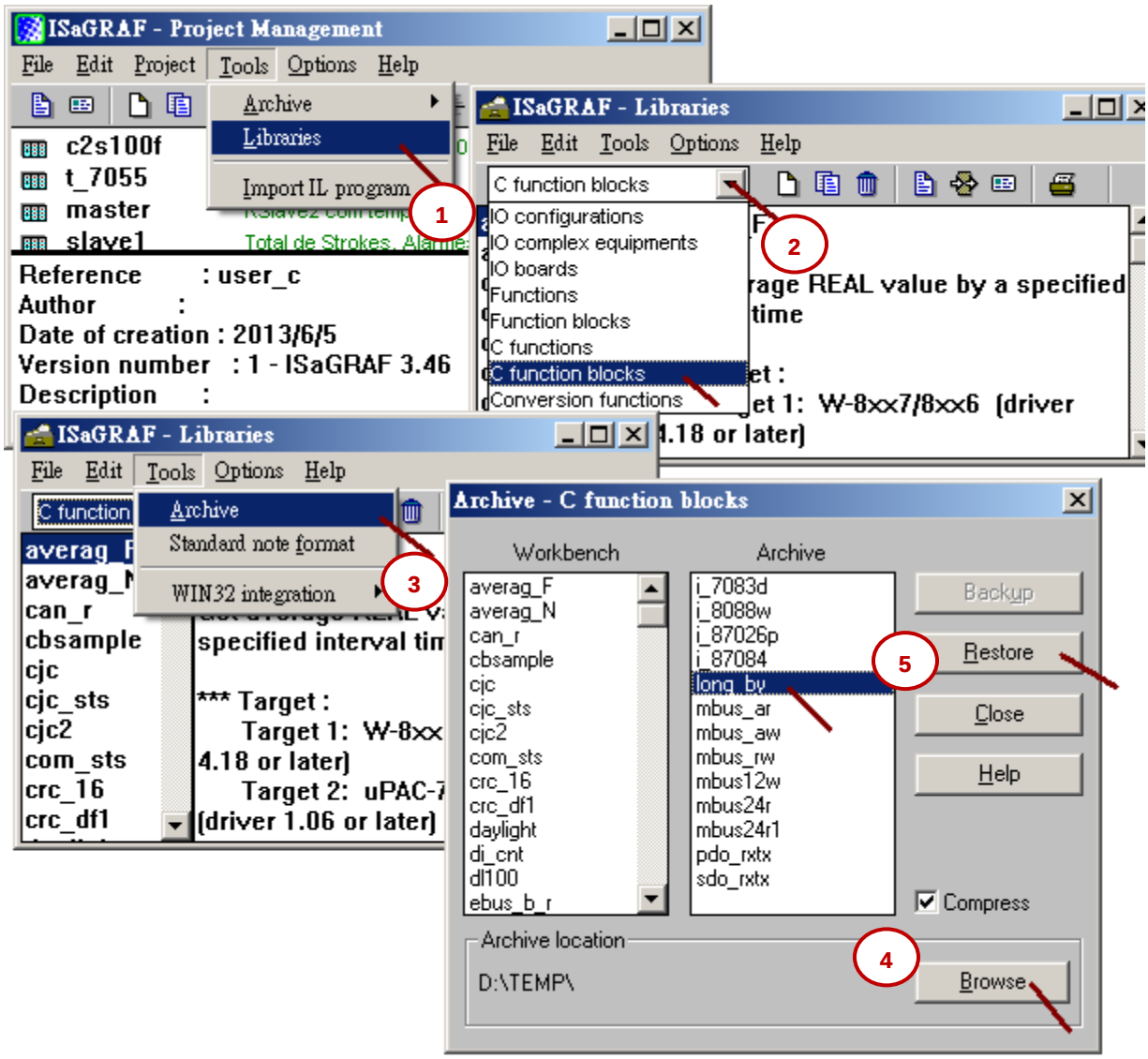
A. Restore the C-function - “by_long”

This file can be used to convert four bytes (0 ~ 255) to become one long integer (32-bit signed integer).



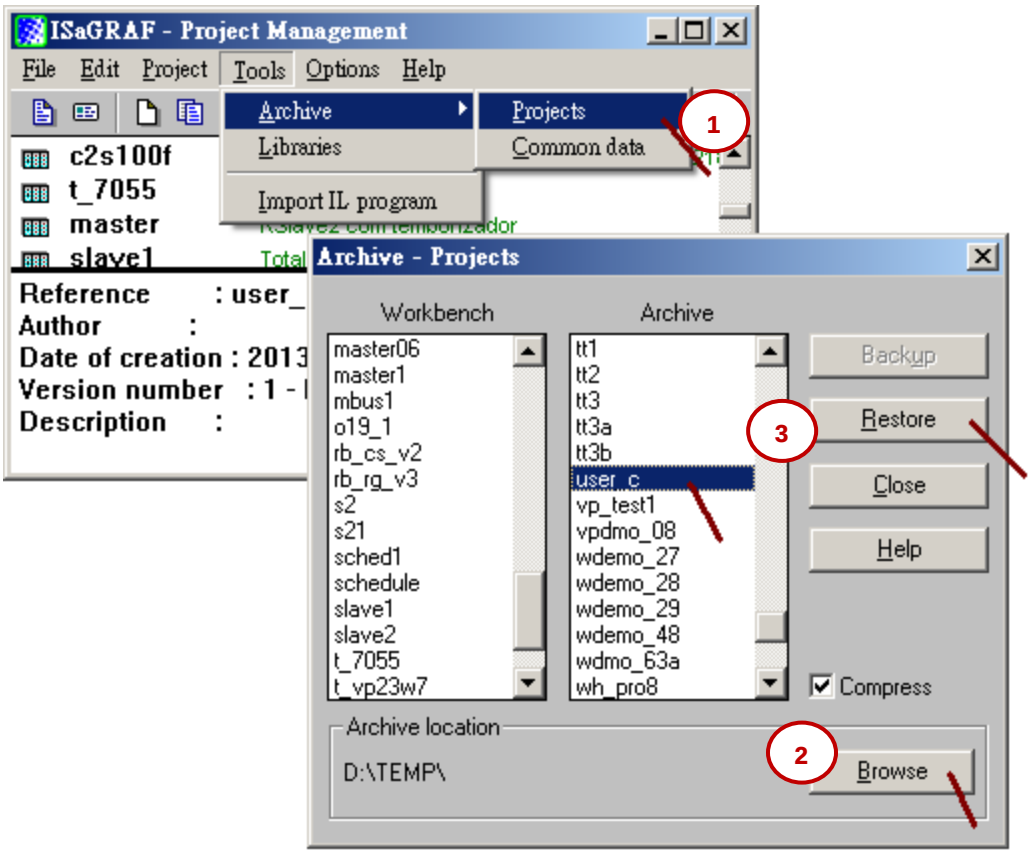
B. Restore the C-function block - "long_by"

This file can be used to convert four bytes to one long integer.



C. Restore the ISaGRAF Project - "user_c.pia"

This "user_c.pia" demo project uses the user-defined C-function - "by_long" and C-function block - "long_by".



After completing this procedure, recompile this ISaGRAF project - "user_c.pia" and then download it to the XP-8xx7-CE6 PAC to use these C-function - "by_long" and C-function block - "long_by".

2. Creating the compiler development environment

To develop the user-defined C-function or C-function block, the following development system is needed.

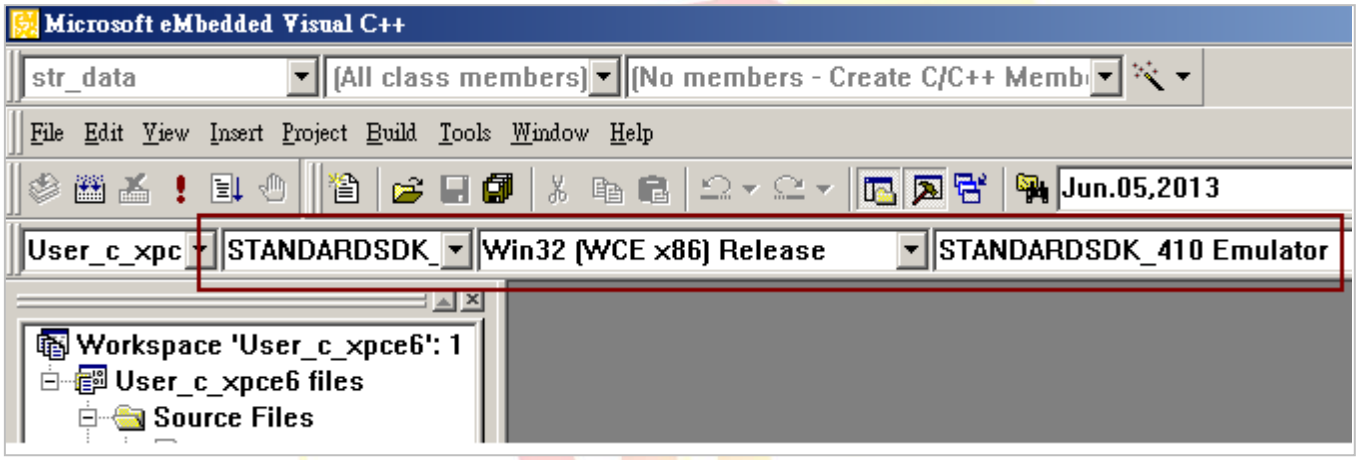
1. EVC++ 4.0 or
2. VS2008

2.1. EVC development environment

Make sure your EVC++ 4.0 project settings are correct for your PAC. (Note: the settings are different between different controllers.)

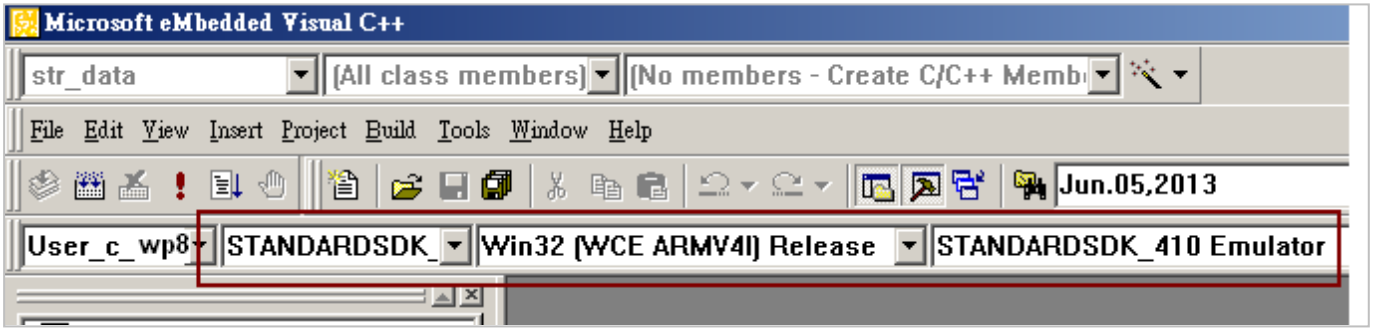
For XP-8xx7-CE6 and XP-8xx7-Atom-CE6:

It is “STANDARDSDK” and “Win32 (WCE **x86**) Release” and “STANDARDSDK_4xx Emulator”.



For WP-8xx7 and WP-5147 and VP-25W7 and VP-4137:

It is “STANDARDSDK” and “Win32 (WCE **ARMV4I**) Release” and “STANDARDSDK_4xx Emulator”.



2.2. VS2008 development environment

Download the SDK (Software Development Kit):

Go to the web page to download the related SDK:

1. For XPAC (XP-8xx7-CE6, XP-8xx7-Atom-CE6):

http://www.icpdas.com/en/download/show.php?num=2488&nation=US&kind1=&model=&kw=wince6

2. For ViewPAC (VP-2xW7/2xW6, VP-4137)、WinPAC (WP-5147/WP-5146, WP-8xx7):

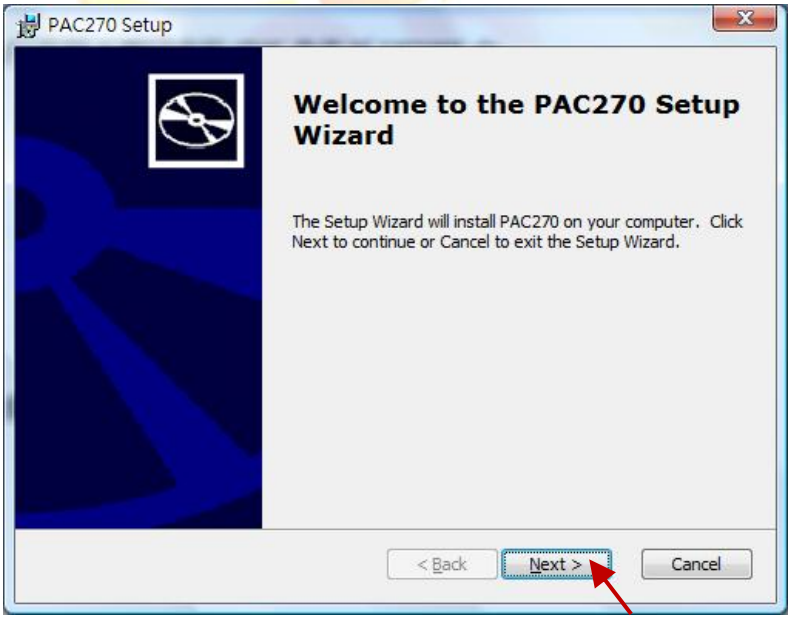
http://www.icpdas.com/en/download/show.php?num=2489&nation=US&kind1=&model=&kw=wince5

2.2.1. Install the SDK of the ViewPAC or the WinPAC

Note: Make sure your PC has been installed the Microsoft VS2008 before doing the following steps.

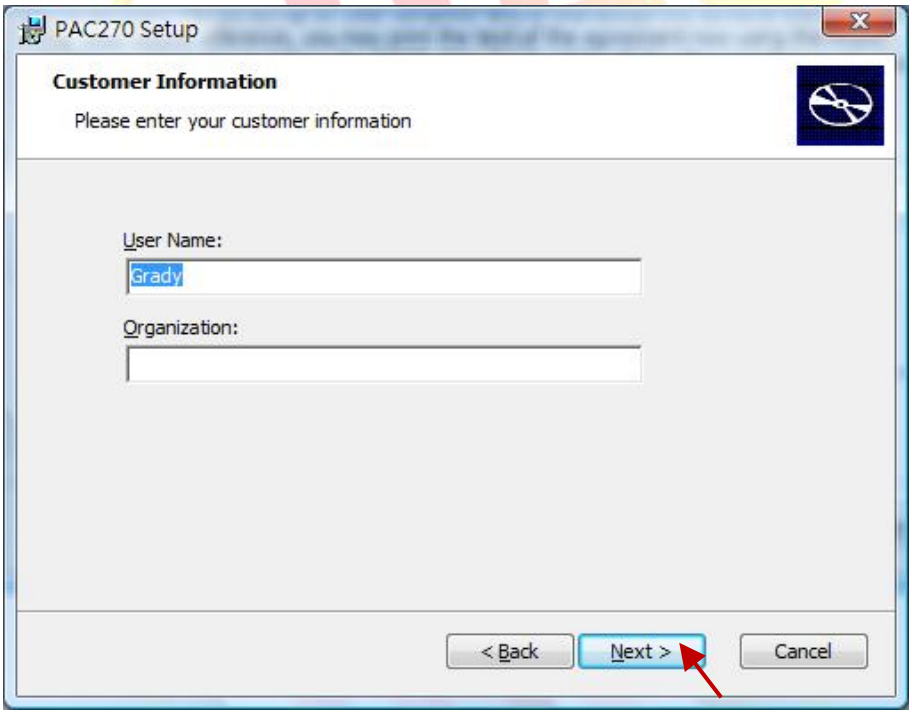
1. Double click the downloaded SDK file (e.g., pac270_sdk_20121015.msi) to install it to the VS2008.

2. Click the “Next” button.

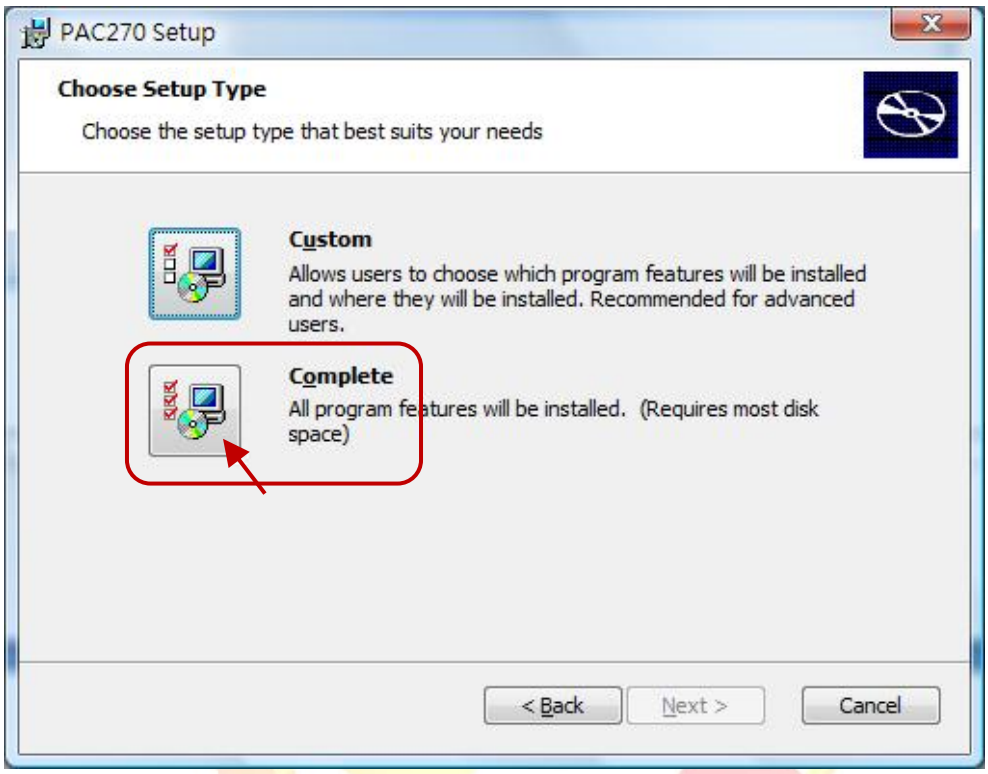


3. Choose the “Accept” radio button and then click the “Next” button.

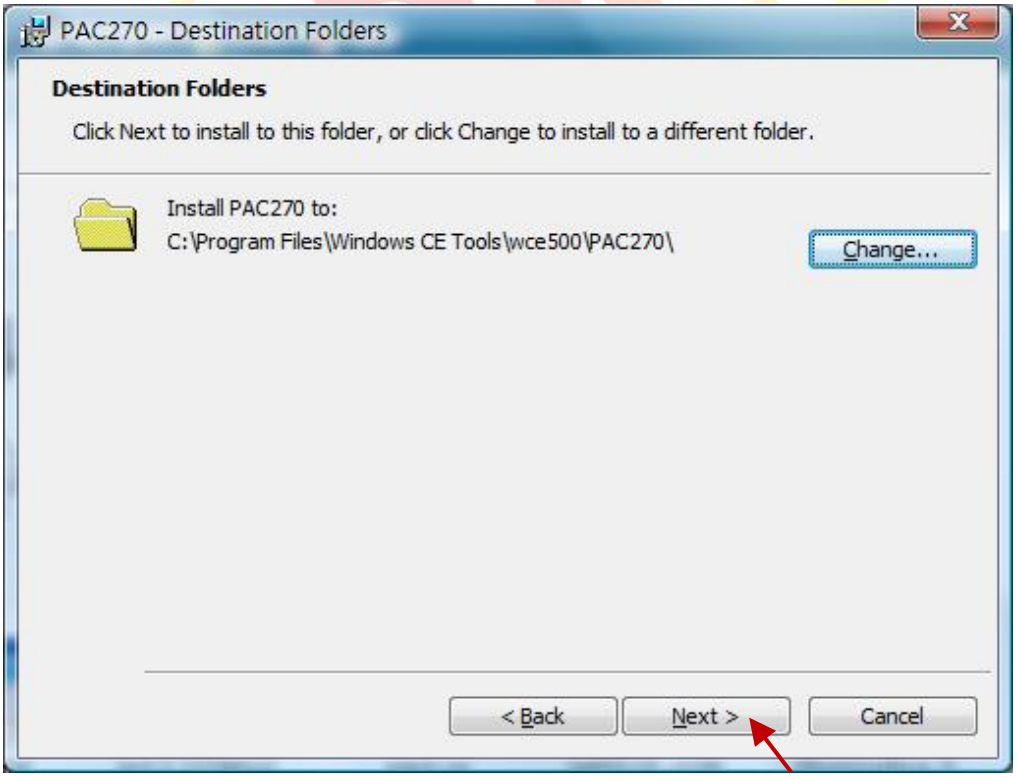
4. Click the “Next” button.



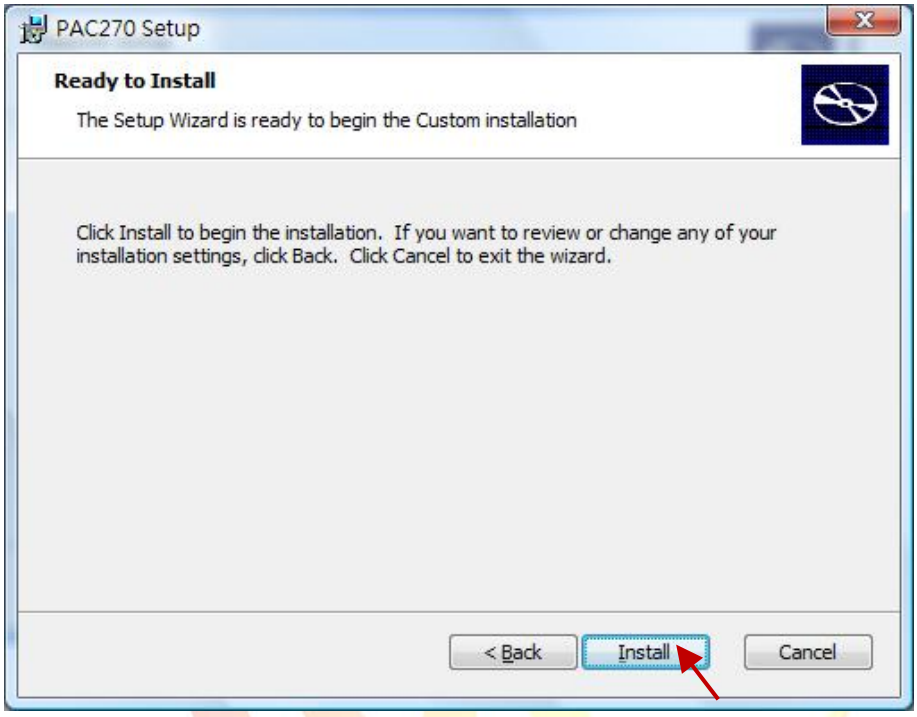
5. Click the “Complete” button.



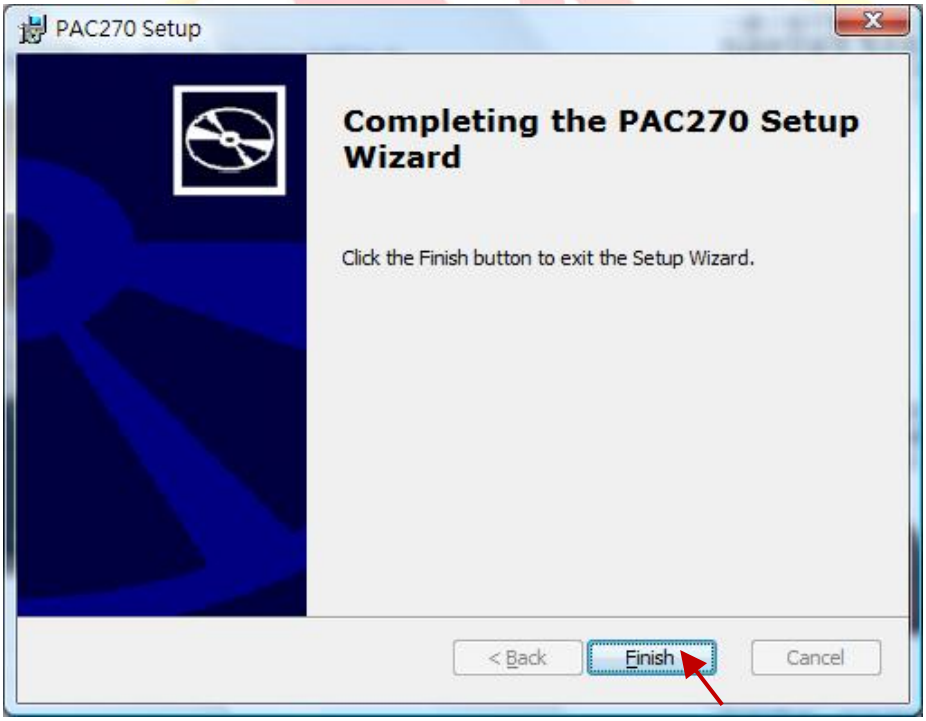
6. Click the “Next” button.



7. Click the “Install” button to install the SDK.



8. After completing the installation, click the “Finish” to quit the procedure.



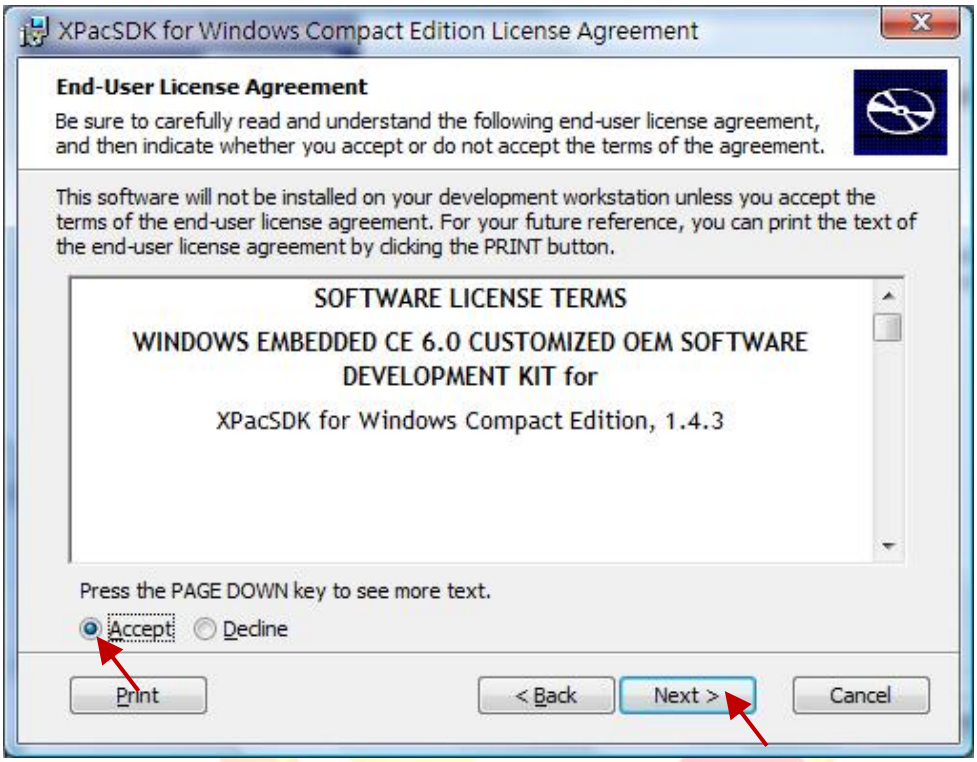
2.2.2. Install the SDK of the XPAC (XP-8xx7-CE6, XP-8xx7-Atom-CE6)

Note: Make sure your PC has been installed the Microsoft VS2008 before doing the following steps.

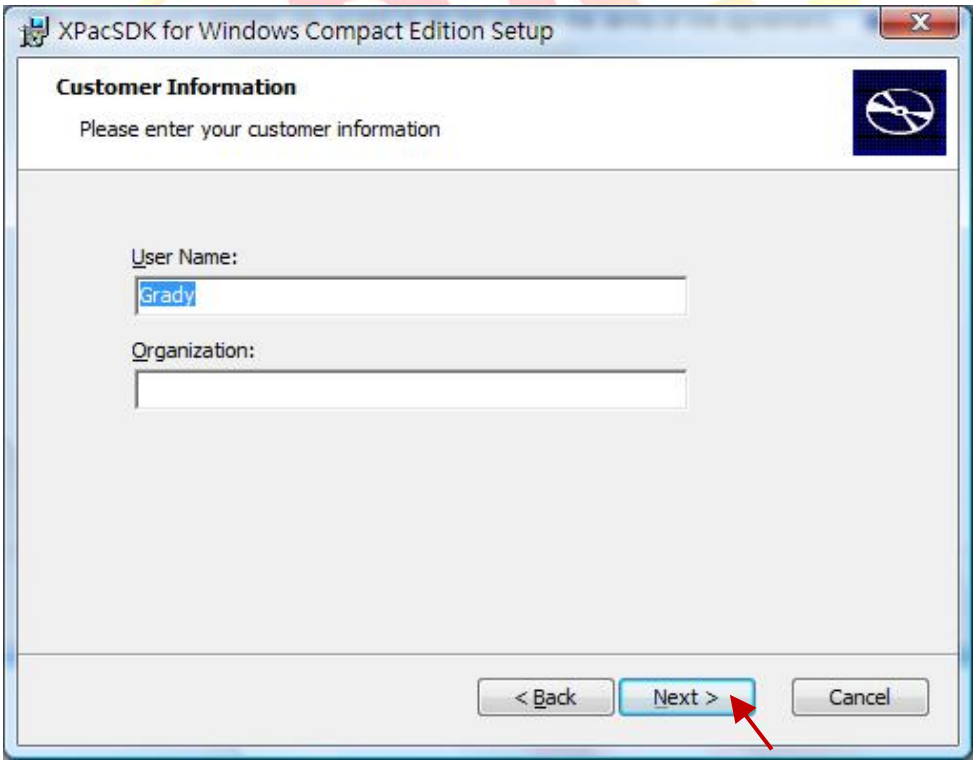
1. Double click the downloaded SDK file (e.g., pacsdk_ce_1.4.3_vs2008.msi) to install it to the VS2008.

2. Click the “Next” button.

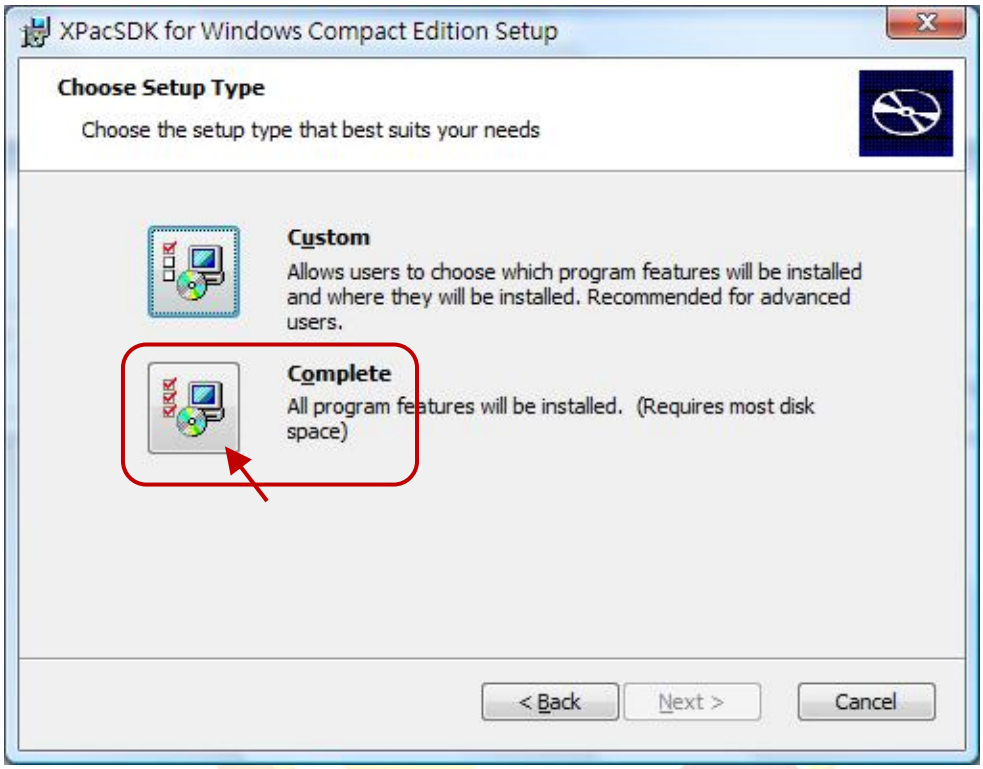
3. Choose the “Accept” radio button and then click the “Next” button.



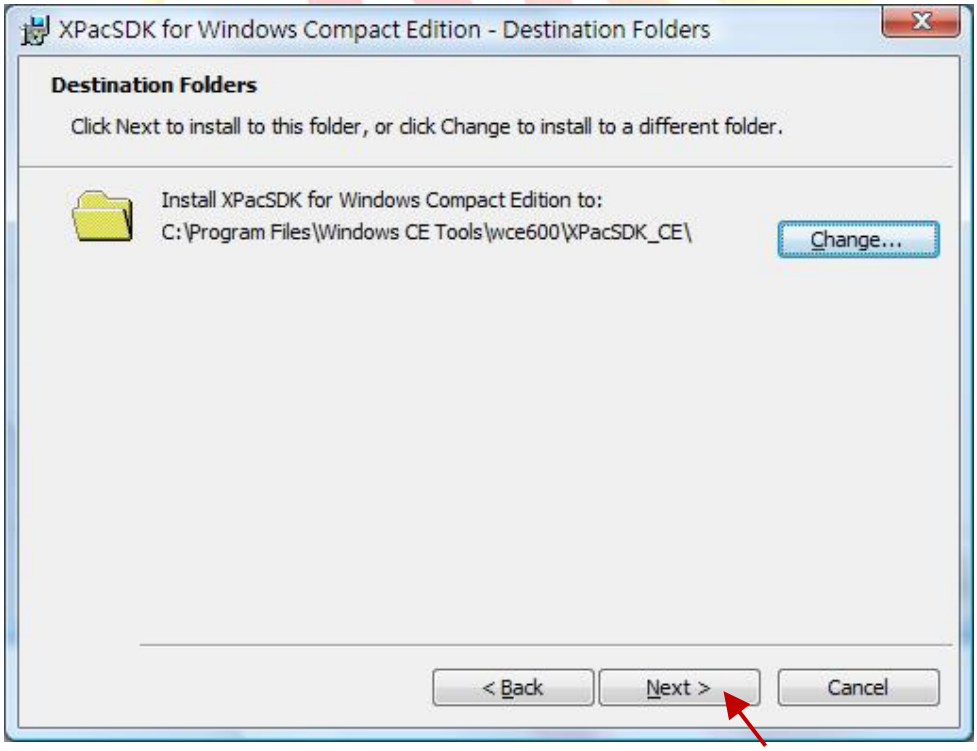
4. Click the “Next” button.



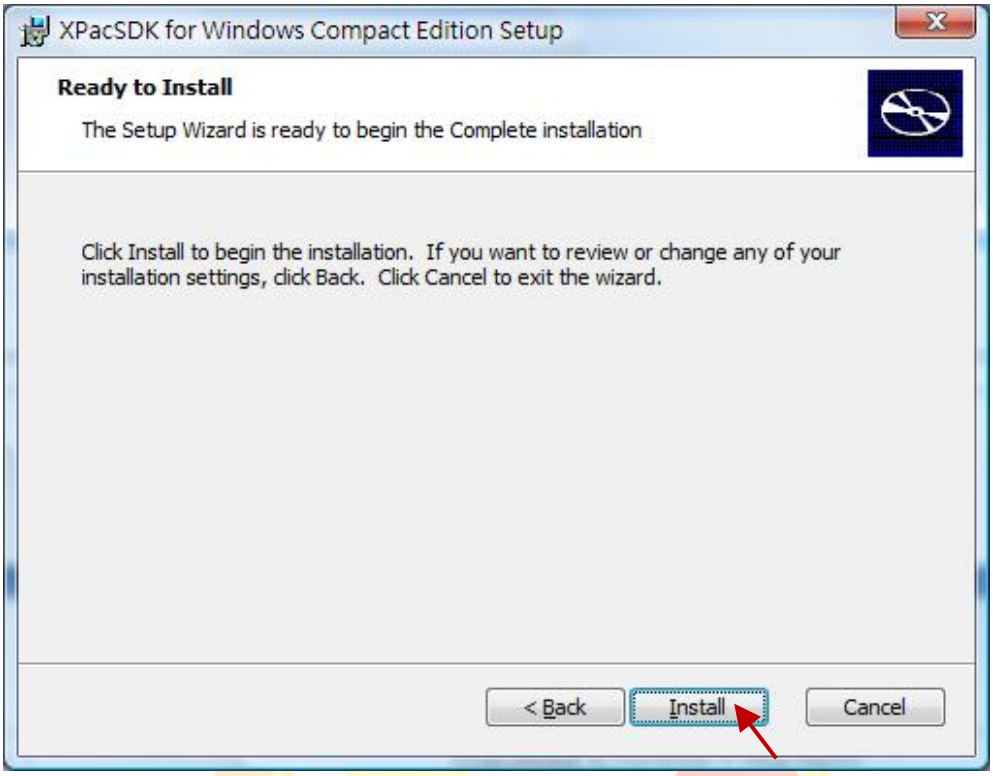
5. Click the “Complete” button.



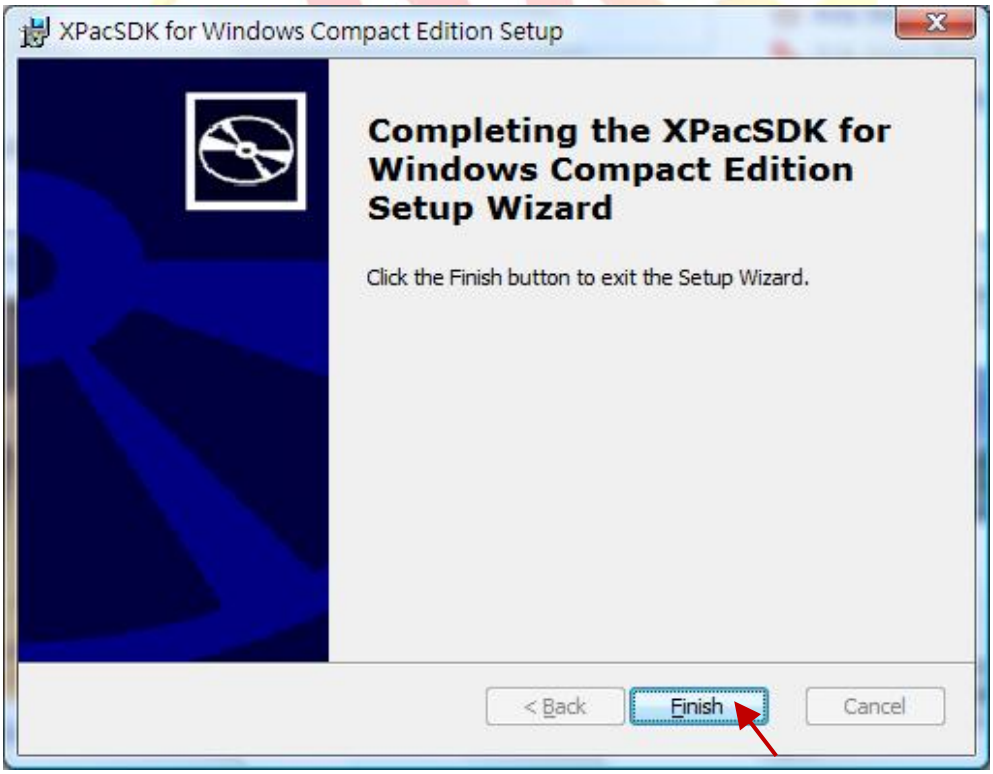
6. Click the “Next” button.



7. Click the “Install” button to install the SDK.



8. After completing the installation, click the “Finish” to quit the procedure.

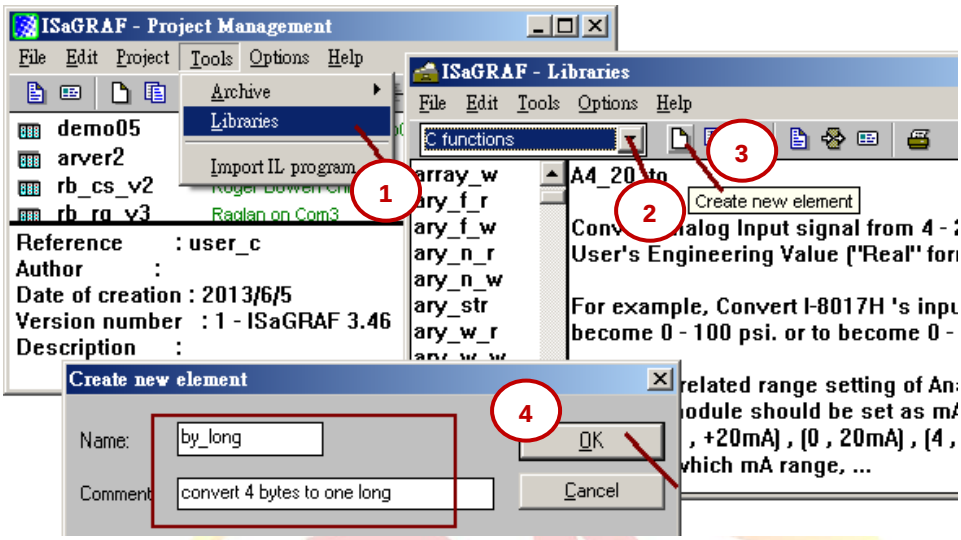


3. Define C-function or C-function block

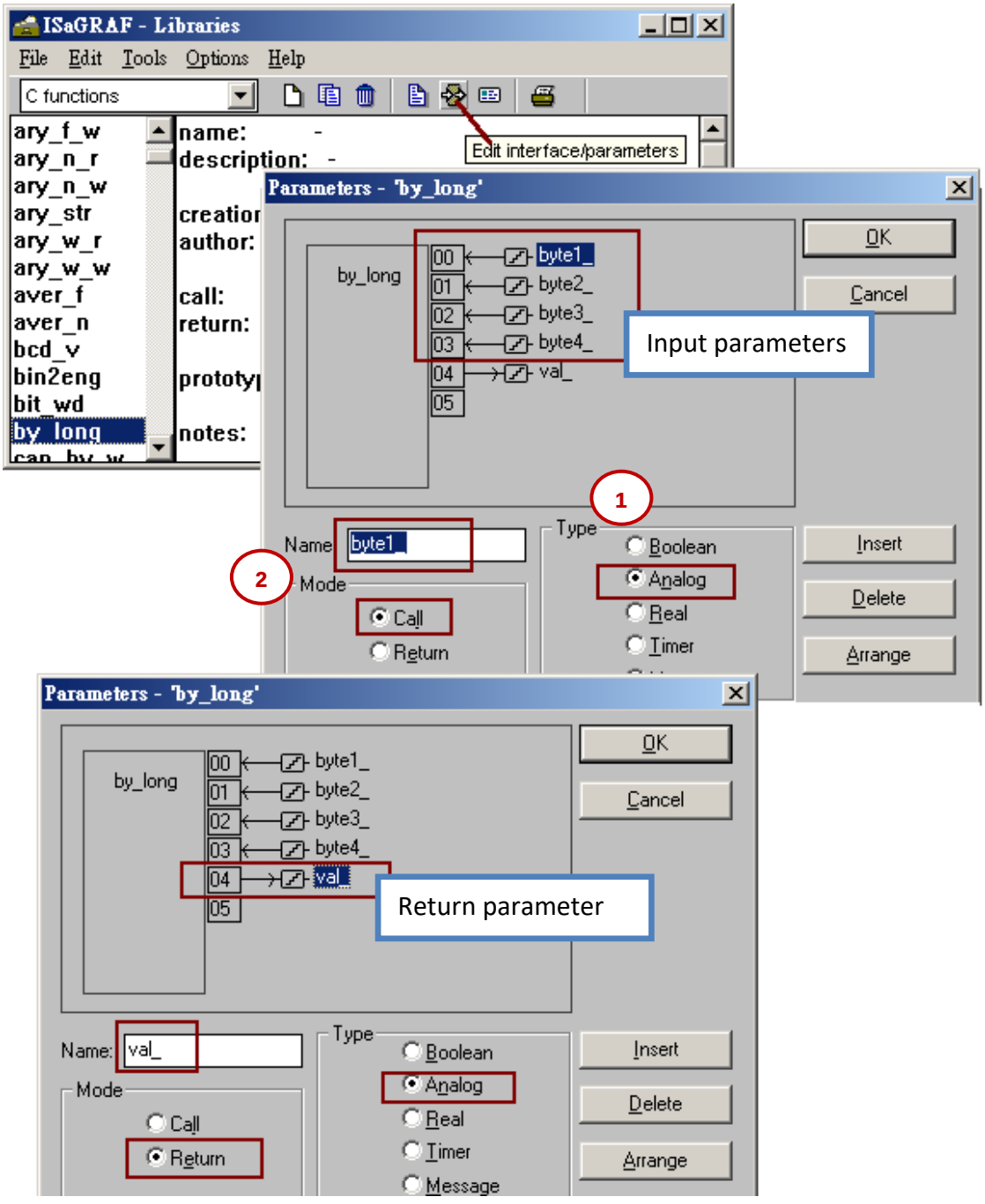
3.1. Define C-function Lib

To create a c-function in the User_c DLL, first define its ISaGRAF Lib. For example, the “by_long” is for converting four bytes to one long integer. The procedure is as the below steps.

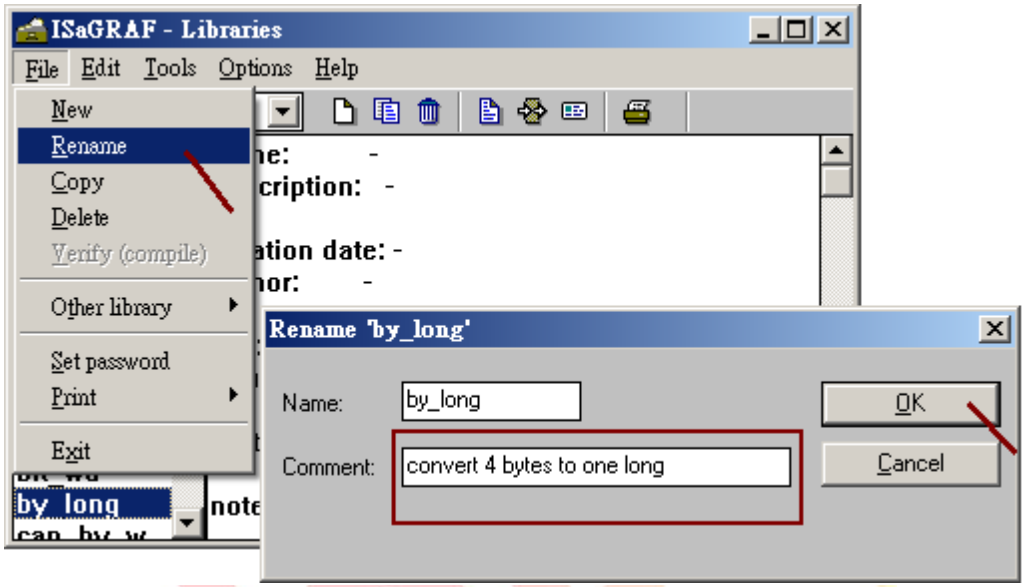
1. Create a new element of c-functions (e.g., “by_long”).



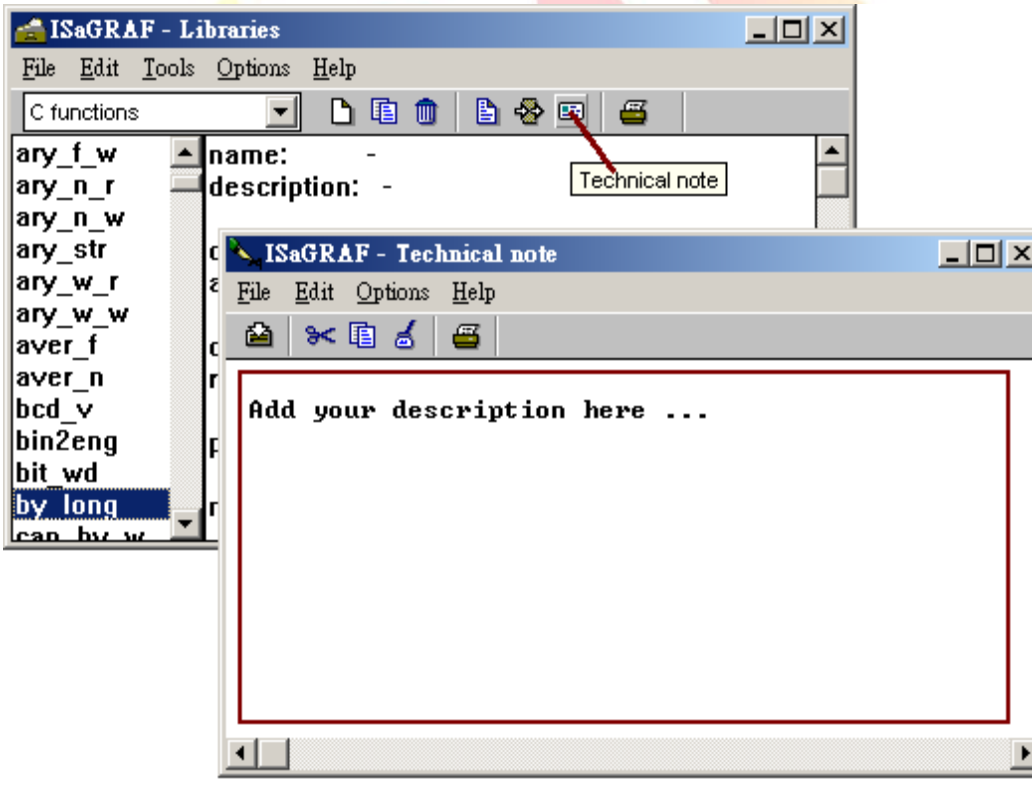
2. Define input and return parameters of the c-function:
   This example shows there are four input parameters : “byte1_”, byte2_”, “byte3_” and “byte4_”. Their type is “Analog” (means 32-bit signed integer). The “Mode” is “Call” (means it is “input” parameter). Please add a “_” char after each name . That is because the name of all parameters and return value will become a reserved word (means any ISaGRAF program cannot use these names as variable names).
   The return parameter here is “val_”. Its Mode setting is “Return”. Type is “Analog” (32-bit integer)



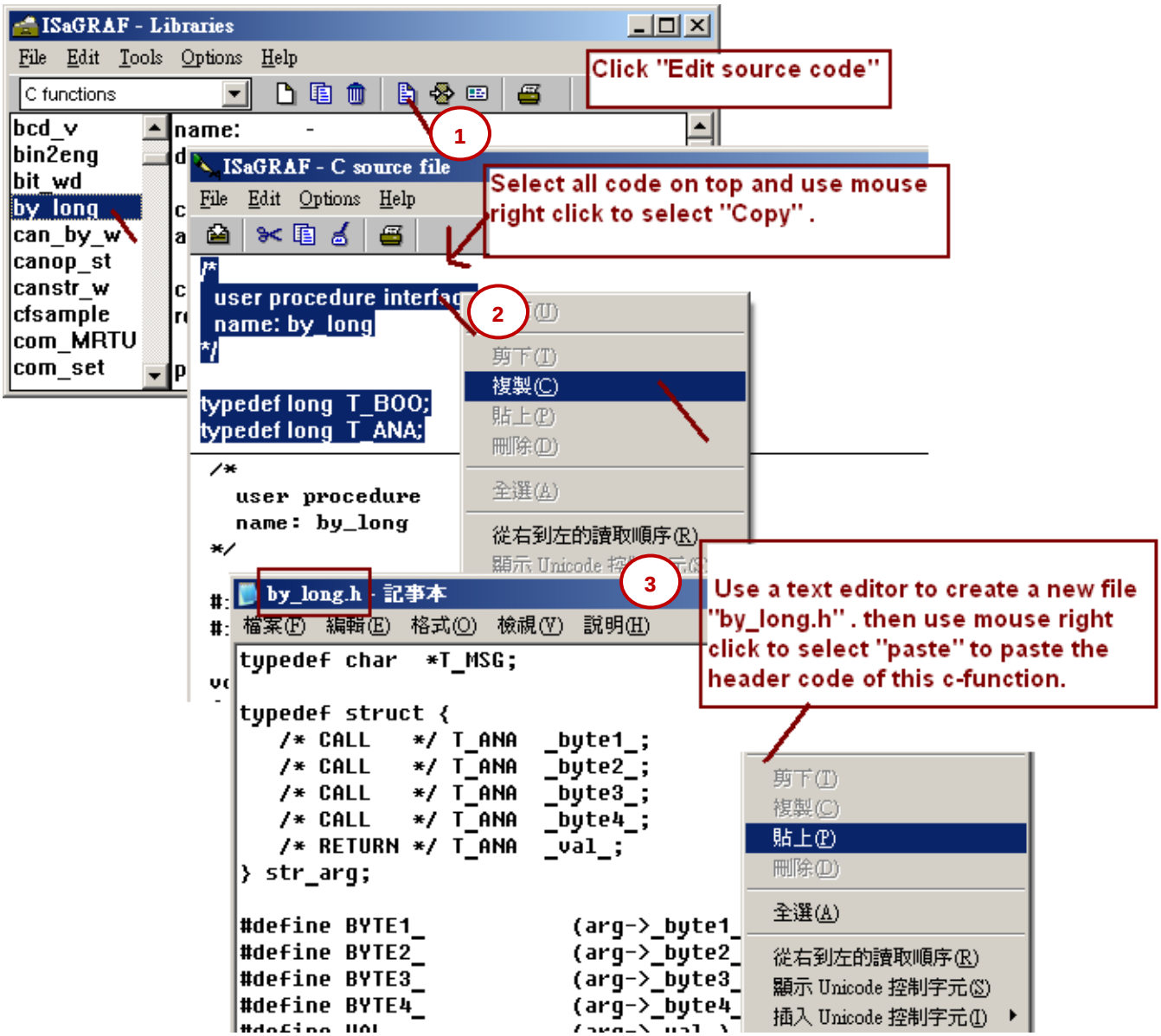
3. Add a comment for this c-function:



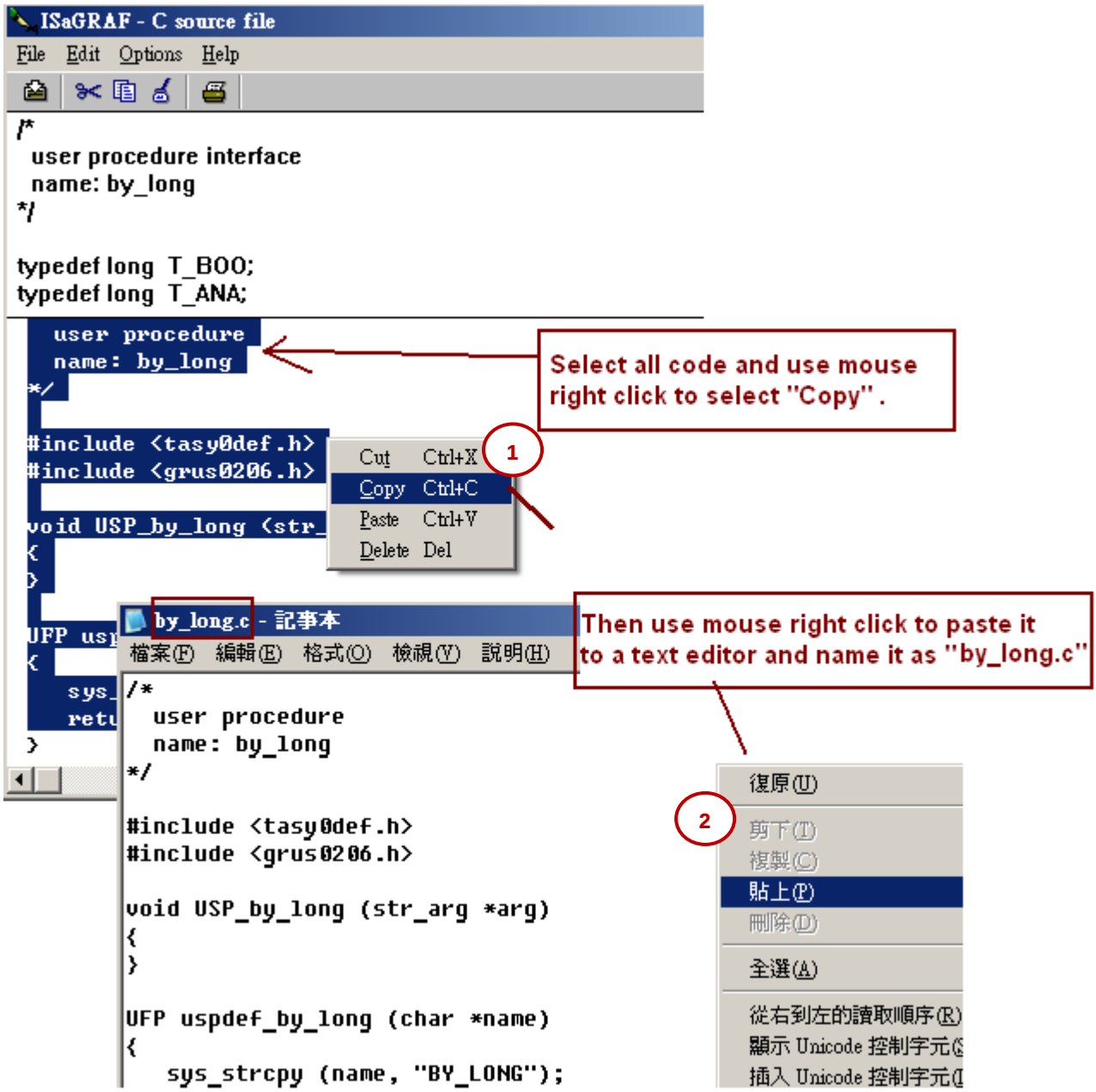
4. Add technical note:



5. Copy the prototype of the header file of the c function to a text editor and name it as "**by_long.h**".



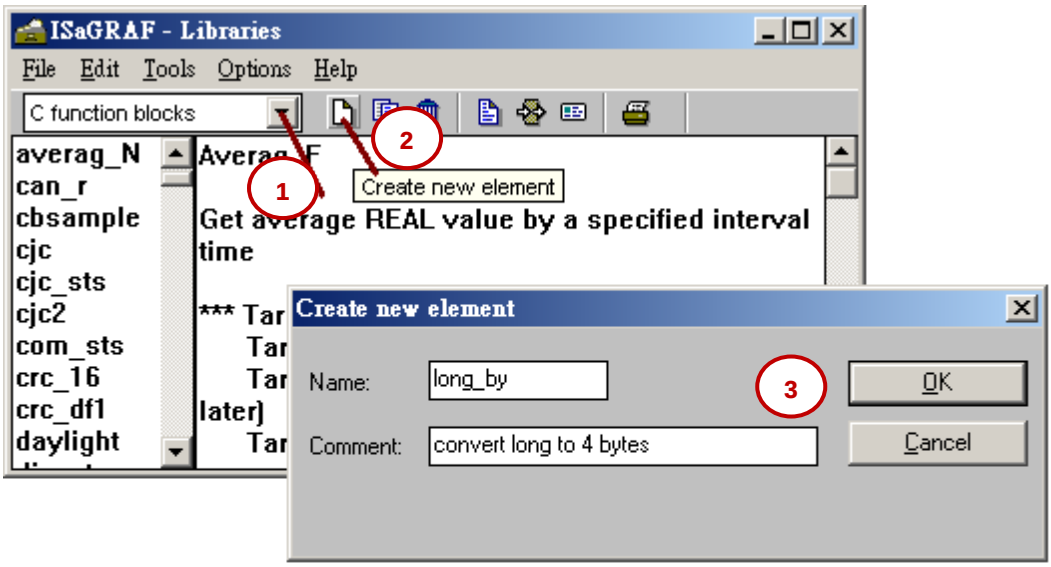
6. Copy the prototype of c source code of the c function to a text editor and name it as "**by_long.c**".

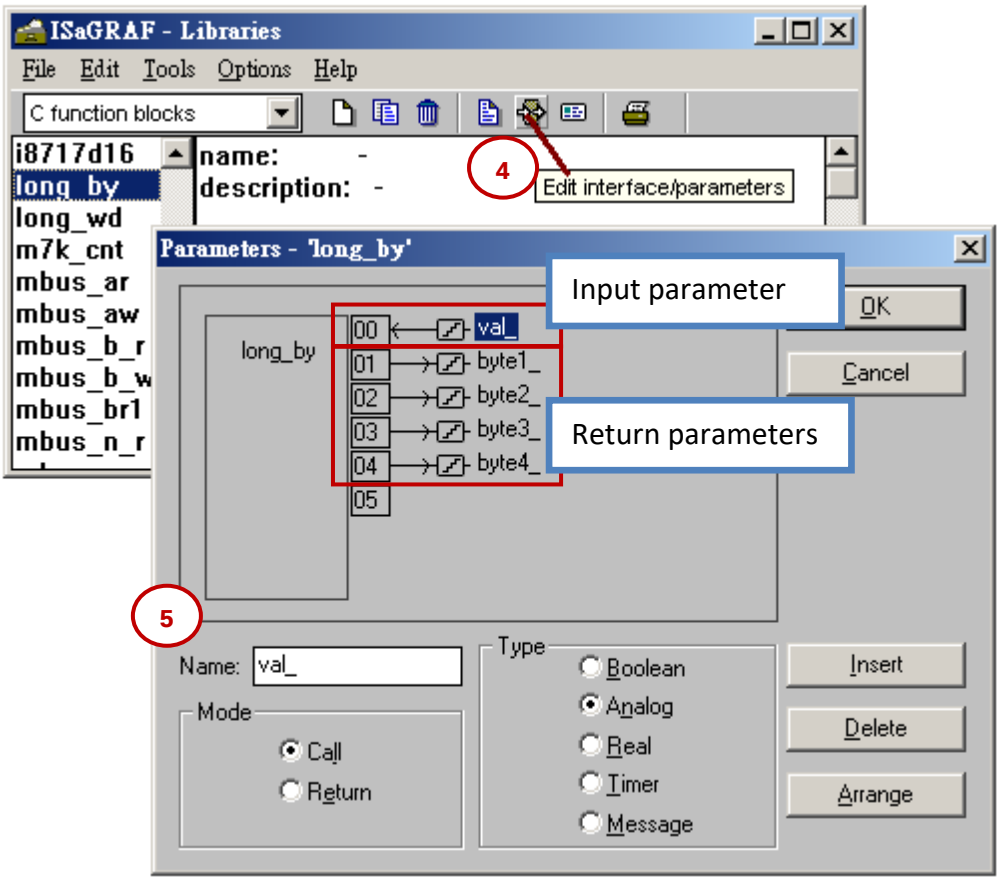


3.2. Define C-function block Lib

The procedure to create a c-function block is similar as the previous section. Please refer to it to define the c-function block Lib and create two text files “long_by.h” and “long_by.c”.

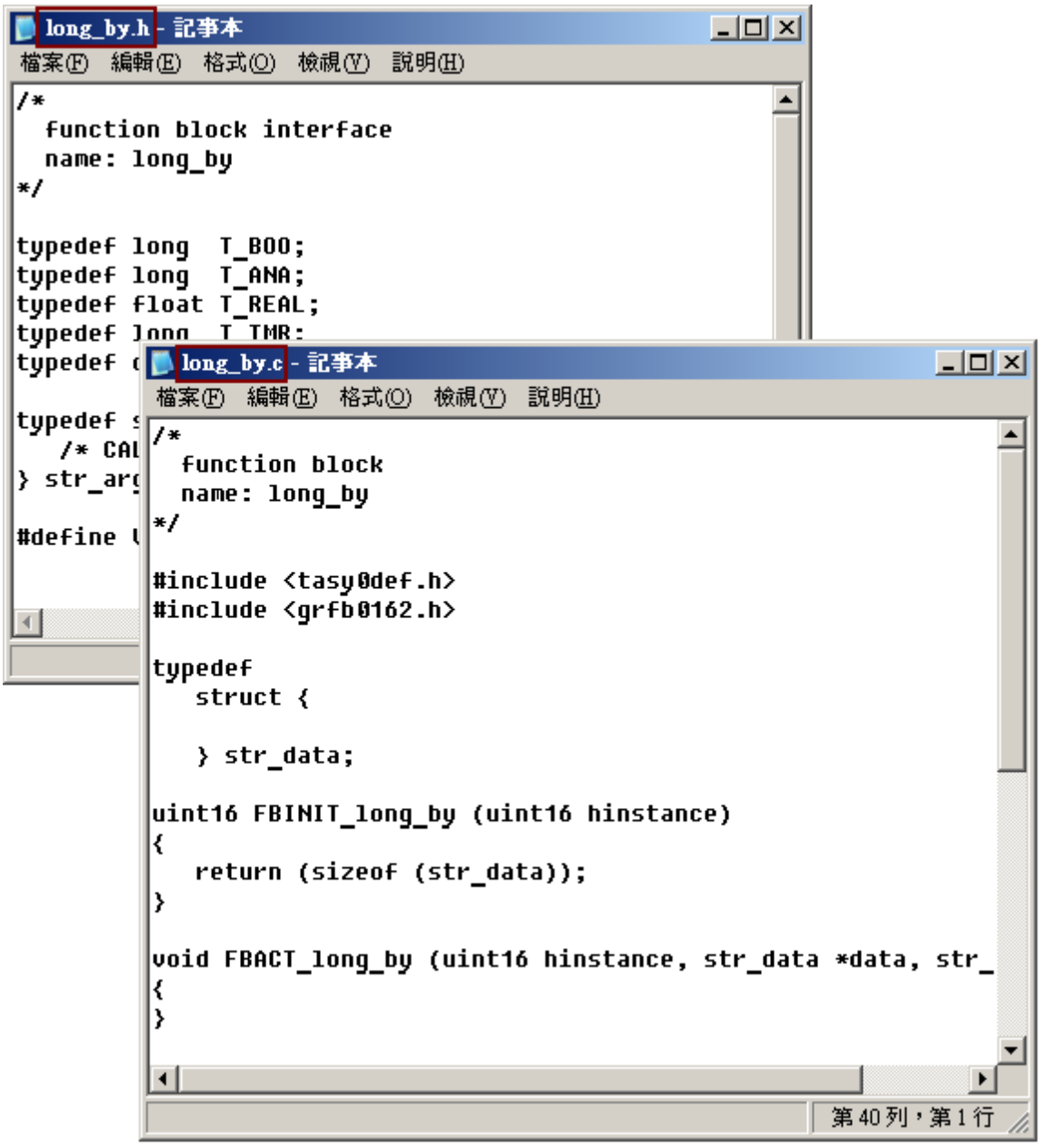
1. In this example, there are four “return” parameters in this “long_by” c-function blocks..





2. Copy the prototype of header files and c-source code to “long_by.h” and “long_by.c”.

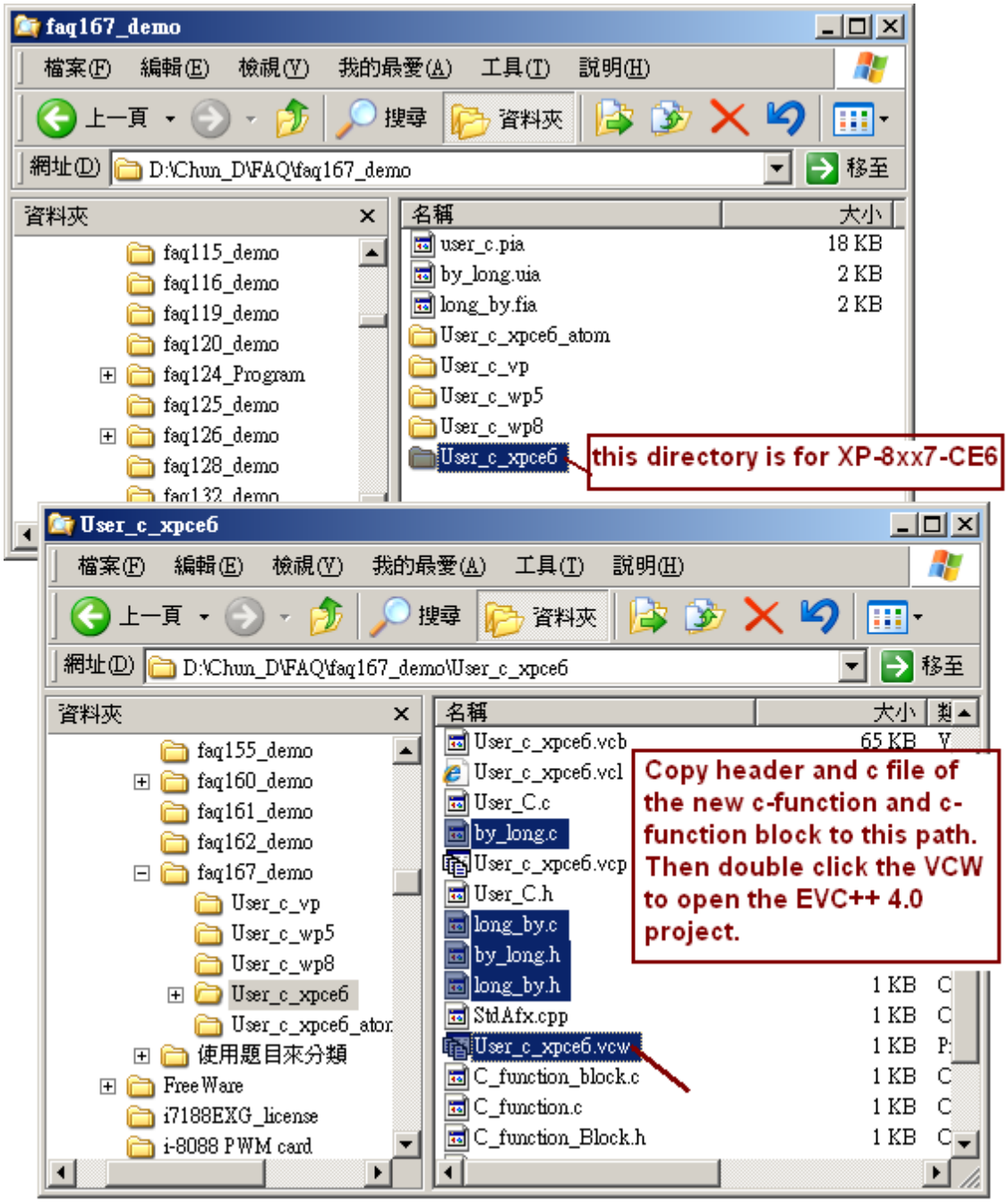
The procedure is similar as the previous section (step 5, 6).



4. Edit the logic of c-function and c-function block

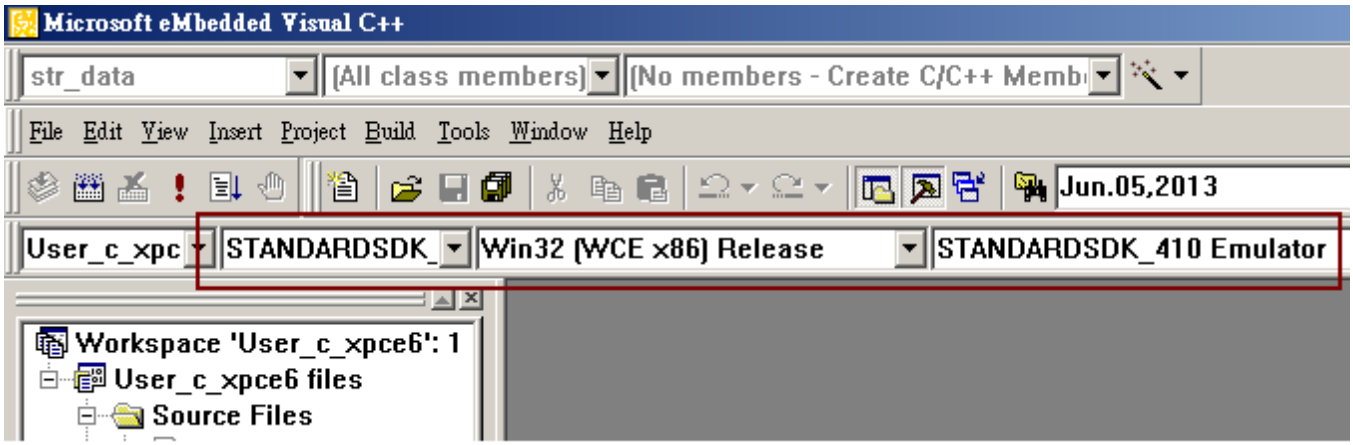
Please make sure your PC has installed the M.S. EVC++ 4.0 before operating this step.

1. First copy the header files and c-source file of the new c-function and c-function block to the EVC++ project path as the following picture. Then mouse double-click the VCW (EVC++ 4.0 project file, for XP-8xx7-CE6 is "User_c_xpce6.vcw") to open this project.

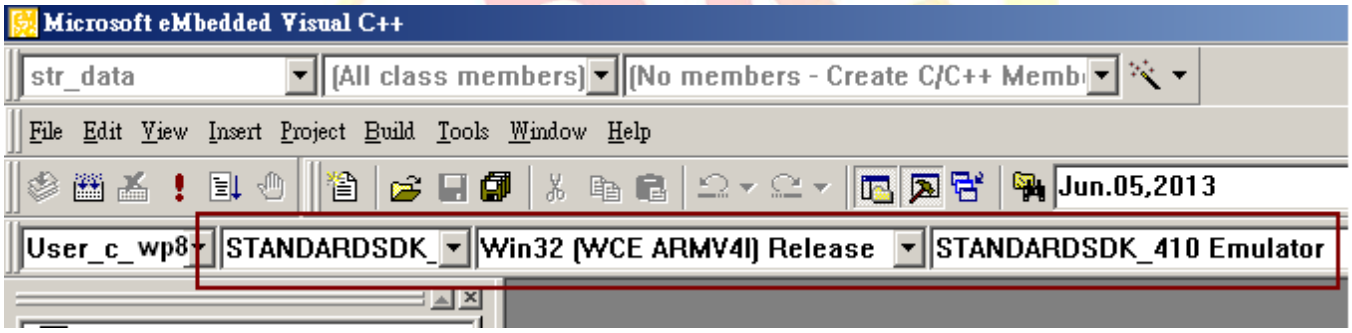


2. Then make sure your EVC++ 4.0 project settings are correct for your PAC. (Note: the settings are different between different controllers.)

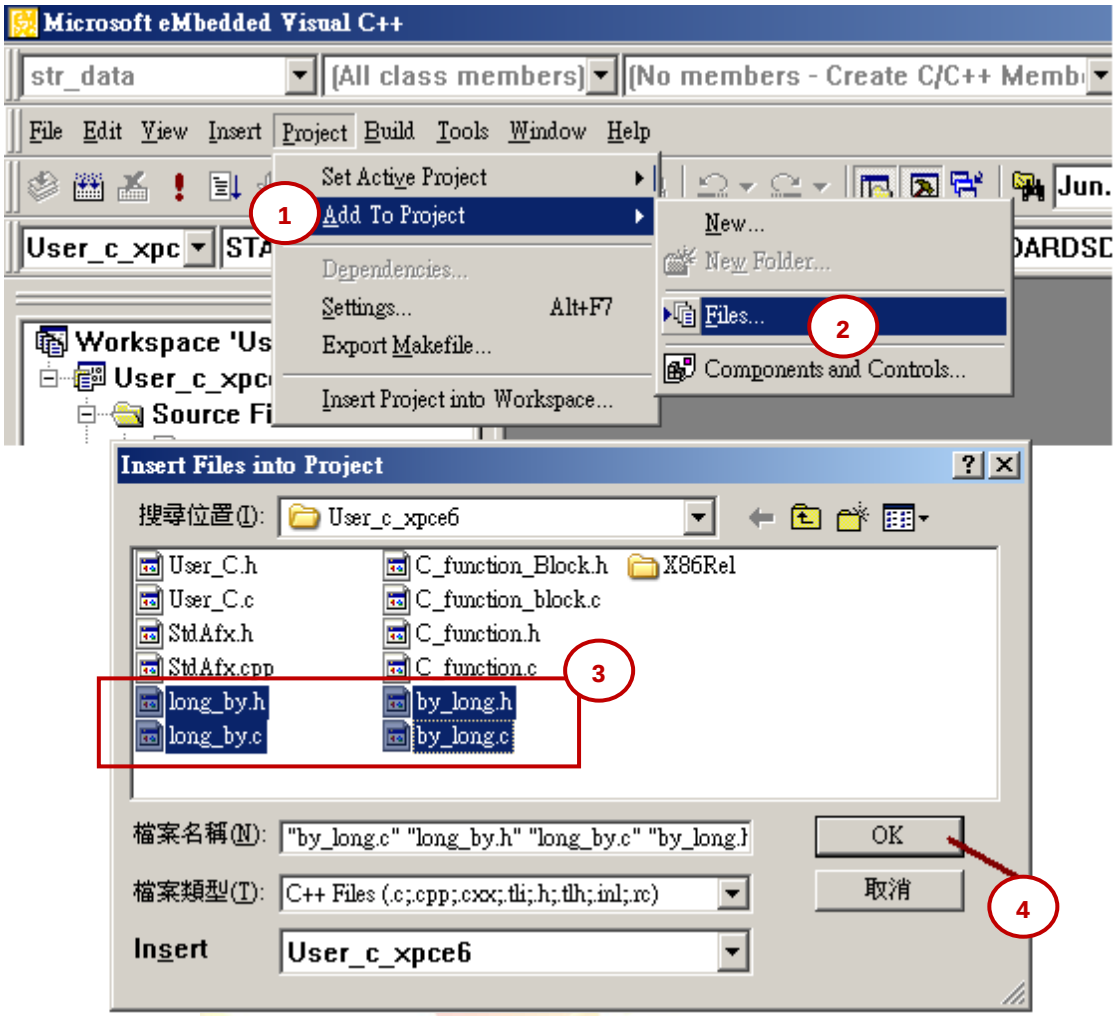
For XP-8xx7-CE6 and XP-8xx7-Atom-CE6: It is
"STANDARDSDK" and "Win32 (WCE **x86**) Release" and "STANDARDSDK_4xx Emulator"



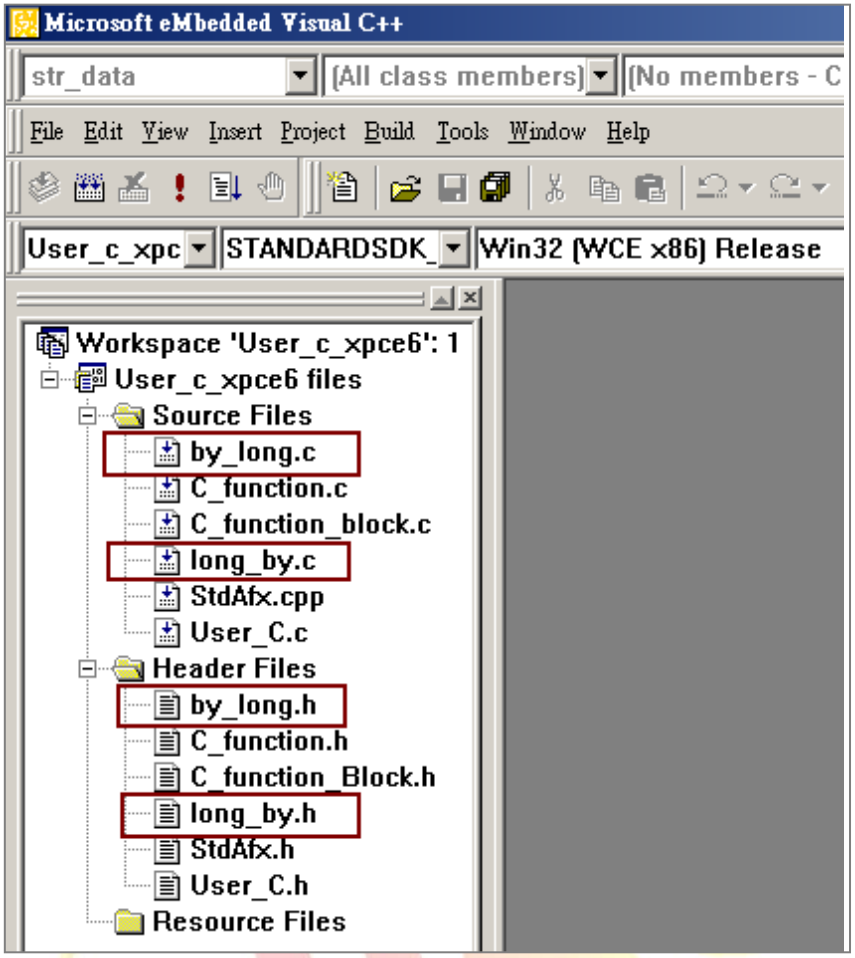
For WP-8xx7 and WP-5147 and VP-25W7 and VP-4137: It is
"STANDARDSDK" and "Win32 (WCE **ARMV4I**) Release" and "STANDARDSDK_4xx Emulator".



Then add c-function and c-function block files to this project.

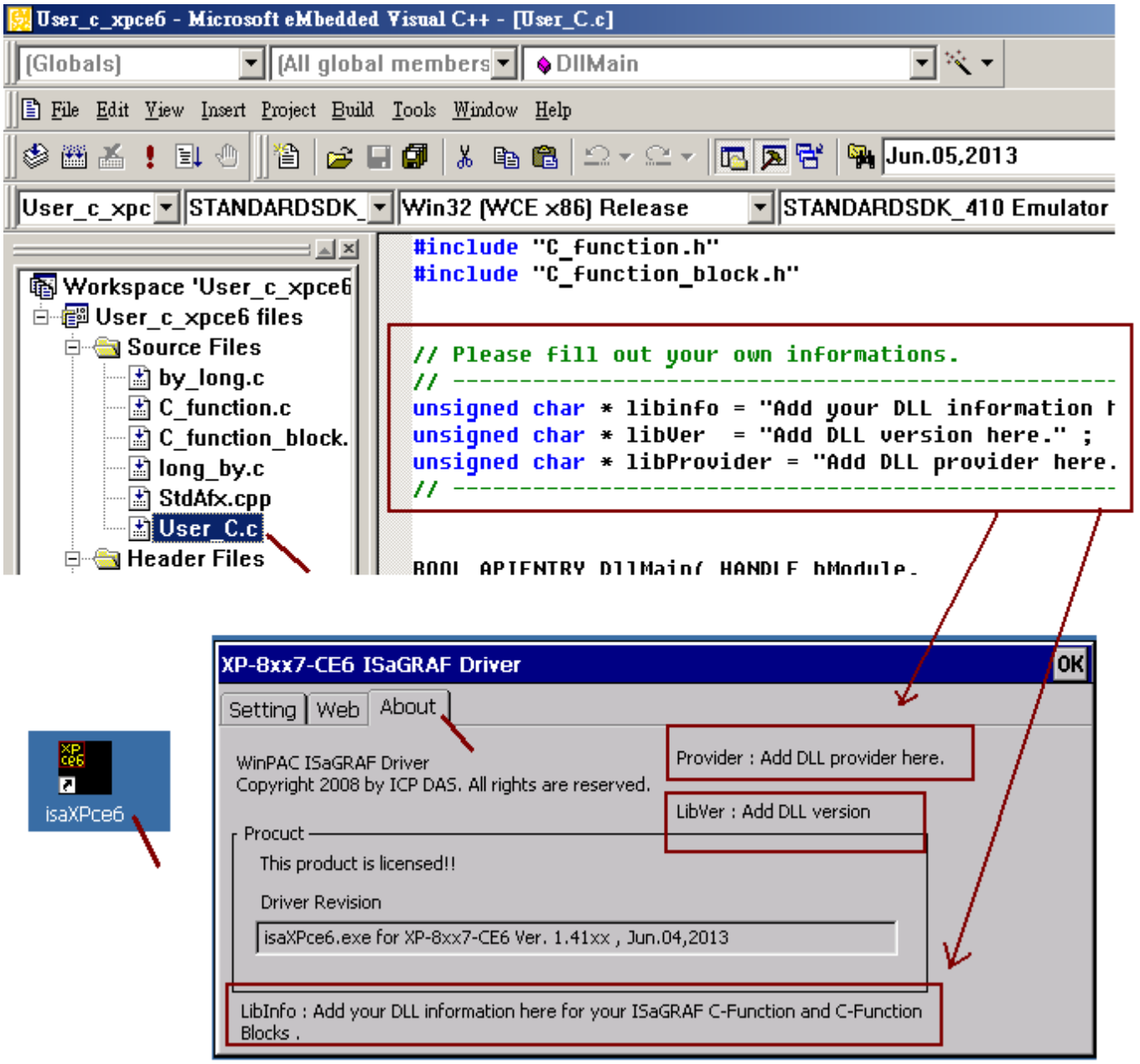


Then you will see the following picture.



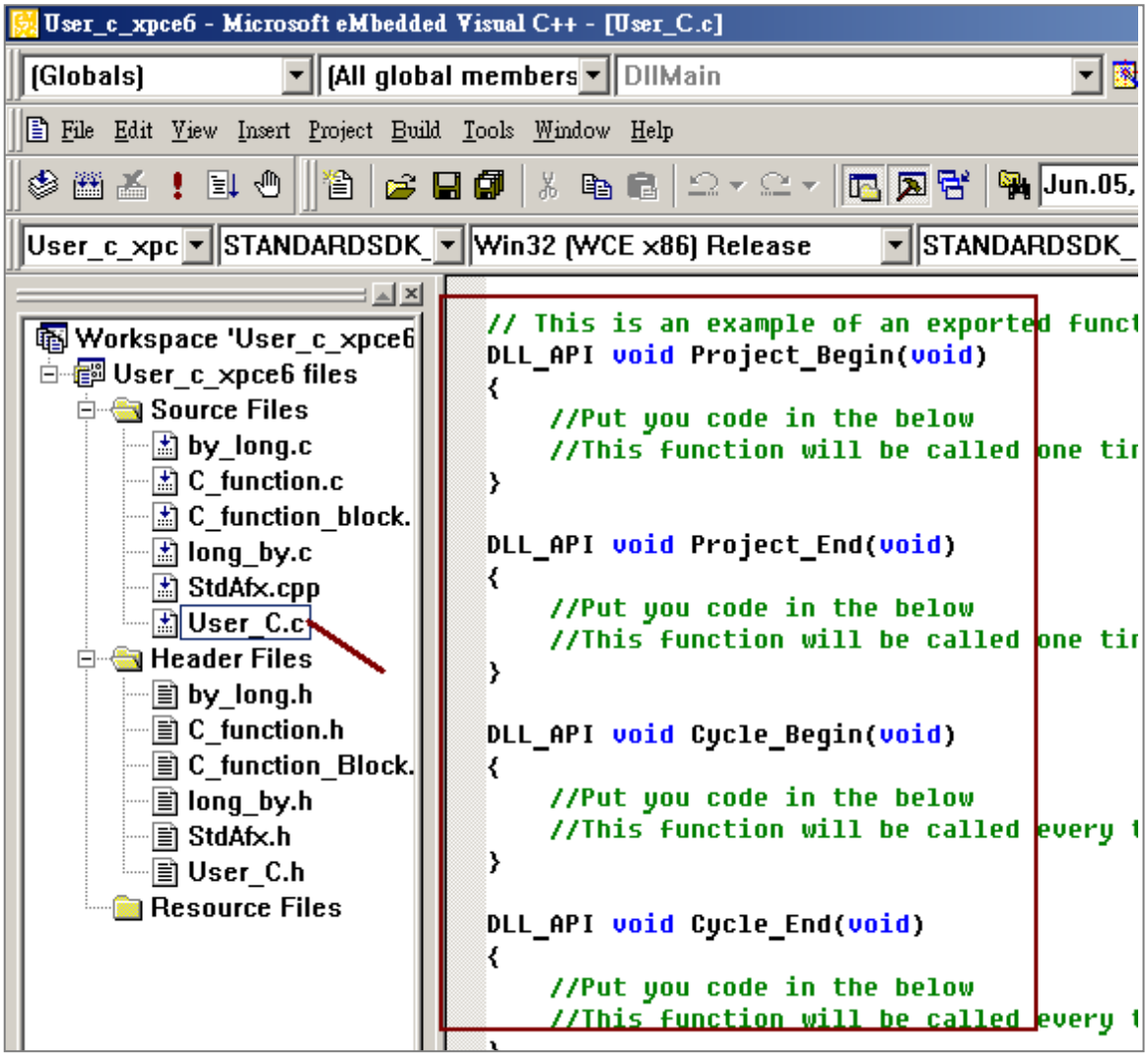
4.1. Edit the “User_C.c”

Please fill out your “libinfo” and “libVer” and “libProvider” information in the “User_C.c”. These three information will show on the ISaGRAF driver dialog 's “About” menu.



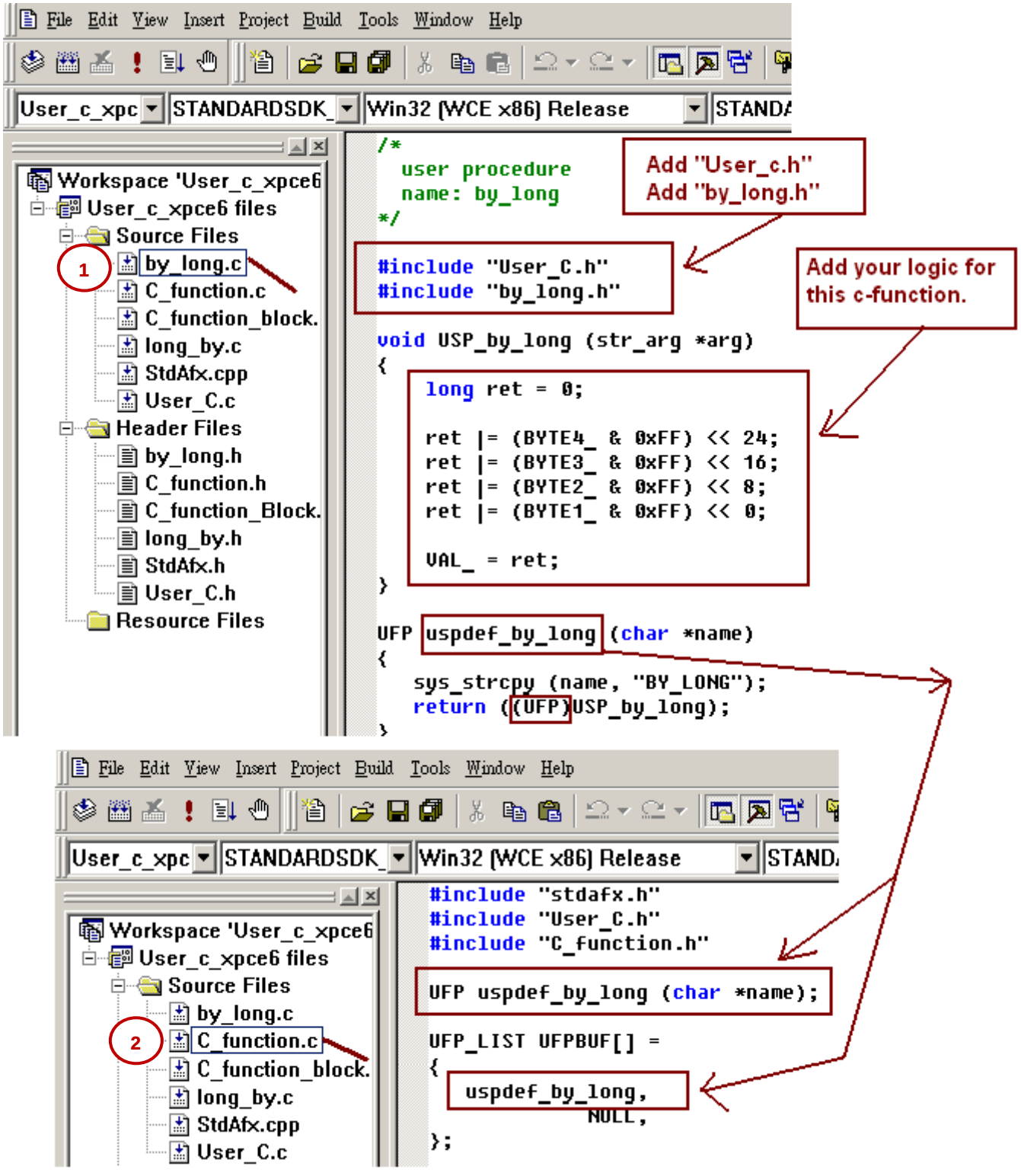
If you have some operations to process when a new ISaGRAF project starts or stops. Please edit the "Project_Begin" and "Project_End" functions in the "User_C.c" file.

If you have some operations need to process before each ISaGRAF cycle or after each cycle, then edit the "Cycle_Begin" and "Cycle_End" functions.



4.2. Edit the C-function file (This example is “by_long.c”)

The definition of input and return parameter is in the header file (this example is “by_long.h”)
Then remember to add this c-function's pointer in the “C_function.c”.



4.3. Edit the C-function block file (This example is “long_by.c”)

The definition of input and return parameters is in the header file (this example is “long_by.h”).

```
#include "User_C.h"
#include "long_by.h"

typedef struct
{
  unsigned char AO[4];
} str_data;

uint16 FBINIT_long_by (uint16 hinstance)
{
   return (sizeof (str_data));
}

void FBACT_long_by (uint16 hinstance, str_data *data, str_arg *arg)
{
  data->AO[0] = (unsigned char)(VAL_ & 0xFF);
  data->AO[1] = (unsigned char)(VAL_>>8 & 0xFF);
  data->AO[2] = (unsigned char)(VAL_>>16 & 0xFF);
  data->AO[3] = (unsigned char)(VAL_>>24 & 0xFF);
}

#define BOO_VALUE   ((T_BOO *)value)
#define ANA_VALUE   ((T_ANA *)value)
#define REAL_VALUE  ((T_REAL *)value)
#define TMR_VALUE   ((T_TMR *)value)
#define MSG_VALUE   ((T_MSG *)value)

void FBREAD_long_by (uint16 hinstance, str_data *data,
      uint16 parno, void *value)
{
    switch(parno)
    {
      case FBLPNO_BYTE1_:
      case FBLPNO_BYTE2_:
      case FBLPNO_BYTE3_:
      case FBLPNO_BYTE4_:
        *ANA_VALUE = (T_ANA)data->AO[parno];
        break;
    }
}

ABP fbldef_long_by (char *name, IBP *initproc, RBP *readproc)
 {
   sys_strcpy (name, "LONG_BY");
   *initproc = (IBP)FBINIT_long_by;
   *readproc = (RBP)FBREAD_long_by;
   return ((ABP)FBACT_long_by);
}
```

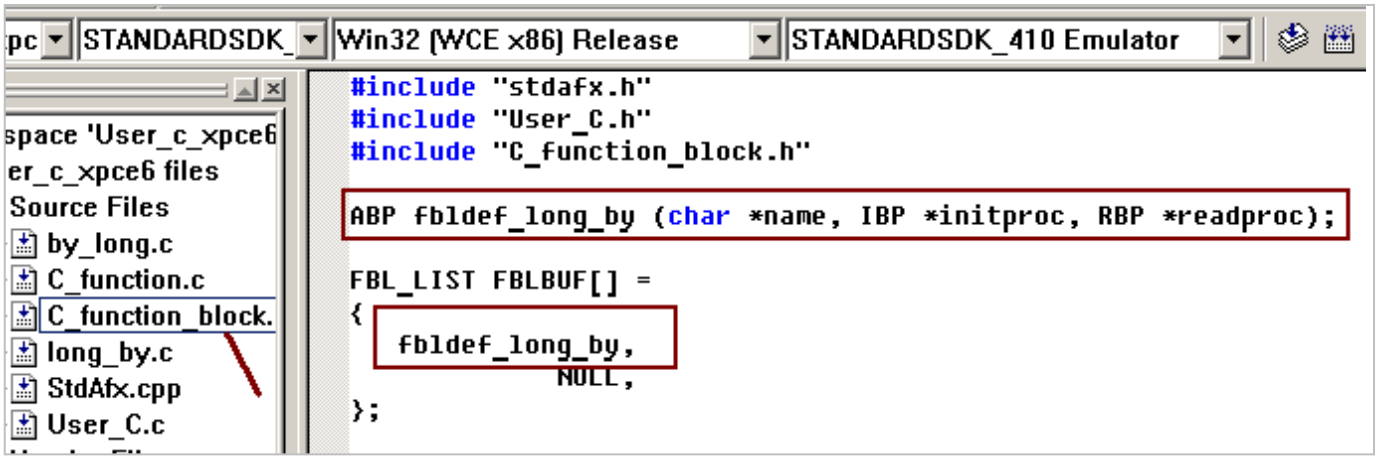
Add "User_C.h"
Add "long_by.h"

Each c-function block instance in the ISaGRAF has its own memory here.

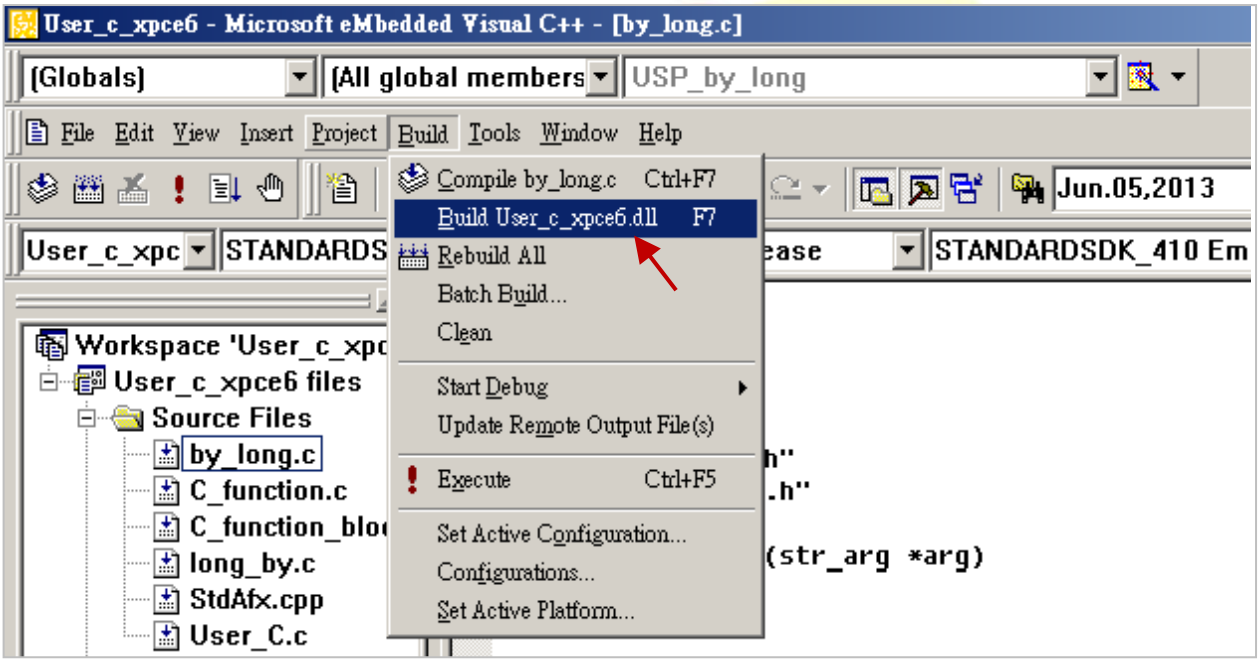
add logic for this c-function block here.

Add code to get return parameters of the c-function block.

1. Remember to add this c-function block's pointer in the “C_function_block.c”.



2. Build the DLL:



3. If the building succeeds, copy the DLL file to the PAC. Then reset the PAC . Then your IsaGRAF project can run your c-function and c-function blocks.

DLL file name for each PAC:

XP-8xx7-CE6	\System_disk\ISaGRAF\User_c_xpce6.dll
WP-8xx7	\System_disk\ISaGRAF\User_c_wp8.dll
WP-5147	\Micro_SD\ISaGRAF\User_c_wp5.dll
VP-25W7 / VP-4137	\System_disk\ISaGRAF\User_c_vp.dll
XP-8xx7-Atom-CE6	\System_disk\ISaGRAF\User_c_xpce6_atom.dll

5. Multi-thread considerations

The ISaGRAF driver for ICP DAS WinCE controllers is a multi-thread process which is built by M.S. EVC++ 4.0 . The major thread lists (not all threads) are at the following table.

The thread priority is set by using the "CeSetThreadPriority" function. Larger priority-value means lower priority. Smaller priority-value means higher priority.

The thread slice value is set by using the "CeSetThreadQuantum" function. The "_tmain" thread handles the cycle logic of the ISaGRAF project. The "sleep" value for the "_tmain" thread is variable (depends on the ISaGRAF cycle scan time of the running project)

Please DO NOT create a thread with a priority value smaller than (or equal to) 113 (means it is a high priority) in the DLL. Or it may crash the project. (Better to set a value larger than 118)
The new created c-function and c-function block are executed in the "_tmain" thread.

Thread Name	Thread Priority WP-8xx7, WP-5147, VP-25W7, XP-8xx7-CE6 (Driver 1.44 or earier version)	Thread Priority XP-8xx7-CE6 (Driver 1.44A or later version)	Slice (ms)	Sleep (ms)
_tmain( )	113	213	X	X
Go_Thread_pwm( )	109	209	1	1
init_SMS( )	116	216	1	50
Go_COM2_write( )	116	216	1	5
Go_COM2_write( )	116	216	1	5
WriteToXML	Above normal	Above normal	1	7
WriteToXML	116	216	1	7
CheckSMemory( )	116	216	1	8
InitEbus	118	218	1	6
Go_COMM	115	215	1	>7
Go_Other_Thread1	124	224	2	7
LP_thread( )	116	216	1	5
T_Go_RDN2( )	118	218	1	25
Thread_Go_RDN( )	118	218	1	60
Go_Thread_I8KE8	117	217	1	x
go_COM1_87K_init	116	216	1	4
Go_Thread_mbus_tcp	117	217	1	5