



HY17P Series Assembly IDE Software User's Manual

Table of Contents

1. -- HY17P IDE OVERVIEW	4
1.1. Introduction	4
1.2. HY17P IDE Installation and System Requirement.....	4
1.3. Installation and Uninstallation	5
1.4. Demo Code import instructions.....	7
1.5. Demo Code Operation	8
2. -- HY17P IDE INTERFACE DESCRIPTION.....	9
2.1. HY17P IDE Edit Interface.....	9
3. -- HY17P IDE DEBUG INTERFACE	19
3.1. Fast Execution	20
3.2. RAM Window	23
3.3. Register Window	26
3.4. Watch Window	28
3.5. Stack Window	30
3.6. Register & SRAM Revise Record	31
3.7. Hint Function of Source Code Window	33
4. -- PROGRAMMING WINDOW.....	35
4.1. Interface setting.....	35
4.2. Operation Procedures	39
4.3. PC Online OTP Programming.....	41
4.4. PC Offline Programming	45
5. -- TROUBLESHOOTING	48
5.1. HYCON-IDE Execution Problem.....	48
6. -- REVISION HISTORY.....	49

Attention :

1. HYCON Technology Corp. reserves the right to change the content of this datasheet without further notice. For most up-to-date information, please constantly visit our website: <http://www.hycontek.com> .
2. HYCON Technology Corp. is not responsible for problems caused by figures or application circuits narrated herein whose related industrial properties belong to third parties.
3. Specifications of any HYCON Technology Corp. products detailed or contained herein stipulate the performance, characteristics, and functions of the specified products in the independent state. We does not guarantee of the performance, characteristics, and functions of the specified products as placed in the customer's products or equipment. Constant and sufficient verification and evaluation is highly advised.
4. Please note the operating conditions of input voltage, output voltage and load current and ensure the IC internal power consumption does not exceed that of package tolerance. HYCON Technology Corp. assumes no responsibility for equipment failures that resulted from using products at values that exceed, even momentarily, rated values listed in products specifications of HYCON products specified herein.
5. Notwithstanding this product has built-in ESD protection circuit, please do not exert excessive static electricity to protection circuit.
6. Products specified or contained herein cannot be employed in applications which require extremely high levels of reliability, such as device or equipment affecting the human body, health/medical equipments, security systems, or any apparatus installed in aircrafts and other vehicles.
7. Despite the fact that HYCON Technology Corp. endeavors to enhance product quality as well as reliability in every possible way, failure or malfunction of semiconductor products may happen. Hence, users are strongly recommended to comply with safety design including redundancy and fire-precaution equipments to prevent any accidents and fires that may follow.
8. Use of the information described herein for other purposes and/or reproduction or copying without the permission of HYCON Technology Corp. is strictly prohibited.

1. HY17P IDE Overview

1.1. Introduction

To facilitate the product development process, the HYCON-IDE development environment is used to deploy HYCON's full range of MCUs. The customer can realize the online of the final product, carry on the simulation on this platform, and program the code to the OTP product of HY 17P series.

1.2. HY17P IDE Installation and System Requirement

The minimum requirement of system configuration to operate HYCON-IDE:

- PC/NB Hardware requirement:
PC compatible machine with PENTIUM® CPU
512 MB Memory (1GB is recommended)
10 GB Hard Disk Space

- Supported Products:
 - HY17P48
 - HY17P51
 - HY17P52
 - HY17P55
 - HY17P56
 - HY17P58
 - HY17P60 & HY17P60B
 - HY17P68

- Supported Hardware Model No:
 - HY17S58-DK02 : HY17S58 IDE hardware (development kit)
 - HY17S68-DK02 : HY17S68 IDE hardware (development kit)
 - HY17S68-DK03 : HY17S68 IDE hardware (For DMM development kit)

- Supported Software version:
HY17P IDE V1.2 or above : HY17P Series Assembly IDE software


- Supported Operating System:
Windows XP, Windows Vista, Windows 7, Windows 8, Windows 10

- Apply the following Interface Modes:
USB Port with HID-compliant device

1.3. Installation and Uninstallation

1.3.1. Installation

With some Windows operating systems, administrator access permissions are required to install software on the computer.

- Find and execute in CD or file  setup executable file
- Follow the instructions on the screen to perform the installation step by step, as shown in Figure 1.

HY17P Series Assembly IDE Software User's Manual

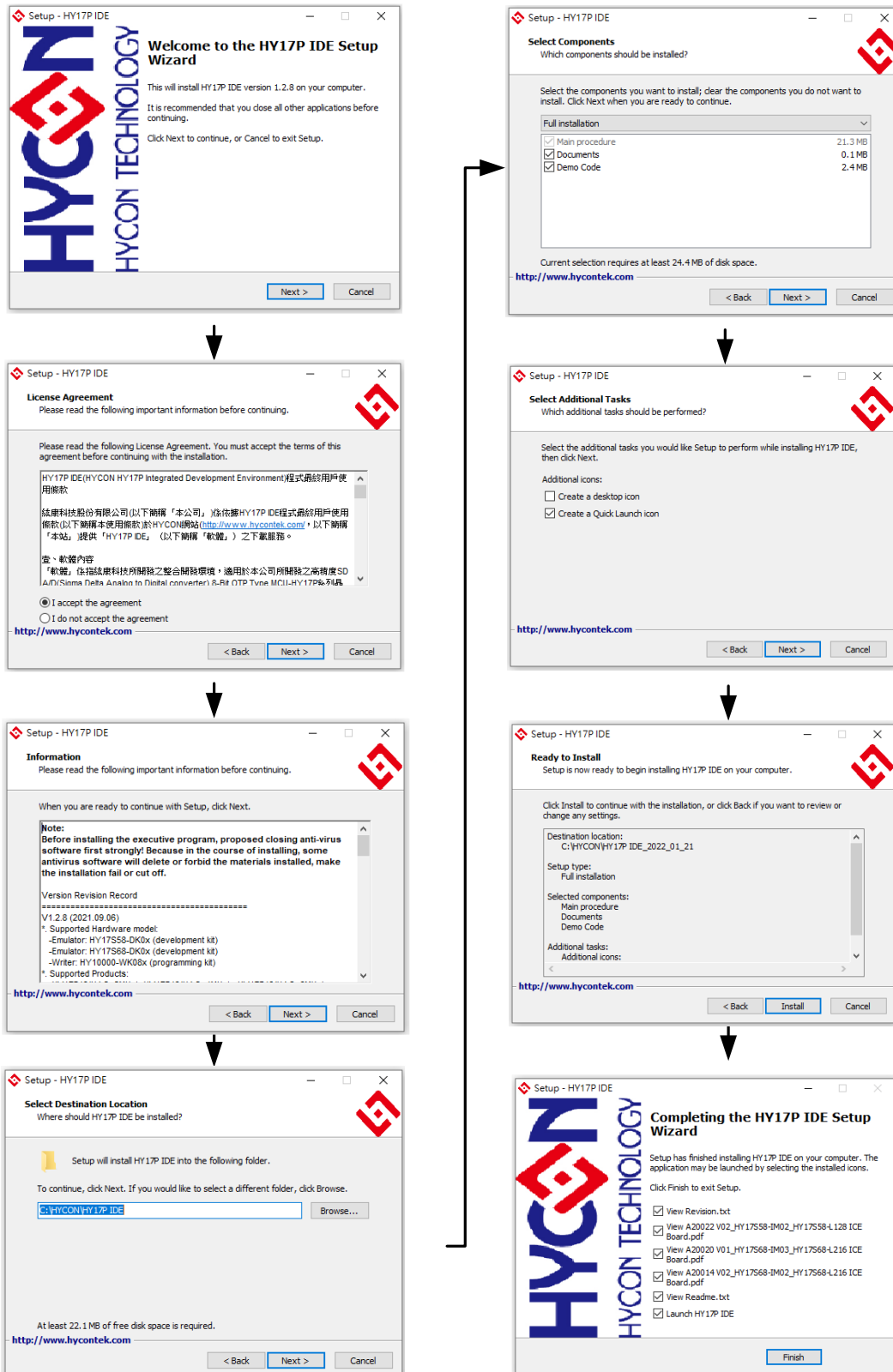


Figure 1

1.3.2. Uninstallation

Please remove the file “HY17P IDE” in “Add/Remove” under Control Panel.

1.4. Demo Code import instructions

- Open C:\HYCON\HY17P IDE\DemoCode
- Set the file as assembly main file
- Assembly start to progress program debug

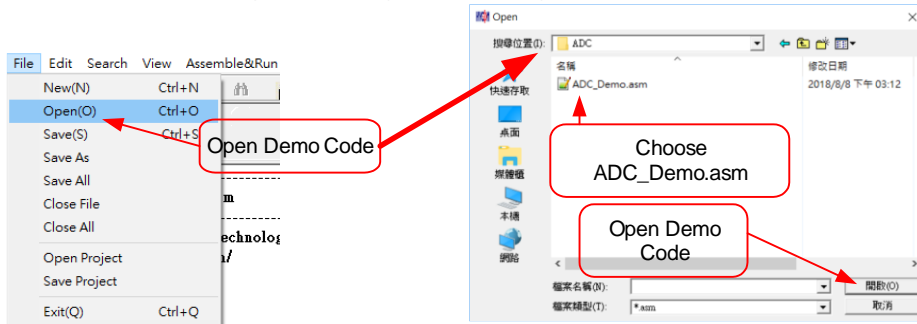


Figure 2

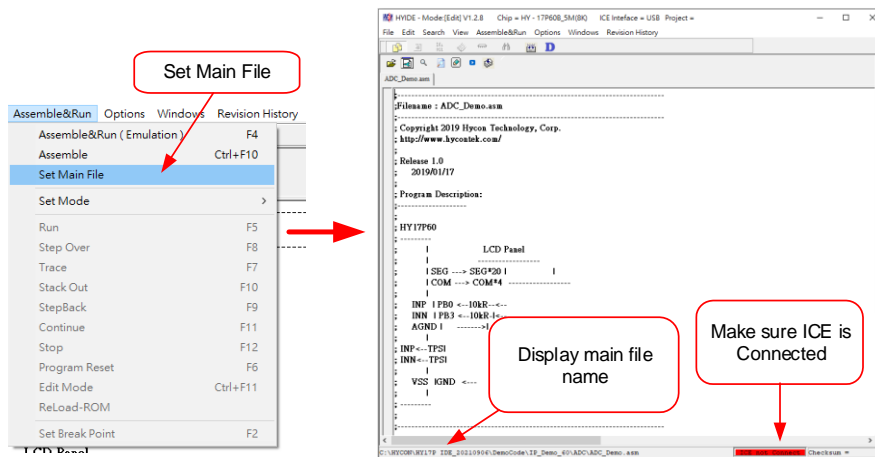


Figure 3

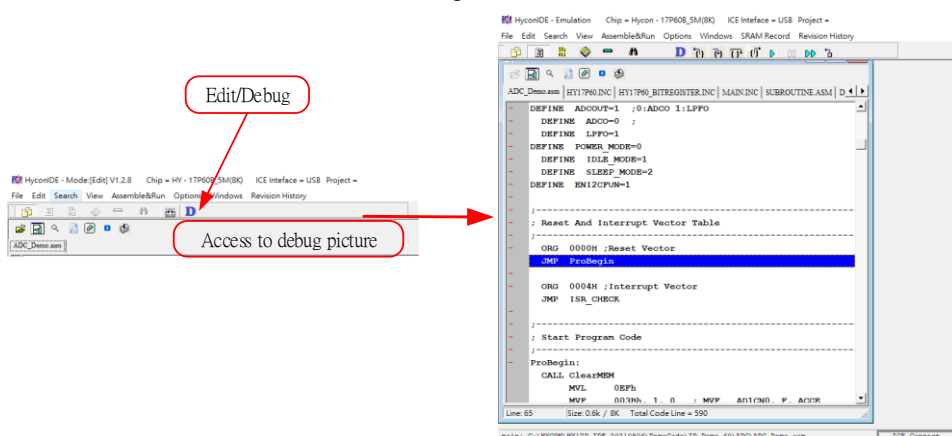


Figure 4

- Any editor can be used to edit Source Code, as long as it can be stored as ASCII Code format. When program assemble, the Source Code will be downloaded again to ensure the program works correctly. Debug and edit function will be elaborated in next Chapter.

1.5. Demo Code Operation

- After executing HY17P IDE software installation, Demo Code will be provided under the main program of C:\HYCON\HY17P IDE\DemoCode for users' reference.

2. HY17P IDE Interface Description

2.1. HY17P IDE Edit Interface

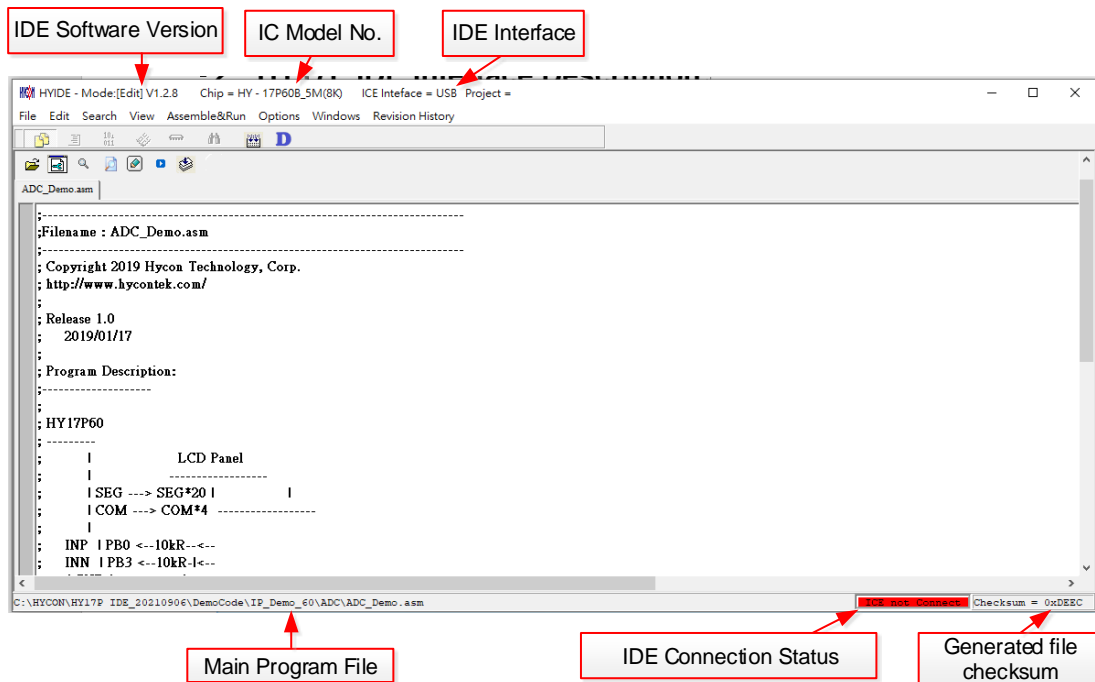







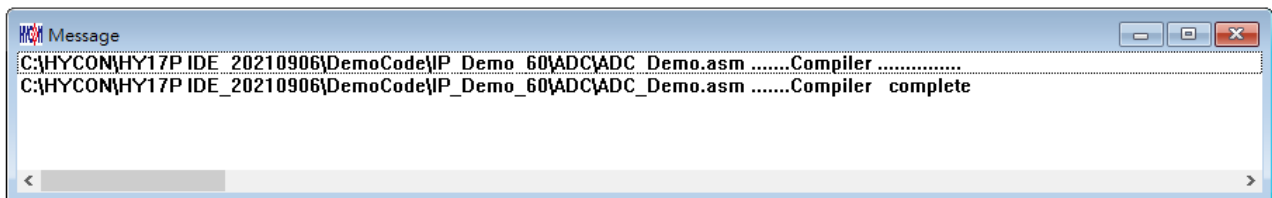


Figure 5

2.1.1. Edit Window

- Open 
Open an existing file
- Label Setting 
Set label. When open too many files, this icon helps to return to the label setting place quickly.
- Jump to Label 
Jump to where to label has been placed.
- Find string 
Search the entered word.
- Search the specified word 
Search the specified word.
- Switch Display Window 
When too many files are opened, this icon helps to command file switching.
- Assembly 
Only assembly is executed. It will not get into program debug status. The information dialog may pop up after assembly accomplished.



2.1.2. File Menu

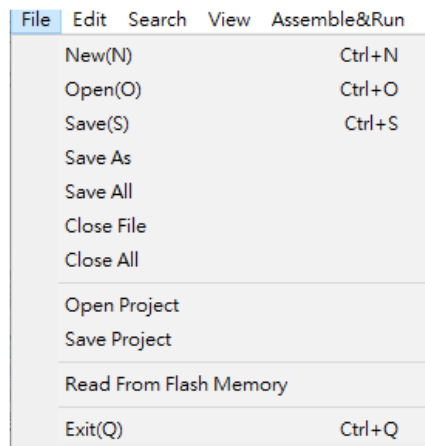


Figure 6

- New → Create a new file.
- Open → Open an existing file.
- Save → Write the active window data to the active file.
- Save As → Write the active window data to the specified file.
- Save All → Write all windows data to the corresponding opened files.
- Open Project → Project includes IC part no., IDE interface, Assembly main filename, Active opened status, and Checksum. Project status will be loaded in after the project is opened.
- Save Project → Write the active project to the active project file.
- Exit → Close the current active HYCON-IDE program.

2.1.3. Edit Menu

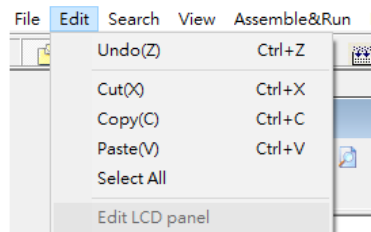


Figure 7

- Undo → Cancel the previous editing operation.
- Cut → Remove the selected lines from the file.
- Copy → Place a copy of the selected lines.
- Paste → Paste the copy lines to the present insertion point.
- Select All → Select all lines in the active file.

2.1.4. View Menu

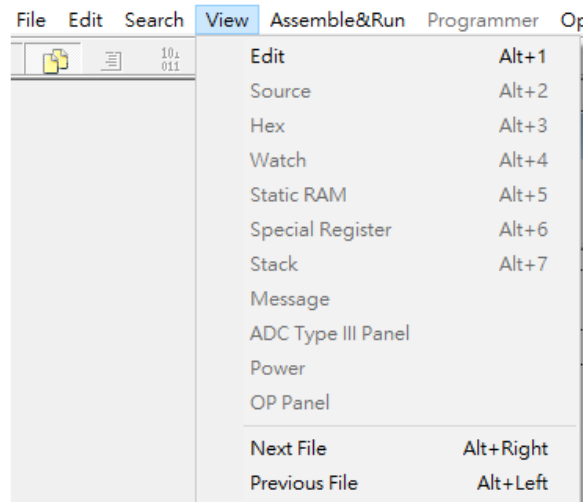


Figure 8

- Edit → Appoint the edit window as the present active window.
- Next File → Appoint the next file as the present active window.
- Previous File → Appoint the previous file as the present active window.

2.1.5. Assemble & Run Menu

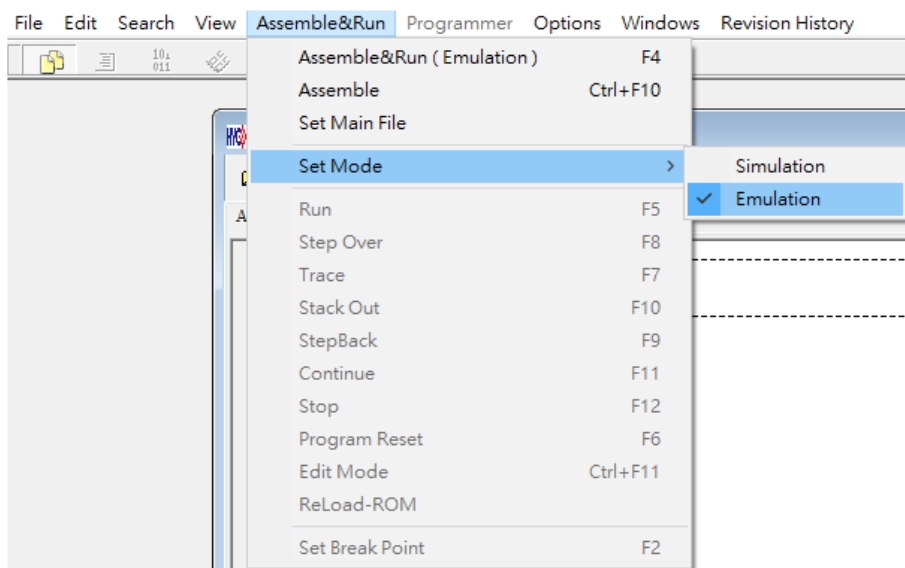


Figure 9

- Assemble & Run (Emulation) → Assemble Source Code and execute program debug mode.
- Assemble → Only execute the program assembly, program debug is not executed. This assembler will not generate error message according to IC part no. Error message will show up when the lines is error. It is usually used in generating OBJ Code (Object).
- Set Main File → Set the file as assembly main file. Files will be named after compiler generated file name, such as Hex, MAP, ASC...etc.
- Set Mode → Debug through software or hardware is selective.

2.1.6. Options Menu

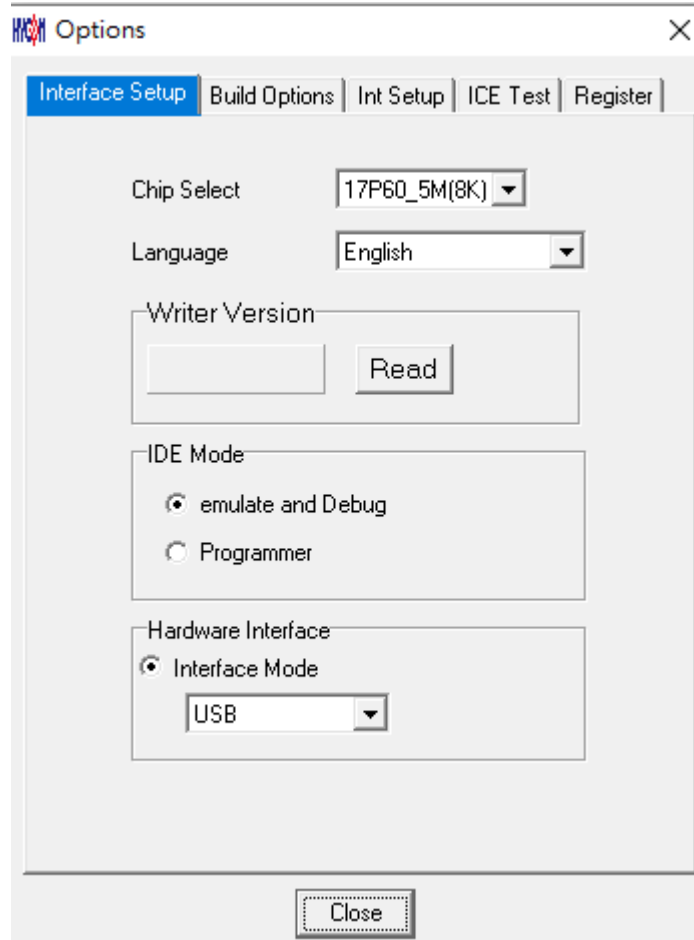


Figure 10

- Interface Setup

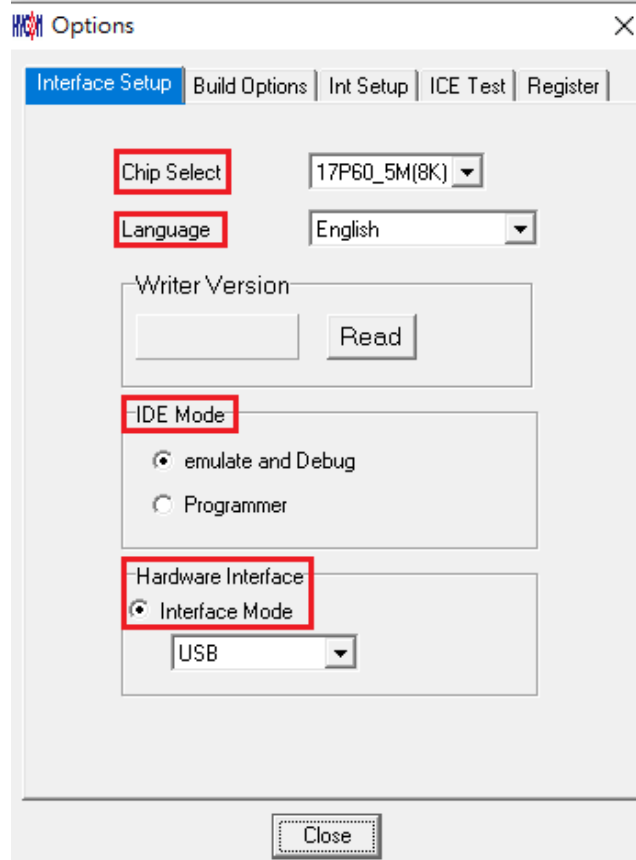


Figure 11

- Chip Select: Select IC part no. Compiler will assemble the selected part no.'s program file. It will determine whether there is any misuse or non-existing Register or SRAM, or has the program exceeded the ROM Size.
- Language option: English and Chinese interface are selectable.
- Mode option: Two choices, Emulate and debug and program.
- Hardware Interface: Select USB interface.

- **Build Options**

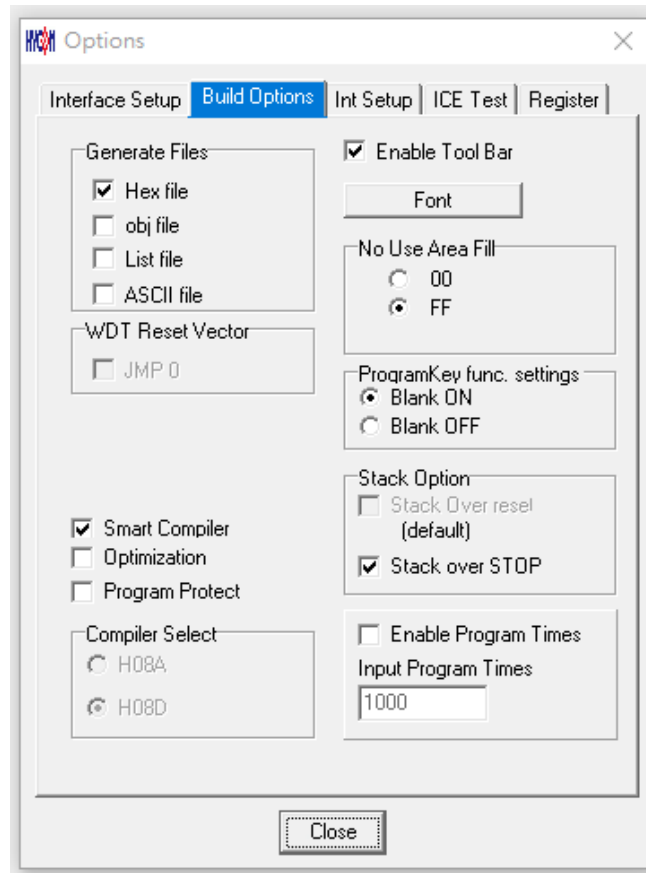


Figure 12

- Assembler generated extension: it is selectable to produce below file format
 - Binary file: Hex
 - Obj file : obj
 - List file : lst
 - ASCII file : asc
- Stack operation: Choose to replace the program after stack overflow. When this option is chosen, Compiler will add to Hex, it will be programmed in to OTP.
- Program number of times limit: Please refer to “Interface Setup” under chapter 4. Programming Window.
- Font option: Choose editor’s fonts.
- Fill unused zone: Fill the unused zone with 0x0000 or 0xFFFF in the program.
- Simplified assemble: Simplified assemble function is selectable. When JMP or CALL is smaller than 2K, it will automatically transform to RJ or RCALL. If the arguments of CALL are set, it will not transform to RCALL.
- Program protection: Please refer to “4.1 Interface Setup” under chapter 4. Programming Window

- **Interrupt Setting**

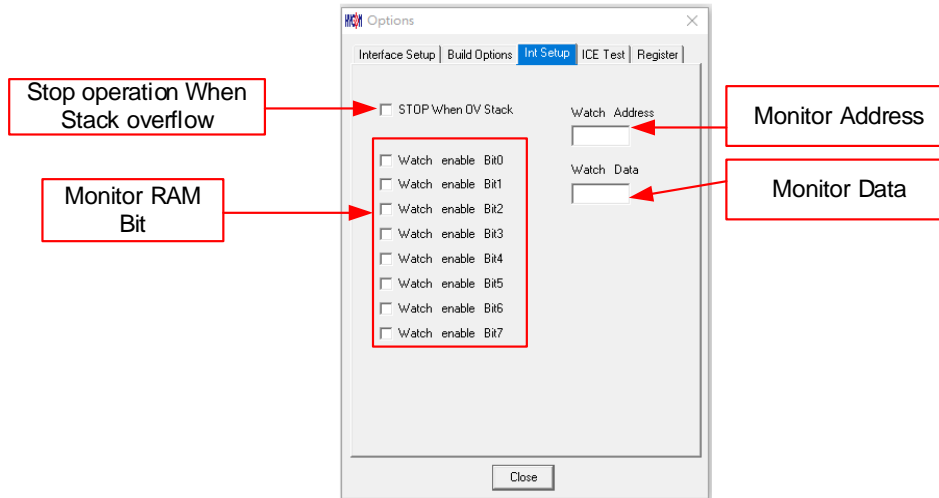


Figure 13

- Stop operation when Stack overflow: IDE will stop when Stack overflow.
- Monitor address: Select the monitored Register or RAM. The program will stop when the program executed RAM or Register value equals to the monitored Data.
- Monitor Data: Monitor value is set when the monitor Data is filled up.
- Monitor RAM bit: Monitor function will be activated if the monitor bit is marked on. The program will stop when the bit of Data value equals to the marked on bit.

- **ICE Test**

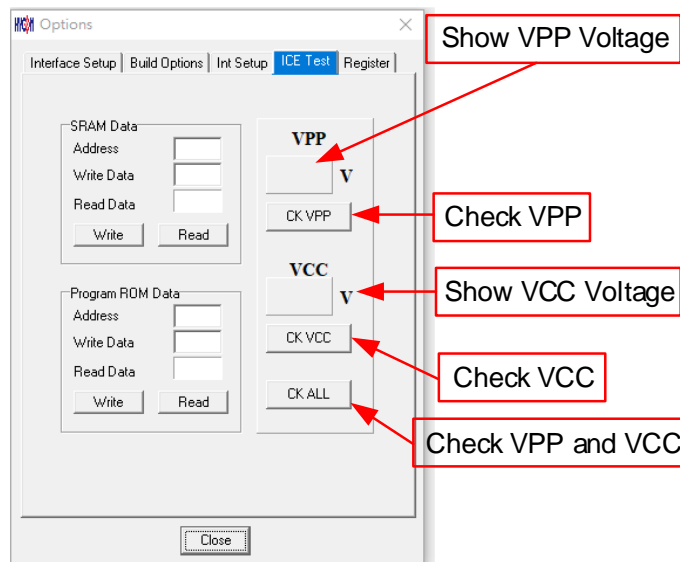


Figure 14

- OSC calibration

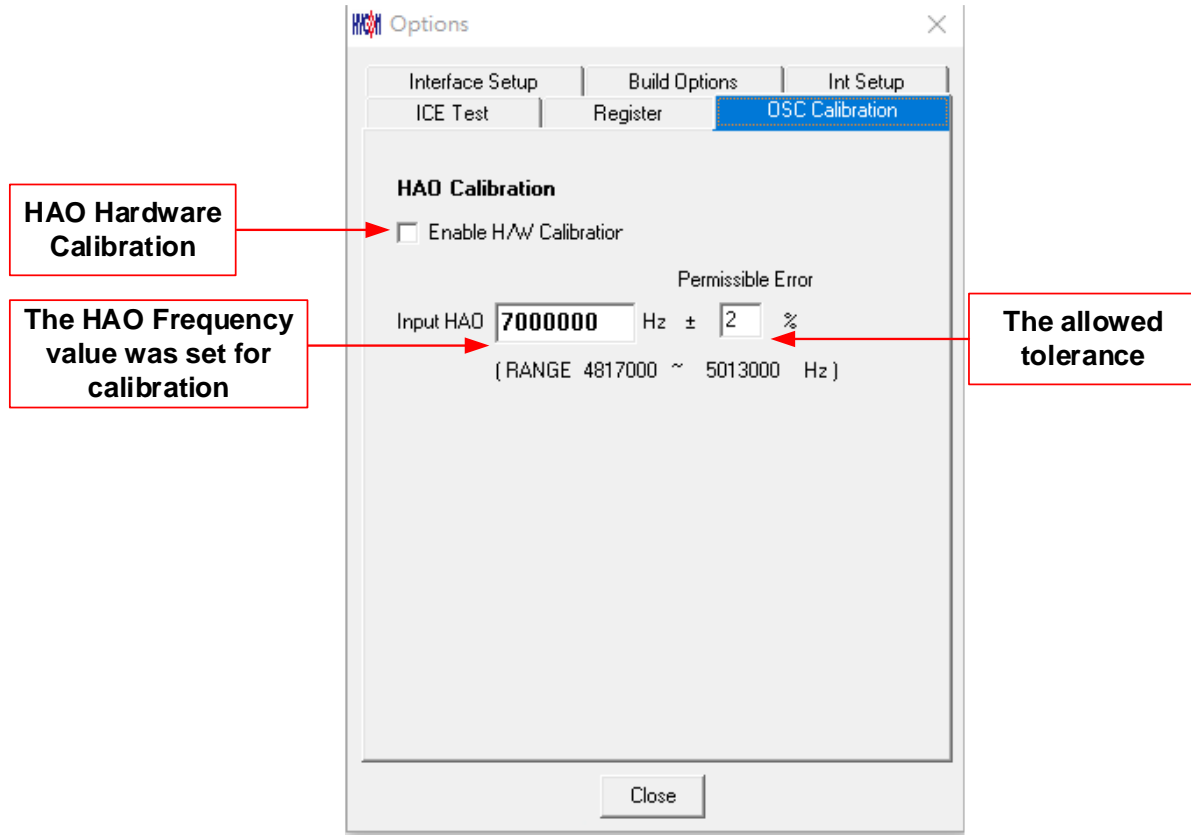


Figure 15

2.1.7. Window

The window can be displayed horizontally or vertically

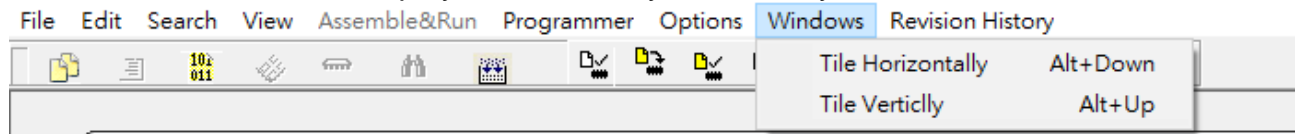


Figure 16

2.1.8. Program Structure

Before editing new program, user must select IC part number through interface setup; Different IC will have different Instruction Set, according to IC part number definition; it is classified as H08A and H08B instruction set; User can refer to the appendix software demo code, Chapter 1.6 gives illustration of demo code usage. Users can refer to following program structure to start writing program. Basic structure description is listed as below:

- Program Name Definition as: *****.ASM**
- Register Name or RAM Definition as: *****.INC**
- Many program contents are listed below:
 - "Main.asm" 、 "Initial.asm" 、 "Interrupt.asm" 、 "Sub.asm" 、 "Mian.inc" 、 "H08.inc"

- "Main.asm" structure:

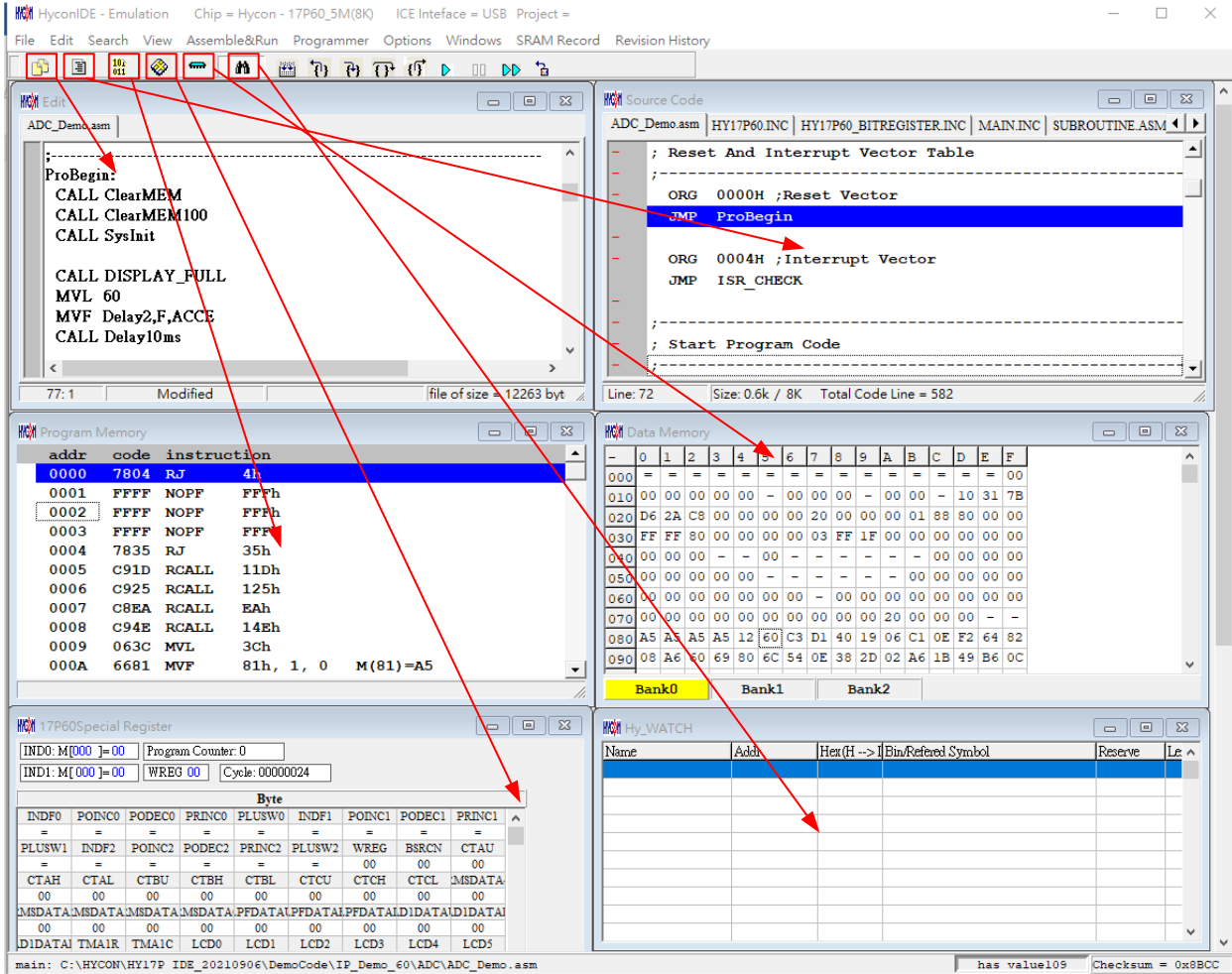
	<code>;Program name can be any name</code>	
<code>Include 17P.inc</code>		<code>;HY17P series special register name and address definition</code>
<code>Include Main.inc</code>		<code>;RAM name, address definition</code>
<code>ORG 00H</code>		<code>;Declare program start</code>
<code>JMP BEGIN</code>		<code>;Jump to main program</code>
<code>ORG 04H</code>		<code>;Declare interrupt flag address</code>
<code>Include Interrupt.asm</code>		<code>;Refer to "Interrupt.asm" interrupt subroutine</code>
		<code>;Include file max. 100</code>
<code>BEGIN:</code>		<code>;Start Main program. Label name definition can be any word</code>
<code>Include Initial.asm</code>		<code>;Reference "Initial.asm" hardware initialization subroutine</code>
<code>JMP T1</code>		<code>; Jump to T1 subroutine</code>
<code>...</code>		
<code>T1:</code>		
<code>NOP</code>		
<code>Include Sub.asm</code>		<code>;Referencing the "Sub.asm" subroutine</code>
<code>END</code>		<code>;Program end</code>

- Reference Document :
 - IP User Manual :
 - Instruction Set User Manual: H08A Instruction Set Manual or H08B Instruction Set Manual
 - HYCON-IDE Compiler User Manual: [HY-MCU COMPILER](#)

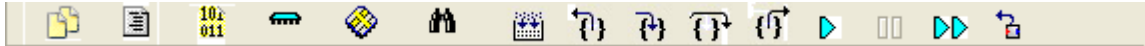
3. HY17P IDE Debug Interface

It can be classified into hardware debug and software debug.

- Hardware debug
The indication column is blue
- Software debug
The indication column is green



3.1. Fast Execution



● Fast Window Switch

(1) Switch to Edit Window

```

ProBegin:
CALL ClearMEM
CALL ClearMEM100
CALL SysInit

CALL DISPLAY_FULL
MVL 60
MVF Delay2,F,ACCE
CALL Delay10ms
            
```

(2) Switch to Source window

```

; Reset And Interrupt Vector Table
;
ORG 0000H ;Reset Vector
JMP ProBegin

ORG 0004H ;Interrupt Vector
JMP ISR_CHECK

; Start Program Code
            
```

(3) Switch to Hex window

addr	code	instruction
0000	7804	RJ 4h
0001	FFFF	NOFF FFFh
0002	FFFF	NOFF FFFh
0003	FFFF	NOFF FFFh
0004	7835	RJ 35h
0005	C91D	RCALL 11Dh
0006	C925	RCALL 125h
0007	C8EA	RCALL EAh
0008	C94E	RCALL 14Eh
0009	063C	MVL 3Ch
000A	6681	MVF 81h, 1, 0 M(81)=A5

(4) Switch to Ram window

Addr	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
020	D6	2A	C8	00	00	00	00	20	00	00	00	01	88	80	00	00
030	FF	FF	80	00	00	00	00	03	FF	1F	00	00	00	00	00	00
040	00	00	00	-	-	-	-	-	-	-	-	-	00	00	00	00
050	00	00	00	00	00	00	00	-	-	-	-	00	00	00	00	00
060	00	00	00	00	00	00	00	-	00	00	00	00	00	00	00	00
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	-	-
080	A5	A5	A5	A5	12	60	C3	D1	40	19	06	C1	0E	F2	64	82
090	08	A6	60	69	80	6C	54	0E	38	2D	02	A6	1B	49	B6	0C










(5) Switch to Reg window

Byte	INDF0	POINCO	PODEC0	PRINC0	PLUSW0	INDF1	POINC1	PODEC1	PRINC1
=	=	=	=	=	=	=	=	=	=
PLUSW1	INDF2	POINC2	PODEC2	PRINC2	PLUSW2	WREG	B SRCN	CTAU	
=	=	=	=	=	=	00	00	00	
CTAH	CTAL	CTBU	CTBH	CTBL	CTCU	CTCH	CTCL	MSDATA	
00	00	00	00	00	00	00	00	00	
MSDATA	MSDATA	MSDATA	MSDATA	MSDATA	MSDATA	MSDATA	MSDATA	MSDATA	
00	00	00	00	00	00	00	00	00	
DIDATA	TMAIR	TMAIC	LCD0	LCD1	LCD2	LCD3	LCD4	LCD5	

(6) Switch to Watch window

Name	Addr	Hex(H -> Bin/Refered Symbol	Reserve	Le ^

- **Fast Debug**

- (1) Step back 
- (2) Trace(Enter into Macro/subroutine) 
- (3) Step over(Not enter into Macro/subroutine) 
- (4) Skip Call 
- (5) Execute(Free RUN) 
- (6) Pause 
- (7) Continue 
- (8) Program replace 
- (9) Back to edit mode 

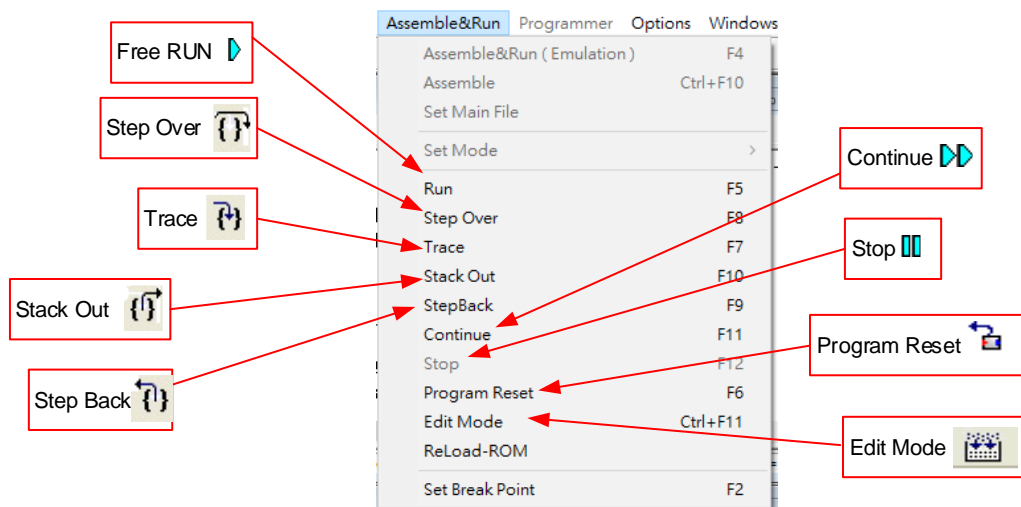


Figure 17

- Two methods to set or remove interrupt:

1. Use mouse to select interrupt place in program code window or machine code window, press "F2" button to set to remove interrupt.
2. Use mouse to select interrupt place in program code window or machine code window, double click the left key to set or remove interrupt.

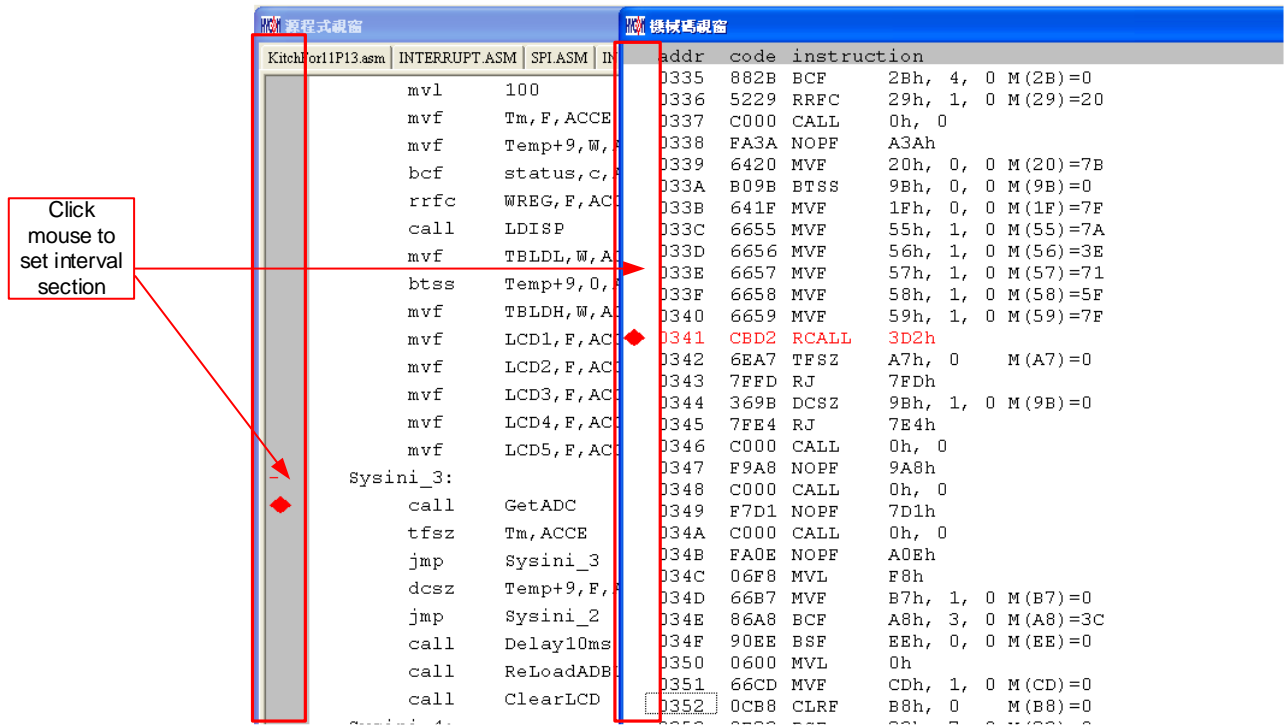


Figure 18

3.2. RAM Window

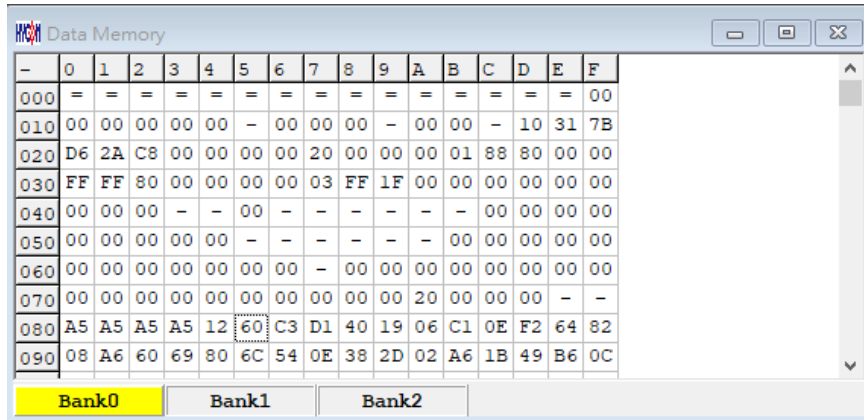


Figure 19

- After opening the RAM window, the Bank will display the corresponding RAM window according to the IC model, each bank has 256 bytes.
 - Bank0 starts from 0x00 to 0xFF. Bank1 starts from 0x100 to 0x1FF...etc.
 - If the address does not exist, it will display " - ".
 - If users intend to switch Bank display, use cursor to point to the desired Bank zone, and then click the left key of the mouse to confirm.
 - If Hint is set, the address will display numbers and will be underlined.
 - **Notice: The Address 0x00 ~ 0x0E of Bank0 is indirect addressing register, it cannot be revised directly, the displayed value is not referable. If revise is required, please refer to Chapter 3.3.** Revise indirect addressing Data or Address.
- **Function Display**
Click the mouse selection key (right key)

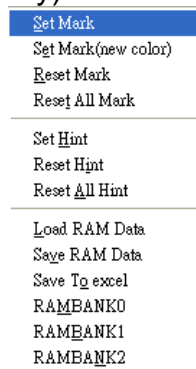


Figure 20

- Set Mark
- Set Mark (new color)
- Reset Mark
- Reset All Mark
- Set Hint
- Reset Hint
- Reset All Hint
- Load RAM Data
- Save RAM Data

- Save To excel
- RAMBANK0

● **Hint**

Use DS defined SRAM; Hint will be automatically generated in corresponding window address. When cursor point to the address, it will show the defined string.

Ex: Program definition SRAM

MEMAR	080h		
MD1	DS	1	
MD2	DS	1	
MD3	DS	1	
MDL1	DS	1	
MDL2	DS	1	
MDL3	DS	1	
MD4	DS	5	
S_REG	DS	1	
r_Len	DS	1	
SQRTmp	DS	4	
Temp	DS	16	

After assembling, it will enter into debug status, displaying memory window.

When cursor points to 80h address, <80>: MD1 will be shown.

When cursor points to 86h address, <86>: MD4[0] will show up.

When cursor points to 87h address, <87>: MD4[1] will show up.

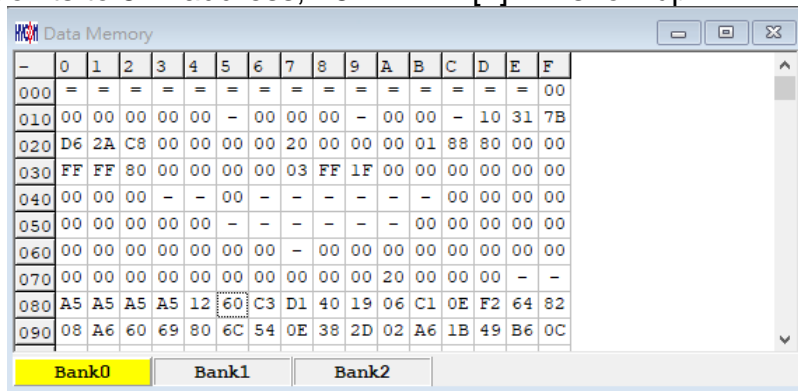


Figure 21

- There are two ways to revise SRAM value
 1. Point the cursor to the selected revised lines, click mouse's left key and Key IN directly.
 2. Point the cursor to the selected revised lines, double click the mouse's left key, a window will pop up as Figure 22 shown. Users can key in on keyboard or press the button by mouse.

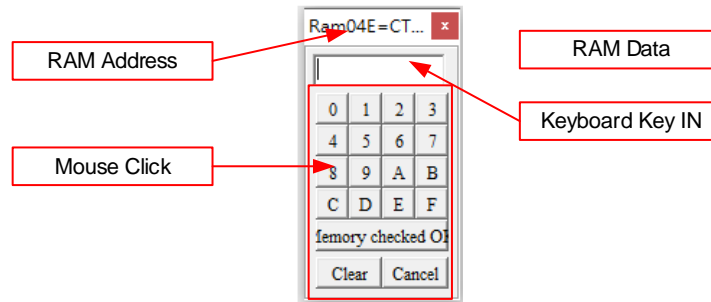


Figure 22

3.3. Register Window

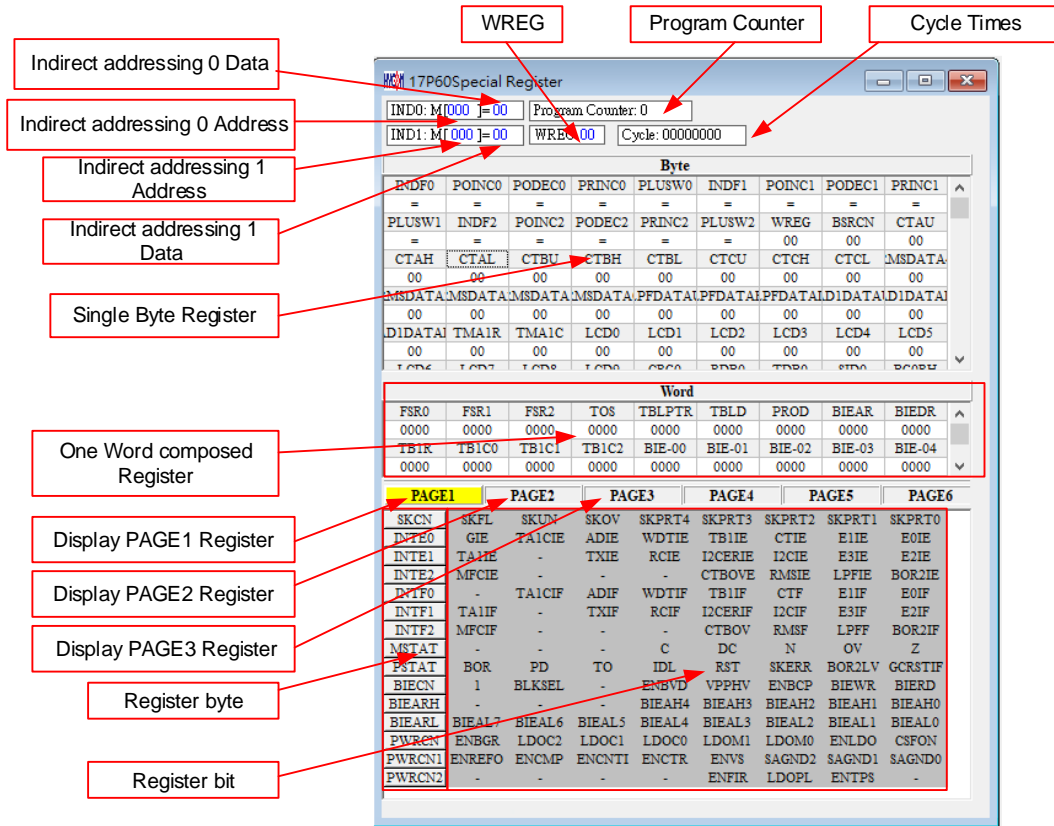


Figure 23

- Revise Indirect Addressing Data or Address
After setup as Figure 24 illustrated, Address can be revised through typing on the keyboard or by pressing the value by mouse.

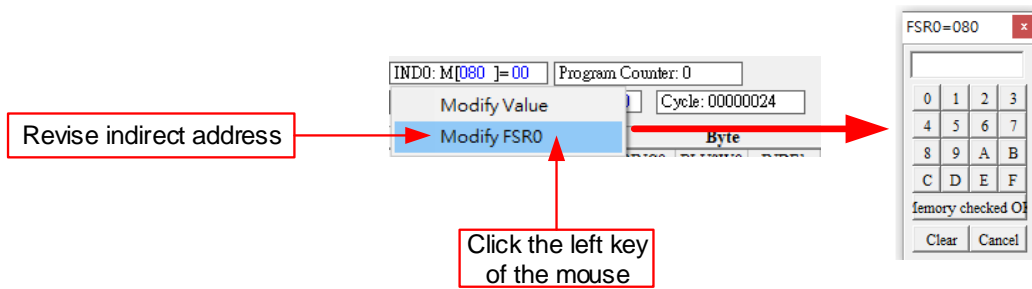


Figure 24

After setup as Figure 25 illustrated, Data can be revised through typing on the keyboard or by pressing the value by mouse.

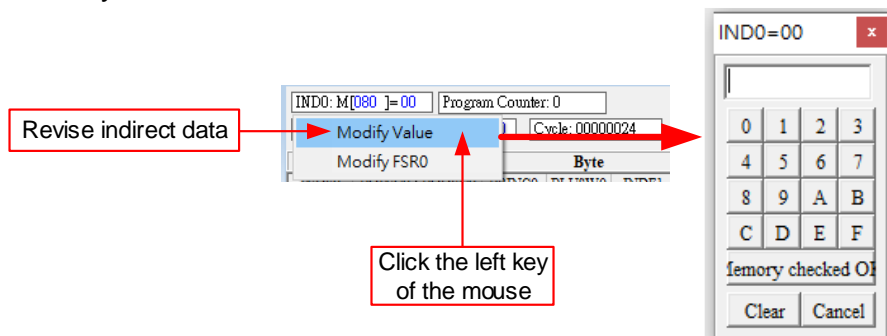


Figure 25

- Revise WREG Data

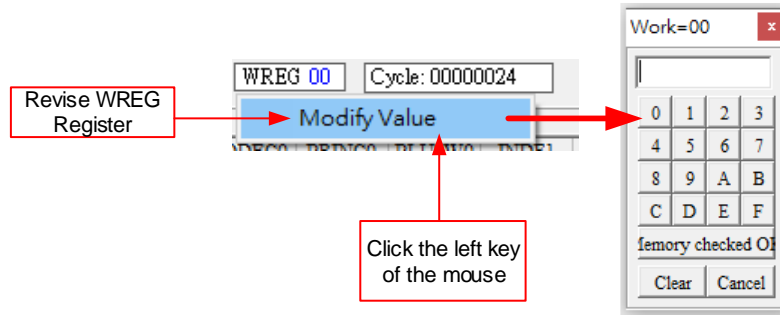


Figure 26

- Revise single 1byte or Word Register Data

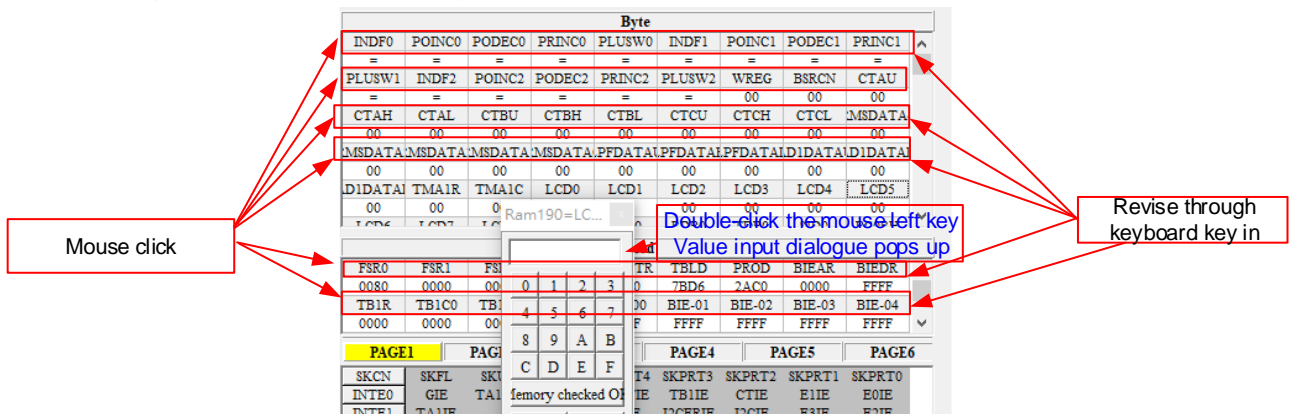


Figure 27

- Revise Register single 1 byte or single 1 bit
After Bit is configured as 1, its value will be highlighted in blue font.
After Bit is configured as 0, its value will be shown in black font.

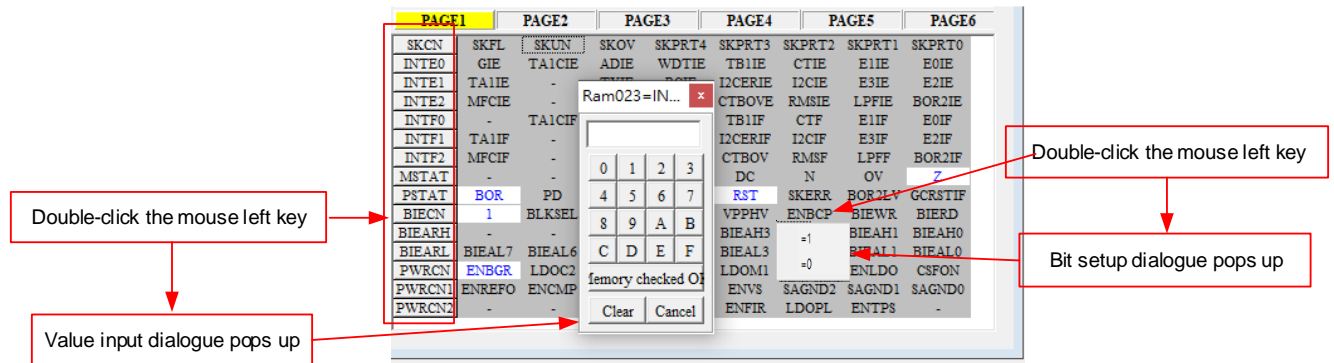


Figure 28

3.4. Watch Window

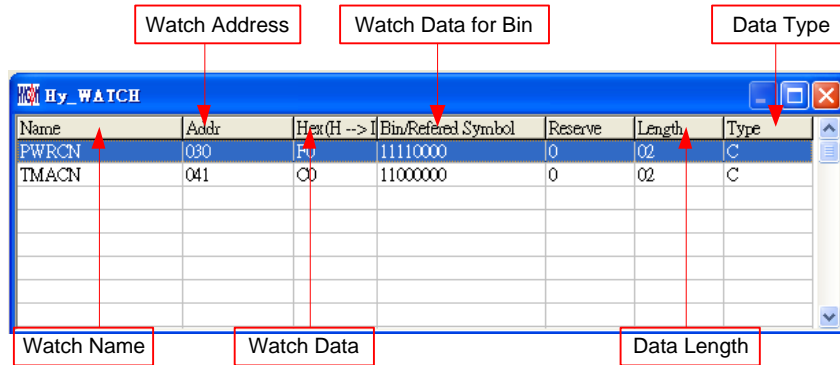


Figure 29

- Watch Name → Monitored Data name, program uses EQU or DS defined name.
- Watch Address → Monitored Data Address
- Watch Data → Reveal data. It is selectable to be arranged from right to left or from left to right. It can also display decimal or hexadecimal system.

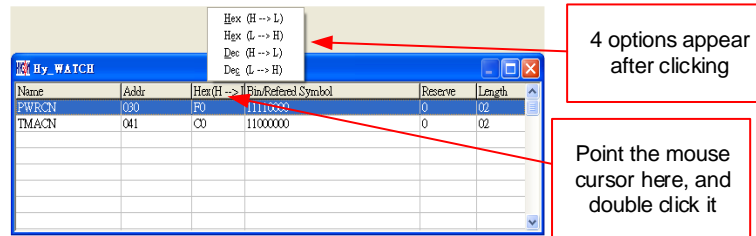


Figure 30

Hex (H → L) : Hexadecimal display, address H/L shows from low to high
 Hex (L → H) : Hexadecimal display, address L/H shows from high to low
 Dec (H → L) : Decimal display, address H/L shows from low to high
 Dec (L → H) : Decimal display, address L/H shows from high to low

- Watch Data for Bin → Data display in binary system, only for those EQU defined Address.
- Data Length → Data length, showing DS definition length; if EQU definition is applied, this value will show “2”.
- Data Type → Data type; D = DS definition; C = EQU definition.



Monitor EQU defined Register or RAM, click the right key of mouse to select add-in monitored Register or RAM as Figure 31 described.

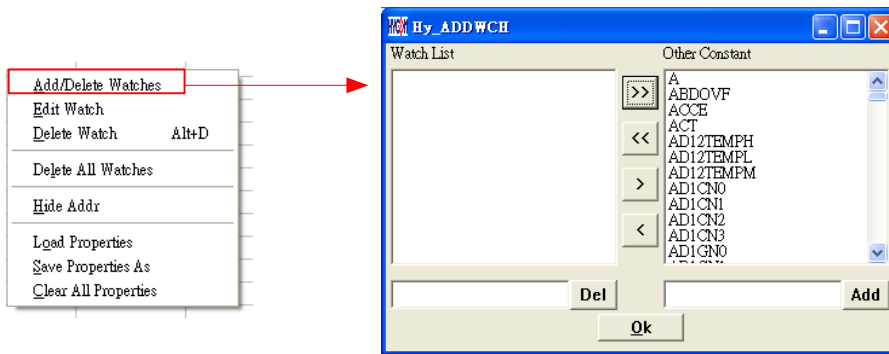


Figure 31

3.5. Stack Window

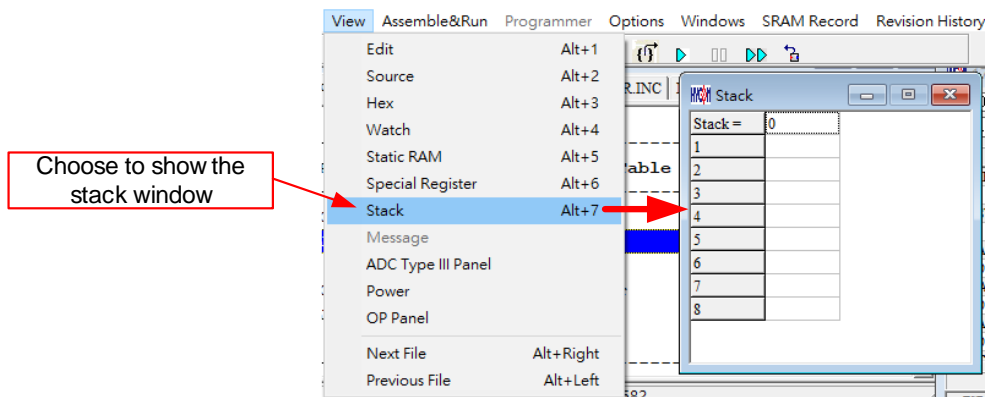


Figure 32

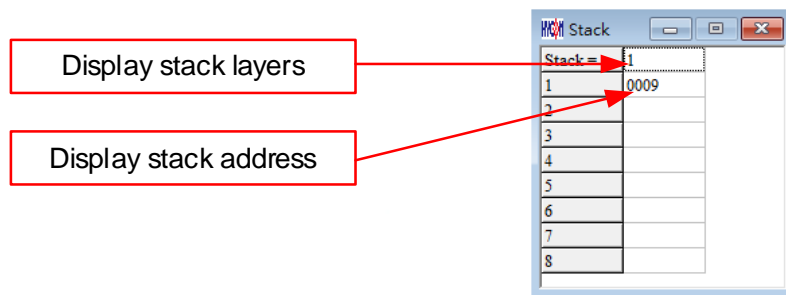


Figure 33

3.6. Register & SRAM Revise Record

If the register or SRAM has been revised manually after access to emulation window (hardware emulation or software emulation), the data will be recorded (despite the RAM, Register, ADC, OP and CMP is revised through any kind of window). The data will be revealed after pressing the button “RAM revise record”. At this time, windows will suspend until it is closed to execute other commands.

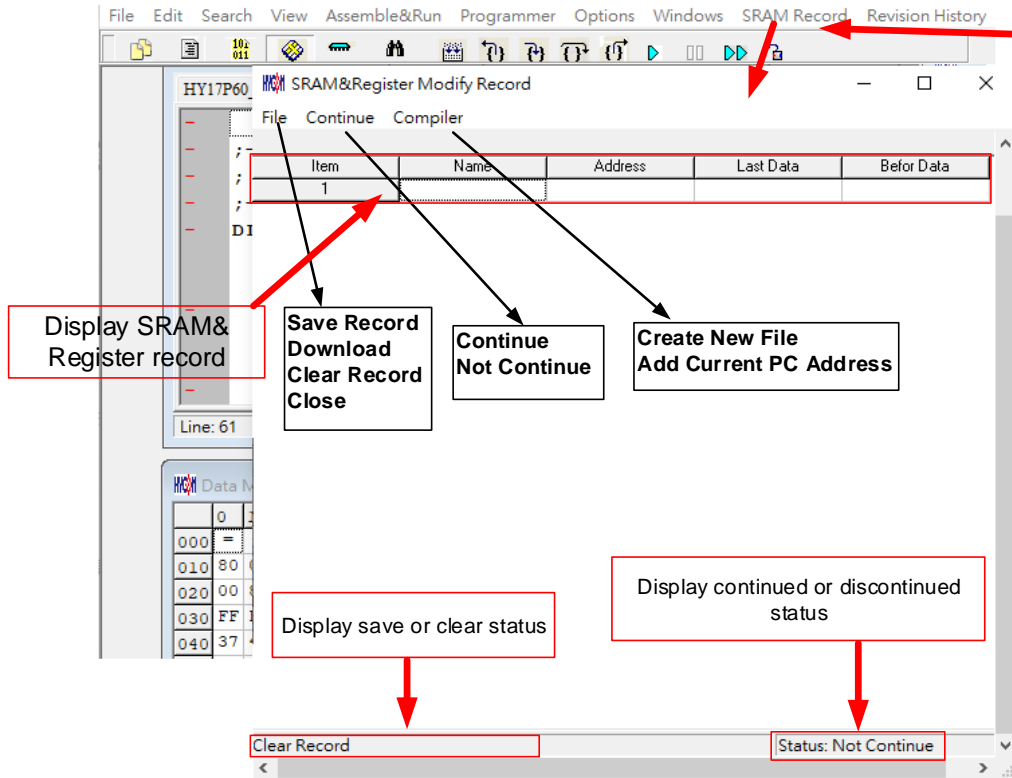


Figure 34

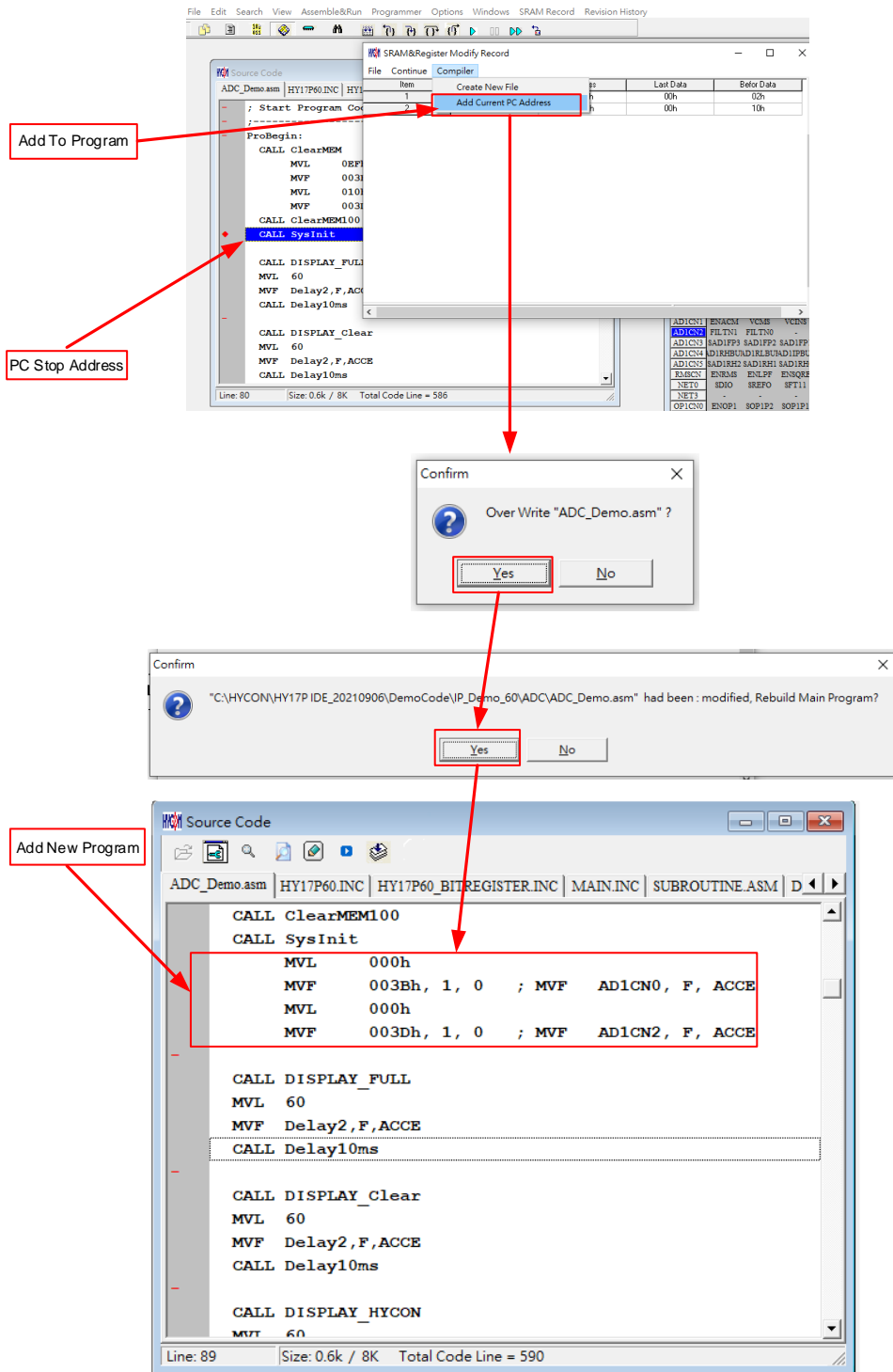


Figure 35

3.7. Hint Function of Source Code Window

If users intend to know Register or SRAM value and Address in source code window, point the cursor to register or SRAM, the name, address and data can be revealed. This function is only applicable to the instructions below: CLRF, ADDF, INF, INSZ, DCF, DCSZ, SUBF, COMF, ADDC, ANDF, IORF, XORF, SUBC, RRF, SETF, MULF, RLF, JZ, RRFC, RLFC, SWPF, DAW, INSUZ, DCSUZ, ARLC, ARRC, CPSG, CPSL, CPSE, TFSZ, BTFF, BSF, BCF, BTSS, BTSZ, MVFF(not Macro).

- Only the first followed argument is effective as Figure 36 described.
- When command is BCF, BSF, BTSS, BTSZ and BTGF, Byte value will be revealed if the cursor points to the first argument. If the cursor points to the second argument, it will display the specified Bit value (1 or 0) as Figure 37 illustrated.
- When command is MVFF (not Macro), first argument value will appear if the cursor points to the first argument. If the cursor points to the second argument, argument value will show up as shown in Figure 38.
- If the argument is INDF0, POINC0, PODEC0, PRINC0, INDF1, POINC1, PODEC1 and PRINC1, the Data will be FSR0 or the address Data of FSR1 as Figure 3-49 described.
- If the argument is PLUSW0 or PLUSW1, the Data is FSR0+WREG or the address Data of FSR1+WREG as illustrated in Figure 40.

```

include    sysincl.asm
;=====
        mvl    0E0h
        mvf    SPIINDEXL, F, ACCE
;===== SPIINDEXL[0F8h] = 7Ah
        mvl    13h        ; 識別碼
        mvf    0F5h, F, ACCE
        mvl    1h        ; 番號
    
```

Annotations: Name (points to SPIINDEXL), Address (points to 0F8h), Data (points to 7Ah).

Figure 36

```

;=====
MainLoop:
        btsz   SVSCN, SVSOP, ACCE
        bsf   SVSCN[02Dh] = 00h
        btss  SVSCN, SVSOP, ACCE
    
```

Annotation: Show Byte Value (points to SVSCN[02Dh] = 00h).

```

MainLoop:
        btsz   SVSCN, SVSOP, ACCE
        bsf   RLCDG, SVSCN[02Dh].5 = 0
        btss  SVSCN, SVSOP, ACCE
        bcf   RLCDG, b_lbat, ACCE
    
```

Annotation: Show Bit Value (points to SVSCN[02Dh].5 = 0).

Figure 37

```

        bsf   INDF0, 4, ACCE
; AS 10, 80h
        mvff  RAMFG+1, ADCFG+1
;===== RAMFG+1[0A5h] = FFh
;=====
MainLoop:
        btsz   SVSCN, SVSOP, ACCE
    
```

Annotation: First argument (points to RAMFG+1).

```

        bsf   INDF0, 4, ACCE
; AS 10, 80h
        mvff  RAMFG+1, ADCFG+1
;===== ADCFG+1[0A9h] = 7Fh
;=====
MainLoop:
    
```

Annotation: Second argument (points to ADCFG+1).

Figure 38

Name	FSR0 address	Data
------	--------------	------

```

mvff    INDF1, PLUSW0
bsf     INDF0, 4, ACCE
; AS 10,80h INDF0[120h] = FEh
mvff    RAMFg+1, ADCFg+1
;=====
;=====
MainLoop:
    btsz    SVSCN, SVSOP, ACCE
    bsf     PLUSW0, 15, ACCE

```

Figure 39

Name	FSR0+WREG Address	Data
------	-------------------	------

```

mvl     4
mvff    INDF1, PLUSW0
bsf     INDF0, PLUSW0[145h] = A7h
; AS 10,80h
mvff    RAMFg+1, ADCFg+1
;=====

```

Figure 40

4. Programming Window

4.1. Interface setting

Click “Option” button and select interface setting to get access to programming window as Figure 41.

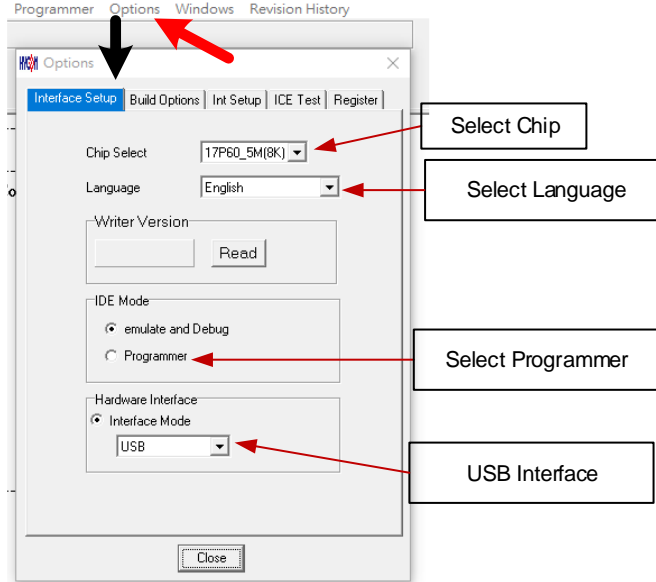


Figure 41

IC Selection → Select the IC part no. If the programming IC differs from the selected part no., Blank Check, Program and Verify will fail.

Language → Select the language of operating interface, either Chinese or English.

Hardware Setting → USB or Parallel Port interface is selectable.

IDE Mode → Select programming.

When the interface setting is accomplished, click” Assemble Option” to select programming setting as Figure 42 indicated.

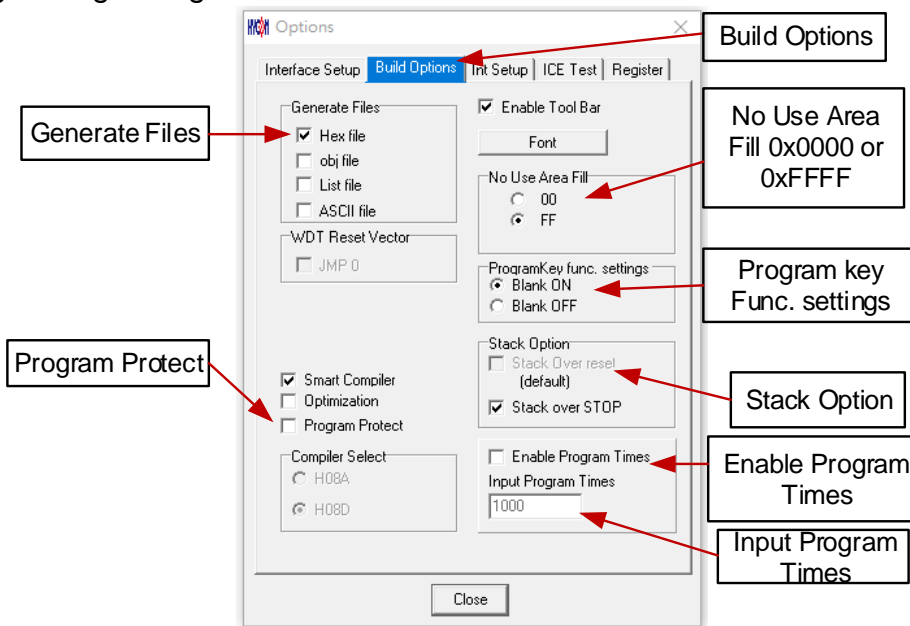


Figure 42

- Assemble Generated Extension: The generated file after selecting the programming program.
- Stack Operation: Select whether to replace when stack overflow/underflow occurred after OTP program operation.
- Fill Unused Zone: Fill the unused zone with 00 or FF in the program after programmed.
- Simplified Assemble: Select whether to simplify assemble.
- Enable Program Times: Select whether to enable Download program's programming times
- Input Program Times: Fill in Download program's programming times.(Maximum is 2147483646, minimum is 1).

After assembling finished, click "ICE Test" to evaluate testing voltage as Figure 43 described

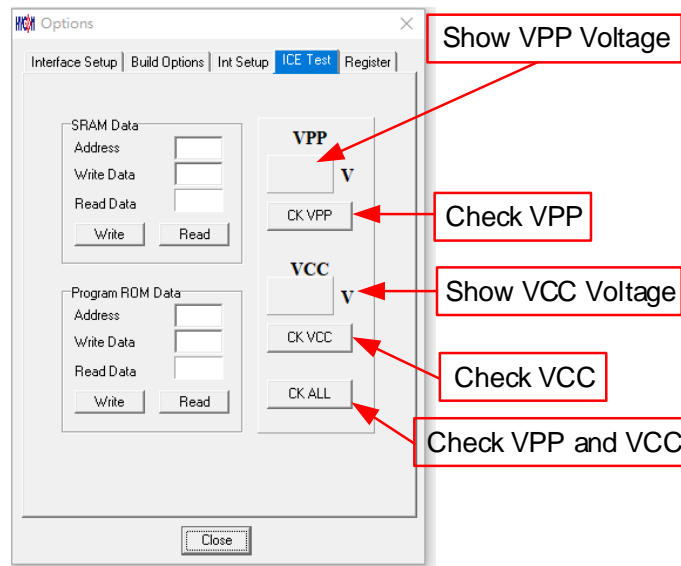


Figure 43

VPP voltage while programming: 8.5V

VDD voltage while programming: 4.5V

After ICE operation voltage test finished, click "OSC Calibration" doing HAO/LPO Calibration.

- **Note for attention before using the function of "OSC Calibration":**
 - If HAO/LPO Calibration programming started, the code, **0FEH/0FFH** of RAM has its own meaning when IC is power-on.
 - The programming time will be **500msec.** longer (upon LPO Software Calibration Started)
 - HAO/LPO Calibration is not doing actual frequency calibration; only provide the difference of frequency for calculation.
 - **PC Online Programming only supports HAO Hardware Calibration, doesn't support Software Calibration.**

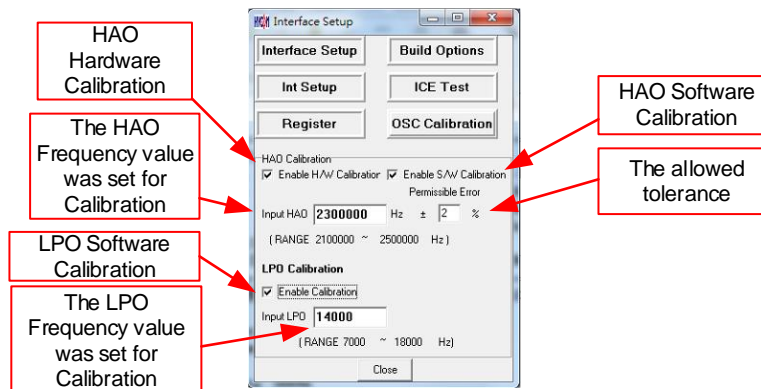


Figure 44

- HAO Calibration:
 - Enable H/W Calibration: Enable HAO Hardware Calibration and doing the system calibration. This function has to be confirmed if it works after the IC model No. has been designed.
 - Enable S/W Calibration: Enable HAO Software Calibration. The difference value is saved in code **0FEH** of RAM.
- LPO Calibration:
 - Enable S/W Calibration: Enable LPO Software Calibration. The difference value is saved in code **0FFH** of RAM.
- Input HAO or Input LPO: Input the set frequency value for calibration.
- Permissible Error: Tolerance between calibrated frequency value and the set frequency value.

Please find the description of Software Calibration as below:

- HAO Software Calibration :
 - After Calibration, the difference value is saved in code **0FEH** of RAM. This function is not doing actual frequency calibration; only writing in the difference of frequency when the IC power-on.
 - HAO Hardware Calibration, HAO Software Calibration can exist at the same time. Moreover, Hardware Calibration enforced first and then proceed Software Calibration to calculate the difference value.
 - The difference value of frequency defined as **4000HZ/LSB**.
 - The code **0FEH** data format:
 - Bit7 : 0= +, 1= - ; Bit6~Bit0 means frequency difference value.
 - 01H mean the frequency difference value is +4000HZ ; FFH mean the frequency difference value is -4000HZ.

■ Example :

The set value for HAO frequency calibration is 2000000HZ , and the actual value from IC is HAO=1920000HZ. The calculation is $(1920000-2000000)/4000 = -8000/4000 = -20$, therefore the code 0FEH will be **1110 1100b**

■ Example1 :

The set value for HAO frequency calibration is 2000000HZ , and the actual value from IC is HAO=2008000HZ. The calculation is $(2008000-2000000)/4000 = 8000/4000 = 2$, therefore the code 0FEH will be **0000 0010b**

- LPO Software Calibration :
 - After Calibration, the difference value is saved in code **0FFH** of RAM. This function is not doing actual frequency calibration; only writing in the difference of frequency when the IC power-on.
 - The difference value of frequency defined as **64HZ/LSB**.
 - The code **0FFH** data format:
 - Bit7 : 0= +, 1= - ; Bit6~Bit0 means frequency difference value.
 - 01H means the frequency difference value is +64HZ ; FFH means the frequency difference value is -64HZ.
 - Example:

The set value for LPO frequency calibration is 28000HZ , and the actual value from IC is LPO=28128HZ. The calculation is $(28128-28000)/64 = 128/64 = 2$, therefore the code 0FFH will be **0000 0010b**
 - Example1:

The set value for LPO frequency calibration is 28000HZ , and the actual value from IC is LPO=27872HZ. The calculation is $(27872-28000)/64 = -128/64 = -2$, therefore the code 0FFH will be **1111 1110b**

When Interface Setup finished, please press “close” and all the set parameters will be saved. Next time, when you open the file, it will load the record parameters automatically and the IC model No. will be displayed on the headline of window as Figure 45.

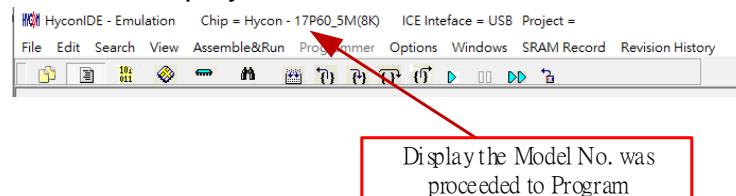


Figure 45

4.2. Operation Procedures

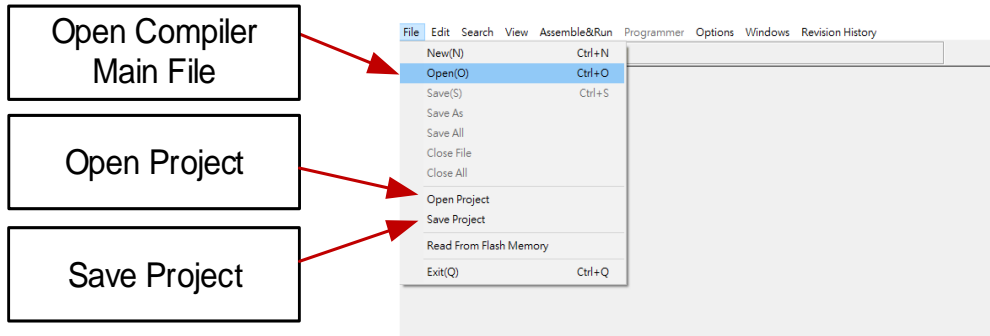


Figure 46

- Open → Open the programmed source code main file.
- Open Project → Open the saved project.
- Save Project → Save the finished project.

4.2.1. Open file and Assembly

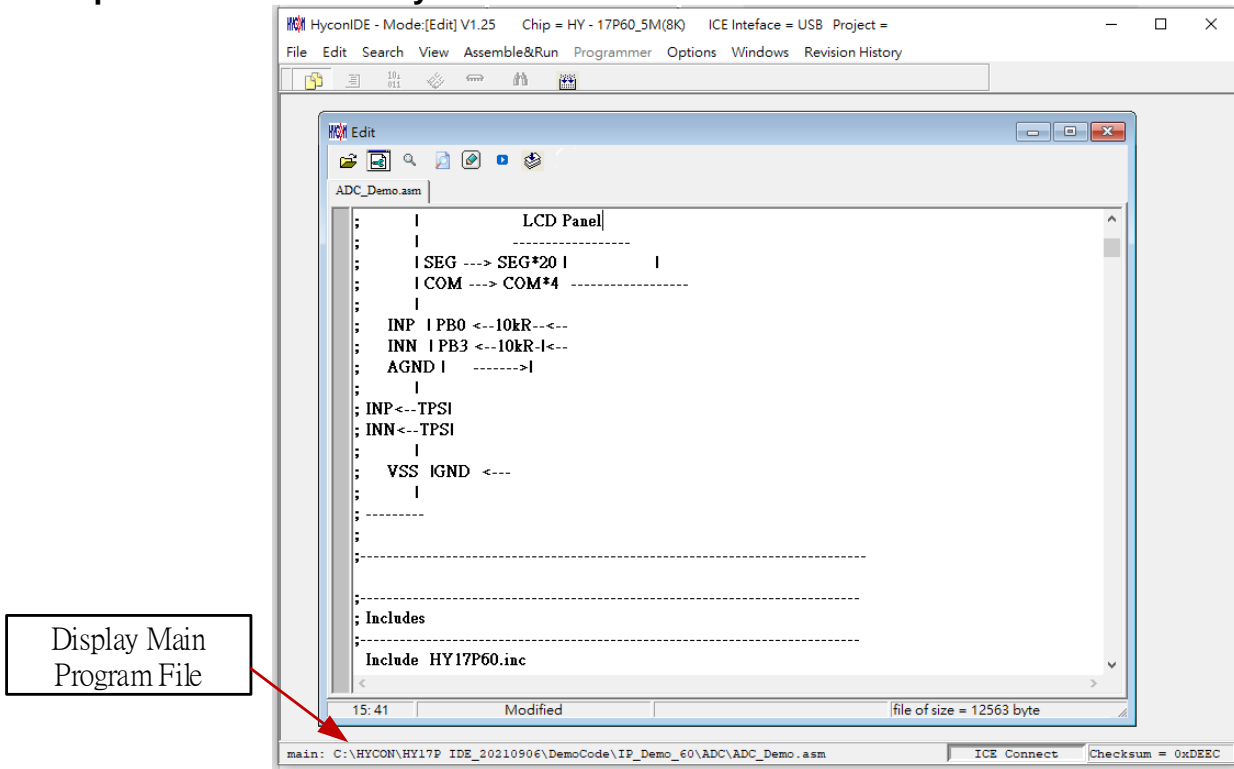


Figure 47

Open source code main file and it will be displayed as the assembly file. If the displayed name differs from main file, points the mouse to the specific file and presses mouse right key. Set this file as the assembly main file as shown in Figure 48.

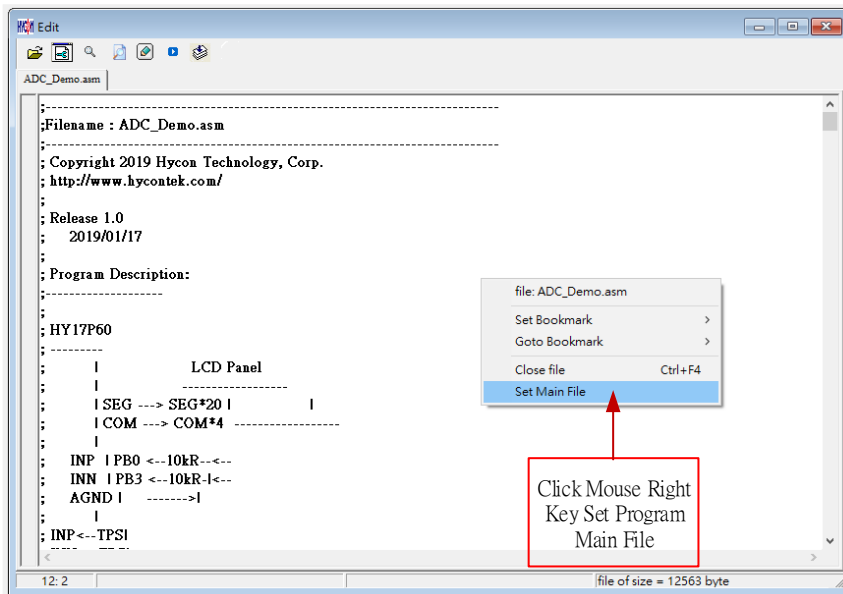


Figure 48

Assembles Source Code and download the file to programmer or IDE Flash Memory, as Figure 49 illustrated.

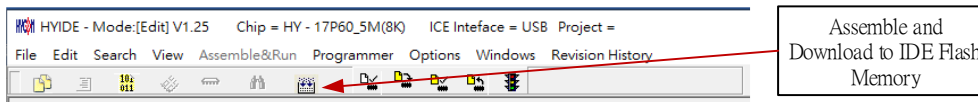


Figure 49

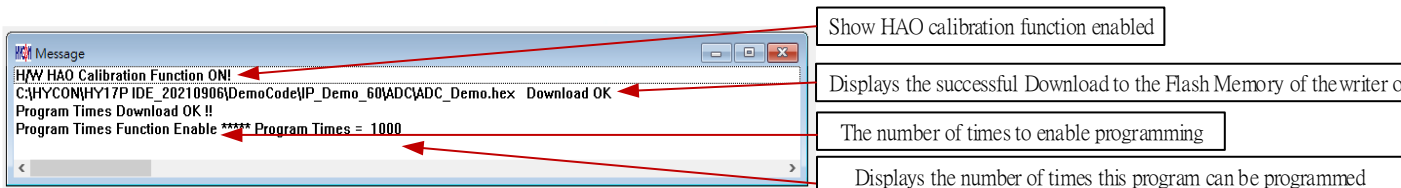


Figure 50

1. When using USB interface, the assembly finished program code will be loaded into programmer or Flash Memory of IDE for mass production programming.
2. If there is enabled program times in the assembly option, information column will display the programming times as shown in Figure 50.
3. After assembling completed, Hex filename and Checksum will be displayed in underneath section, as Figure 51 illustrated.

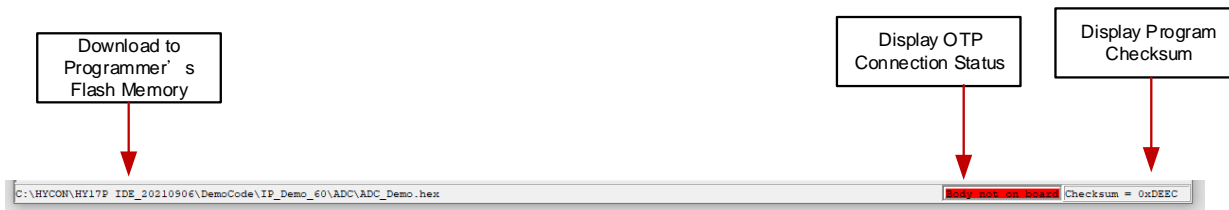


Figure 51

4.2.2. Download HEX File

If users would like to download the Hex File, please download it by HY17P-Hex Loader software and follow the guidance of user manual.

4.3. PC Online OTP Programming

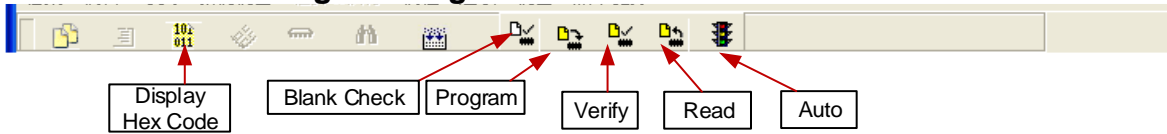


Figure 52

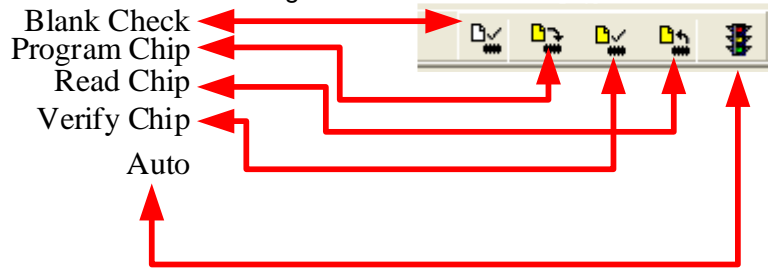


Figure 53

Blank Check, Programming, Verify and Read Commands can be implemented when the programmed file being successfully loaded into programmer or IDE Flash Memory. On the contrary those commands will not be activated if the download failed.

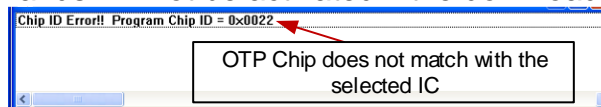


Figure 54

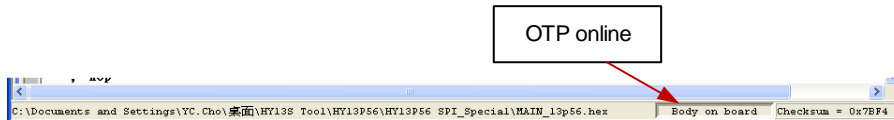


Figure 55

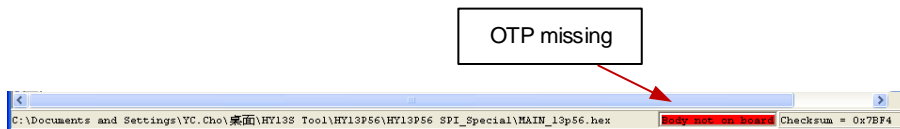


Figure 56

Make sure the selected programming IC part number is the same with the OTP part number in the topic window. When programmer executes Blank Check, Programming and Verify commands, Program will check whether the IC part number and OTP programming part number are identical. If the part number is different, the data will not be written into OTP and an error message will be displayed in information column as Figure 54 described. If users intend to find out whether the part number is correct before programming, point the cursor to "IC Connection Status Display" and click the mouse left key. If the selected IC is correct, a message will show up as Figure 55. If it is incorrect, the message will be displayed as Figure 56. If "Enable Program Times" has been marked up, the spare program times will display in the message column as Figure 57 illustrated.



Figure 57

4.3.1. Blank Check


The icon of Blank check is . The internal code of Blank ICs that have yet been programmed is 0xFFFF. The purpose of checking the IC is to make sure the OTP address content is 0xFFFF. If the IC selection is correct and the content is empty, a message will appear as Figure 58.




Figure 58

If the IC selection is incorrect or the content is not empty, a message will show up as Figure 59 described.



Figure 59

4.3.2. Program

The icon of Program is . The purpose of programming is to write Compiler accomplished program into IC OTP. When programming is completed and the IC is assembled as finished goods, programmer can operate the program as users commanded. Program the downloaded or assembly finished Hex file (displayed at the bottom of the column) in the selected IC and verify the correctness of the programming content (please refer to Chapter 4.2.1 or 4.2.2 for programming procedures). If the selected IC is correct and the programming succeeds, message will appear at the information column as Figure 60 illustrated. If "Enable Program Times" is ticked up, the enable program times will minus 1 and the program times left will be revealed in the message column.

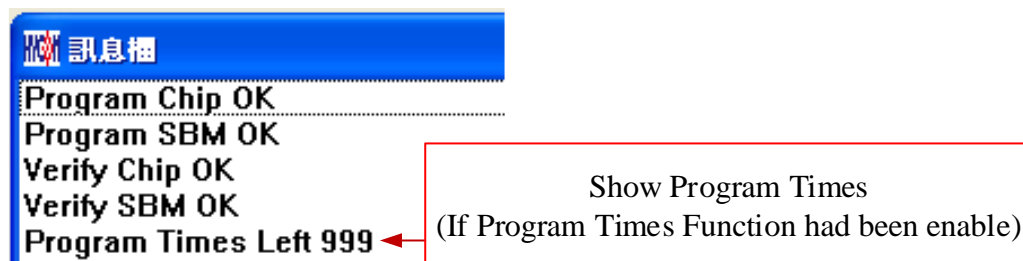


Figure 60

If the IC selection is incorrect or the programming fails, a message will show up as Figure 61



Figure 61

4.3.3. Verify

The icon of Verify is . The purpose to verify programming IC is to compare if the program written into IC OTP equals to the program downloaded to programmer or IDE Flash Memory. Verify programming IC content consistency with the downloaded or assembled Hex file (displayed at the bottom of the column). If the IC is protected by program, this verification is ineffective or the comparison failed.

If IC selection and program verification is success, a message will appear as Figure 62.



Figure 62

If IC selection is incorrect or the program verification miscarries, a message will pop up as Figure 63



Figure 63

4.3.4. Read

The purpose to read the IC is to verify the consistency of OTP Checksum and programmed Hex file. To read IC content, the procedures are illustrated as Figure 64. Its content will reveal at "Display Code" window. If the IC is protected by program, this function is ineffective or the comparison failed.

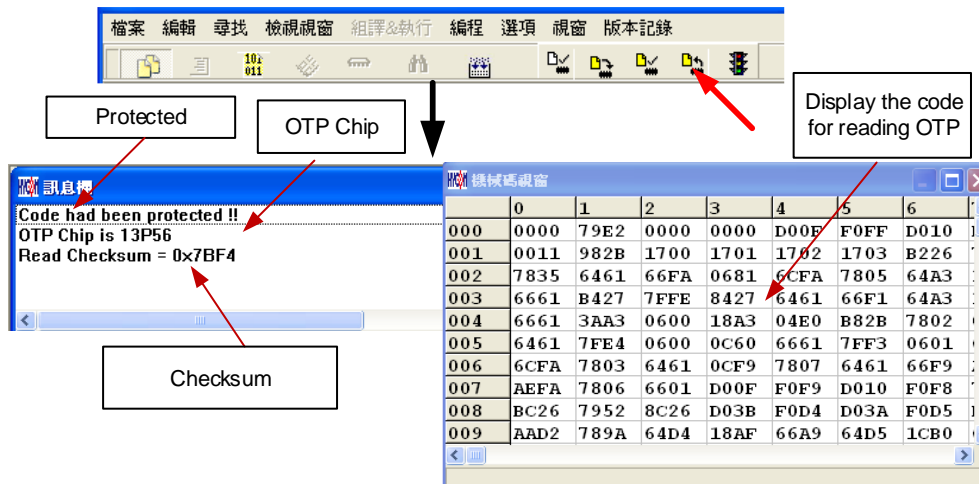



Figure 64

4.3.5. AUTO

The icon of Auto is . Auto integrates Blank Check, Program and Verify function. If user selects Auto, it will first check whether the IC is blank, then to program and verify. After the execution succeeded, a message will be displayed as Figure 65 displayed. If the option, "Enable Program Times" is ticked up, the program permitted times will reduce 1 and the program times left will be shown in the message column

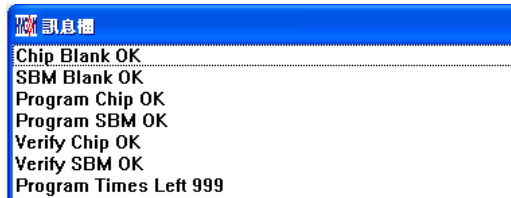


Figure 65

If any function fails, the whole process will stop and display an error message in the message column.

4.4. PC Offline Programming

4.4.1. Program Description

The following description takes HY17P60B as an example

When the user program enters the engineering trial production stage from the development stage, the HY10000-WK08D writer can be used alone (as shown in Figure 66) without connecting to the PC.



Figure 66

- ◆ USB: USB connector to PC end
Downloading programming code for HY17P series.
- ◆ Programming control end of HY17P series and is connected to OTP.

PIN 1	VPP	connected to VPP of IC
PIN 2	PSCK	connected to PSCK of IC
PIN 3	PSDI	connected to PSDI of IC
PIN 4	PSDO	connected to PSDO of IC
PIN 5	VDD	connected to VDD of IC
PIN 6	VSS	connected to VSS of IC
- ◆ Program, (Program->Verify), for offline program operation.
- ◆ Blank, Blank Check button, for offline operation.
- ◆ Information, View the internal Hex file information of the writer
- ◆ L1 Green LED : USB or Adapter is Power-on signal. OTP Programming、 Blank Check...etc. Success signal.
L2 Red LED : OTP Programming、 Blank Check...etc. Failure signal.
L3 Yellow LED: Programming

Figure 67 below shows the connection method of the chip programmed online and the programming pins of the control board when the program is Download and the programming pin is on the PC:

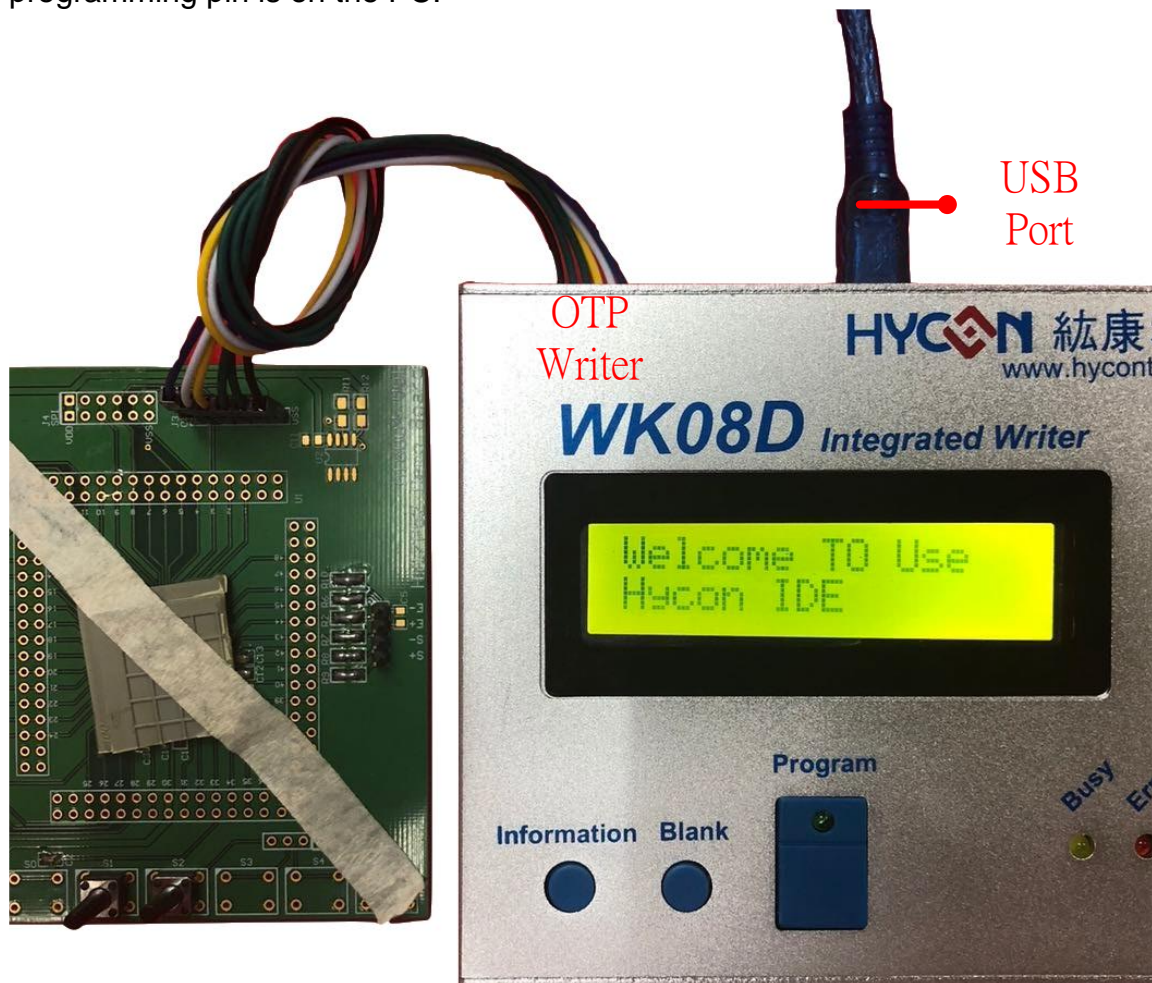


Figure 67

- To implement offline operation, Hex file must be firstly downloaded to programmer Flash Memory. The procedure can refer to chapter 4.2.1 or chapter 4.2.2.
- To implement offline operation, press blank button can check if the IC is blank and the OK Green LED should be lighted up.
- Programming button. Its procedures are: Blank Check -> Program ->Verify
If "Program Protection" of "Assemble Option" is picked up before downloading data to Flash Memory, program protection will be executed after Verify completed. If "Program Protection" is not picked up, it will stop after Verify accomplished and OK Green LED will be lighted up.
- When Programming finished, please press blank to check if the IC is blank. At this moment, the Error Red LED should be lighted up means the programming is success (because the data has been programmed into IC, so Blank Check is failed.)
- If any failure or error happened during execution procedures, Error Red LED will be lightened up. On the contrary, OK Green LED will be lighted up if success.

4.4.2. Program Times Restriction

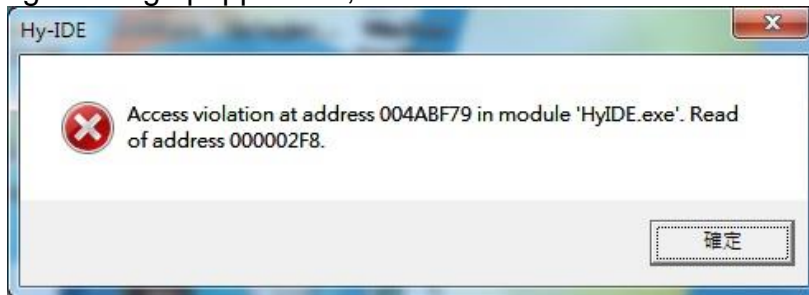
The menu of "Assemble Option" in interface setup has an option of "Enable Program Times". This option restricts the permitted program times of download program. This is a safety mechanism that restrains the permitted program times, preventing it from over-programming on the production line.

This is a safety mechanism that restrains the permitted program times, preventing it from over-programming on the production line. After ticking up "Enable Program Times", key in the program times in the column below "Input Program Times" (maximum is 99999999, minimum is 1). This argument will be written into EEPROM of the programmer after the compiler programmed file is downloaded to Flash Memory. Afterwards, the enabled program times will reduce 1 each time when programming completed. If the value reduced to 0, the programming action may not be executed. At this time, an error signal (Red LED) will be lighted up but Blank Check still operates normally.

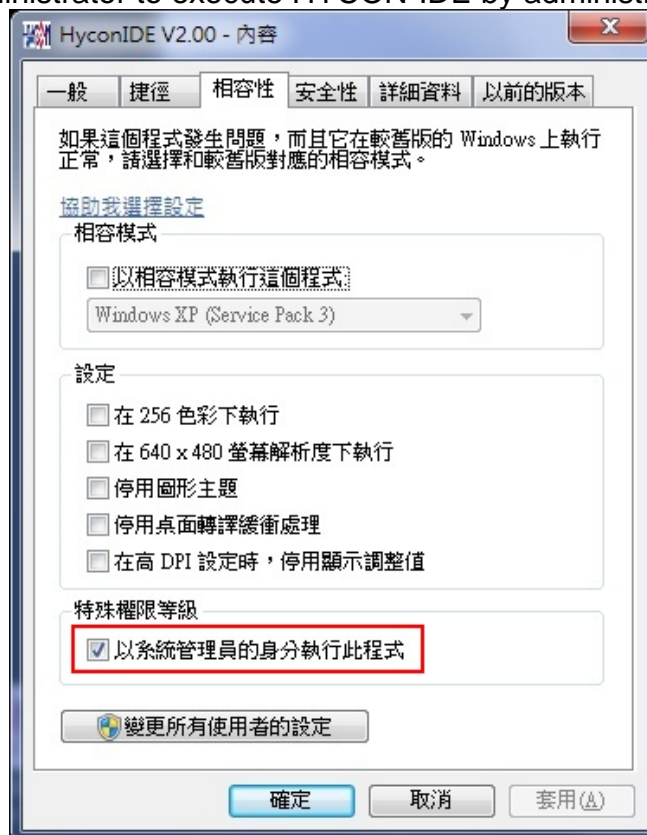
5. Troubleshooting

5.1. HYCON-IDE Execution Problem

If the following message popped out, it means HYCON-IDE can't be executed normally



The problem might be happened under Microsoft Vista or Windows 7 system environment. To avoid the problem, the limit of authority for HYCON-IDE execution has to be set as system administrator to execute HYCON-IDE by administrator status.



6. Revision History

Major differences are stated thereafter

Version	Page	Date	Revision Summary
V01	ALL	2018/11/16	First edition
V02	ALL	2022/01/21	Modify the product model, modify the layout