



# **HY-MCU Compiler User Manual**

## Table of Contents

1. COMPILER.....	3
2. COMPILER DOCUMENT FILE RESTRICTION .....	4
3. PSEUDO .....	5
4. OBJECT CODE.....	11
5. DIFFERENCES OF ASSEMBLING AND ASSEMBLING && DEBUGGING .....	12
6. ERROR MESSAGE.....	12

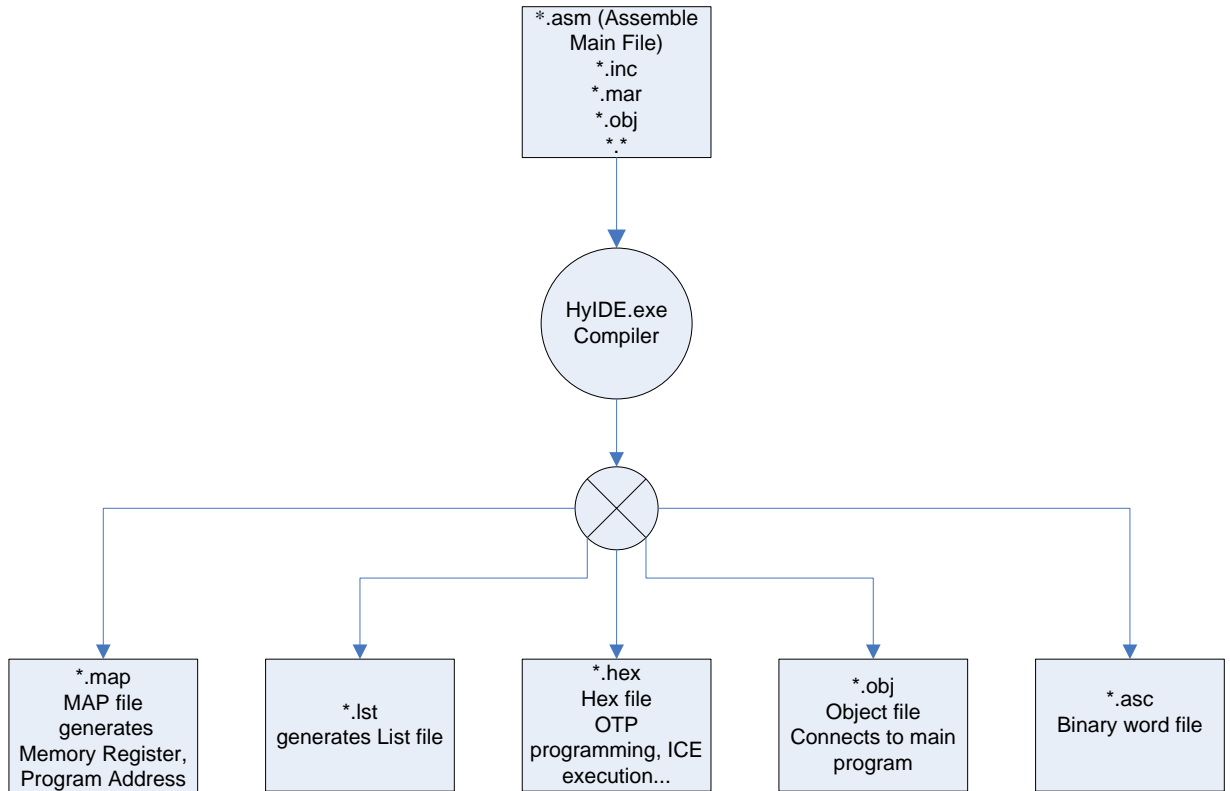
Attention :

1. Information included in this manual concerning device applications and analogous content is furnished only for user convenience and HYCON does not accountable for the content usage and any incurred outcome. The content of specification may be replaced by updates; it is users' responsibility to assure that your application meets with your requirements and standards.
2. Should buyers applied HYCON products as key devices in life support or safety system is entirely at buyers' risk. Any use of HYCON products in such application is forbidden. HYCON reserves the right to change without further notice. For most up-to-date information, please visit our website:

<http://www.hycontek.com>

## 1. Compiler

File generated flow:



### 1.1 Compiler Main File

Program executes main file to compile main filename. Compiler can accept all kinds of filename extension excluding obj, hex, asc, lst, map, msk, pro, err and ifo.

### 1.2 Object File

Object connected file, the filename extension is \*.obj that offers main program quoted function.

### 1.3 MAP File

Argument generated file, filename extension is \*.map. Arguments include all EQU, DS defines Memory or Register Address, Label Address of Object file.

### 1.4 List File

Generates data after compiling, program lines, Address, machine code, main program.

### 1.5 Hex File

Generates Hex file, for ICE procedure emulation, OTP programming.

### 1.6 Asc File

Generating machine code binary word file.

## 2. Compiler Document File Restriction

1. Maximum document file is 128k byte. If maximum restriction is exceeded, Compiler will not send out a warning. Documents may be damaged. Users must pay extra attention to document size.
2. If the document exceeds 128k, users must utilize Include way to separate the file. Users must take notice of the documents size that are 128k byte respectively.
3. No matter the text is uppercase or lowercase, if word set is the same, it will be recognized as the same word set.
4. Include `xxxx.obj` , must be put the end
5. Call `xxxx.obj` function ,under this line of program ,the instruction of “nopf” function should be added.

Ex:

```
CALL HY17P52WR3
NOPF HY17P52WR3
```

6. Filename length cannot exceed 128 byte, including Path.

Ex:

```
C:\Example\ASM\example.asm ← this string length cannot exceed 128 byte.
```

7. Word length constrain is 160 byte of every line in editing program.

Ex:

```
MVF TEMPABLE, F, BANK ← This string length cannot exceed 160 byte
(space included).
```

8. Maximum Label, Memory and Register definition word length is 32 byte.

Ex:

```
Label1: ← This string length cannot exceed 32 byte
ECK equ 80h ← 'ECK' cannot exceed 32 byte
```

9. Label, Memory and Register can include at most 8192 definition (Total).

10. Macro Maximum number is 4096.

11. Macro defines maximum argument as 32.

Ex:

```
Exm Macro A,B,C..... ← A,B,C argument cannot exceed 32
```

12. If Macro refers to another Macro, maximum levels cannot exceed 32.

Ex:

```
ANB Macro CC
    Exm
ENDM
ANB1 Macro CC
    ANB
ENDM
ANB2 Macro CC
```

```

        ANB1
    ENDM
    .....
    ANBn  Macro  CC
        ANBn-1          ← Cannot exceed 32 levels.
    ENDM
    
```

### 3. Pseudo

Pseudo instructions are used when defining program, memory or special register address. These instructions do not occupy memory or program space (except pseudo instructions, 'DB' and 'DW' for defining program).

Compiler's pseudo instructions are listed in below:

INCLUDE	Open another document
ORG	Define program address
END	End Program
MACRO	Definition module
ENDM	End Module definition
EQU	Define constant
DB	Define 1 Byte program
DW	Define 1 Word program
MEMAR	SRAM Address declaration
DS	SRAM length declaration
EXTENR	External argument reference
GLOBAL	Provide arguments for external program usage.
SET	Reserved
LOCAL	Reserved
UP	Label or address / 0x10000
HIGH	Label or address / 0x100
LOW	Label or address mod 0x100
DEFINE	Offer value to IF event arguments
IF	Conditions statement. Judge whether to compile statements before ENDIF
ELSE	Else, match up with IF wording
ENDIF	End IF

#### 1. INCLUD

Open files of main program, including Marco file, object file, Definition file and

Assemble file.

If Object or Assemble file is included, Compiler will arrange the instructions in sequence in ROM Address.

Ex:

SYSINI.asm content:

```
CLRF 080h
CLRF 081h
.....
```

MAIN.asm content:

```
ORG 0000h
RJ START
INCLUDE SYSINI.asm
ORG 0100h
```

START:

....

After Compiling MAIN.asm

Address	Program
0000h	RJ 100h
0001h	CLRF 080h
0002h	CLRF 081h
.....	
0100h	
....	

0100h

....

Neither Marco nor definition file will be arranged in ROM Address, unless Marco is called in the program. At this time, Marco programs will be arranged in corresponding ROM Address.

Ex:

Def.mar content:

```
W EQU 0
F EQU 1
ACCE EQU 0
BANK EQU 1
ADDWORD MARCO A, B
LBSR B
MVF B, W, BANK
LBSR A
ADDF A, F, BANK
LBSR B
MVF B, W, BANK
LBSR A
ADDC A, F, BANK
```

ENDM

MAIN.asm content:

```
INCLUDE Def.mar
BUFA EQU 080h
BUFB EQU 102h
ORG 0000h
GOTO START
.....
```

....

```

ORG      0100h
START:
MVL     012h
MVF     BUFA, F, ACCE
MVL     034h
MVF     BUFA+1, F, ACCE
MVL     056h
LBSR    BUFB
MVF     BUFB, F, BANK
MVL     078h
MVF     BUFB+1, F, ACCE
ADDWORD A, B
....

```

After Compiling MAIN.asm

Address	Program
0000h	GOTO START
....	
0100h	MVL 12h
0101h	MVF 80h, 1, 0
0102h	MVL 34h
0103h	MVF 081h, 1, 0
0104h	MVL 56h
0105h	LBSR 1
0106h	MVF 02h, 1, 0
0107h	MVL 78h
0108h	MVF 03h, 1, 0
0109h	LBSR 1
010Ah	MVF 02h, 0, 1
010Bh	LBSR 0
010Ch	ADDF 80h, 1, 1
010Dh	LBSR 1
010Eh	MVF 03h, 0, 1
010Fh	LBSR 0
0110h	ADDC 81h, 1, 1
....	

## 2. ORG

Appoint program memory address.

Ex:

```

ORG      0000h
RJ       Begin
....
ORG      0100h
Begin:
MVL     10h
....

```

After Compiling

0000h	RJ	101h
....		
0100h	MVL	10h

## 3. END

End Program

When pseudo instruction, END exists in the program, Compiler will stop to

assemble END below programs.

## 4. MACRO

Macro instruction

Instructions can be gathered into Macro to provide convenience to program reference.

Ex:

```
MOVFW  MACRO  A, B
MVF    A, 0, B
ENDM
```

```
MOVWF  MACRO  A, B
MVF    A, 1, B
ENDM
ORG    0000h
MOVFW  80h, 0
MOVWF  10h, 1
```

```
After Compiling
0000h   MVF   80h, 0, 0
0001h   MVF   10h, 1, 1
```

## 5. ENDM

End Macro instruction

After MACRO declaration, ENDM is necessary to end the instruction.

## 6. EQU

Define constant

EQU can be used to declare SRAM address

Ex:

```
COUNT  EQU  80h
```

COUNT address is 80h

## 7. DB

Define 1 Byte program

DB requires argument in even numbers. If it is singular number, 00h must be added ahead to compose a WORD.

Ex:

```
ORG    100h
DB     012h
DB     053h, 046h
```

After Compiling

```
0100h   0012h
0101h   05346h
```

## 8. DW

Define 1 WORD program

Ex:

```
ORG    100h
DW     01234h
```



After Compiling

0100h 01234h

## 9. MEMAR

Declare SRAM Address

SRAM initial address is defined by compiler through the arguments that followed MEMAR.

## 10. DS

Declare SRAM length

Ex:

```
MEMAR 080h    ← SRAM is arranged from 080h
TEMP   DS    16 ← Define TEMP as 16 byte (from 080h to 08Fh)
COUNT DS    2 ← Define COUNT as 2 Byte, address: 090h, 091h
BUF    DS    1 ← Define BUF as 1 Byte, address: 092h
```

## 11. EXTERN

Refer to external argument

When establishing Object Code, EXTERN declaration helps to define the external program arguments.

Ex:

```
EXTERN  TEMP1, TEMP2
GLOBAL  ADDBLK

        LEN    EQU    3
ADDBLK:
        MVF    TEMP2, 0, 0
        ADDF   TEMP1, 1, 0
        .....
```

## 12. GLOBAL

Provide arguments for external program usage.

When establishing Object Code, program arguments or subprograms can be released to provide reference to external program.

## 13. UP

Numeric highest bit → Numeric / 0x10000

Ex:

```
MVL    UP    Label1 → UP    Label1 = 0x1000 / 0x10000 = 0x00
...
ORG    01000h
Label1:
```

## 14. HIGH

Numeric highest bit → Numeric / 0x100

Ex:

```
MVL    HIGH  Label1 → HIGH  Label1 = 0x1000 / 0x100 = 0x10
...
ORG    01000h
Label1:
```

## 15. LOW

Numeric lowest bit → Numeric MOD 0x100

Ex:  
MVL    LOW    Label1    →    LOW    Label1 = 0x10F2 MOD 0x100 =  
0xF2  
...  
ORG    010F2h  
Label1:

## 16. DEFINE \ IF \ ELSE and ENDIF

DEFINE: Provide numeric for IF event arguments.

Condition statement, IF

Usage:

```
IF Event
    Content1
ELSE
    Content 2
ENDIF
```

Description:

- DEFINE variables and EQU defined variable name can be the same (they are irrelevant)
- When event is established, content 1 will be executed; otherwise content 2 will be implemented instead.
- When event is established, it can be "TRUE" or "1".
- Event can be the variable of DEFINE
- Can be used in MACRO

Restriction : Not allow IF NEST

Ex:

```
Define    FSR0 = 0
Define    FSR1 = 1
Define    H08B = 0

MOVLF    MACRO    RAMLabel, FSR
         MVL    RAMLabel
         IF    FSR = 0
             MVL    FSR0L, F, ACCE
         ELSE
             MVL    FSR1L, F, ACCE
             ENDIF
         ENDM

             ORG    0
             JMP    BEGIN
             .....
             .....
BEGIN:    MOVLF    TEMP, FSR0
             .....
             .....
FSR0L    EQU    010h
FSR0H    EQU    012h
```

```

W          EQU    0
F          EQU    1
ACCE      EQU    0
BANK      EQU    1
TEMP      EQU    080h
TEMP1     EQU    081h
    
```

After assembly:

```

          ORG    0
          JMP    BEGIN
          .....
          .....
BEGIN:
          MVL    TEMP
          MVF    FSR0L, F, ACCE
          .....
          .....
    
```

## 4. Object Code

- Program code is saved as binary format.
- Using pseudo instruction, Global to generate function for main program quotation.
- Using pseudo instruction, Extern to quote external argument or program.
- Main program uses Include way to contain Object Code.
- Using MAP file to determine what functions can be quoted.

=====

Memory, Register and Const

=====

Name	Location	Address
	----	-----
INDF0	Const	0x0000
POINC0	Const	0x0001
PODEC0	Const	0x0002

=====

Local Program Address

=====

Name	Location	Address
	----	-----
RESET	Program	0x0000
MAINLOOP	Program	0x000C
MAINLOOP_DATA	Program	0x0025
MAINLOOPAA	Program	0x0100
TEST	Program	0x0265

=====

Extern Program Address

=====

Name	Location	Address
		----
		-----
		-----
	CLEAR	Program 0x0102
	MULTIPLY	Program 0x0106
	DIVIDE45	Program 0x0128
	DIVIDE	Program 0x0145
HTOB	Program	0x0186
	BTOH	Program 0x01B9
	SQRT	Program 0x01F0
	TT1	Program 0x025F
	TEMM	Program 0x025C

## 5. Differences of Assembling and Assembling && debugging

### Assembling

1. If there is a declaration of external argument (EXTERN), and it has been referred to, assembling is successful.
2. Assembling cannot judge whether RAM or Program has exceeded IC specification range. Thus, every IC can be assembled.

### Assembling && debugging

1. Assembling cannot be implemented if external argument (EXTERN) is referred in the program. Error message will be displayed at this time.
2. Assembling will check whether RAM or Program has exceeded IC specification range. If Program Size or the reference RAM does not exist, error message will be displayed.

## 6. Error Message

1. ERROR[Code 0] : Unknown error  
Instruction name error  
Instruction name error or the name has not been defined in Macro
2. ERROR[Code 1]: Couldn't open source file  
Can not find the Source file
3. ERROR[Code 2]: cannot open file  
Include file name error or file name missing
4. ERROR[Code 3]: Temp file creation error  
Temporary file open error.  
Repeat opening file
5. ERROR[Code 4]: Duplicate label or redefining symbol  
Repeat definition, Symbol or Label

6. ERROR[Code 5]: Undefined Symbol  
Undefined Symbol or Label
7. ERROR[Code 6]: Out of memory...  
Register, Memory or Label definition exceeds 8192, or Macro exceeds 4096...
8. ERROR[Code 8]: Illegal Symbol  
Register, Memory or Label define name and instruction name is the same
9. ERROR[Code 9]: Illegal argument  
LBSR argument greater than 16
10. ERROR[Code 10]: Overwriting previous address contents  
Reduplicated Address  
Ex:

```
ORG    0
Nop
Nop
Nop
Nop
JMP    Start
ORG    0004H
```
11. ERROR[Code 11]: Addresses above ....  
Program exceeds ROM Size
12. ERROR[Code 13]: Missing argument...  
Insufficient arguments behind the instruction  
Ex:

```
MVF   080h
```
13. ERROR[Code 14]: Too many arguments...  
Too many arguments behind the instruction  
Ex:

```
CLRF  080h, 1, 0
```
14. ERROR[Code 15]: Undeclared variable  
Undeclared variable
15. ERROR[Code 16]: Illegal Macro...  
Illegal Macro
16. ERROR[Code 19]: Macro nested too deep...  
Macro referenced Marco nest is more than 32 layers
17. ERROR[Code 20]: Circular file reference...  
Repeat quoted file, or there is no filename after INCLUDE
18. ERROR[Code 21]: Circular macro reference...  
Repeat definition, Macro
19. ERROR[Code 23]: Over Range  
Exceeding RJ or RCALL range
20. ERROR[Code 24]: Illegally Memory or Register  
Defining the non-existing register or memory
21. ERROR[Code 25]: Illegally Bit

Defining the non-existing bit

Ex:

```
INTF1 - ADCIF - - TMAIF WDTIF E1IF E0IF
```

```
BSF INTF1, 7, 0
```

22. ERROR[Code 26]: Memory Over  
Definition exceeds memory range

## 7. MODIFICATION RECORD

The following is the important modification of the document, but it does not include the changes of the punctuation and character form.

---

Document version	Page	Remark
V01	All	First version
V02	4	Add include application notes