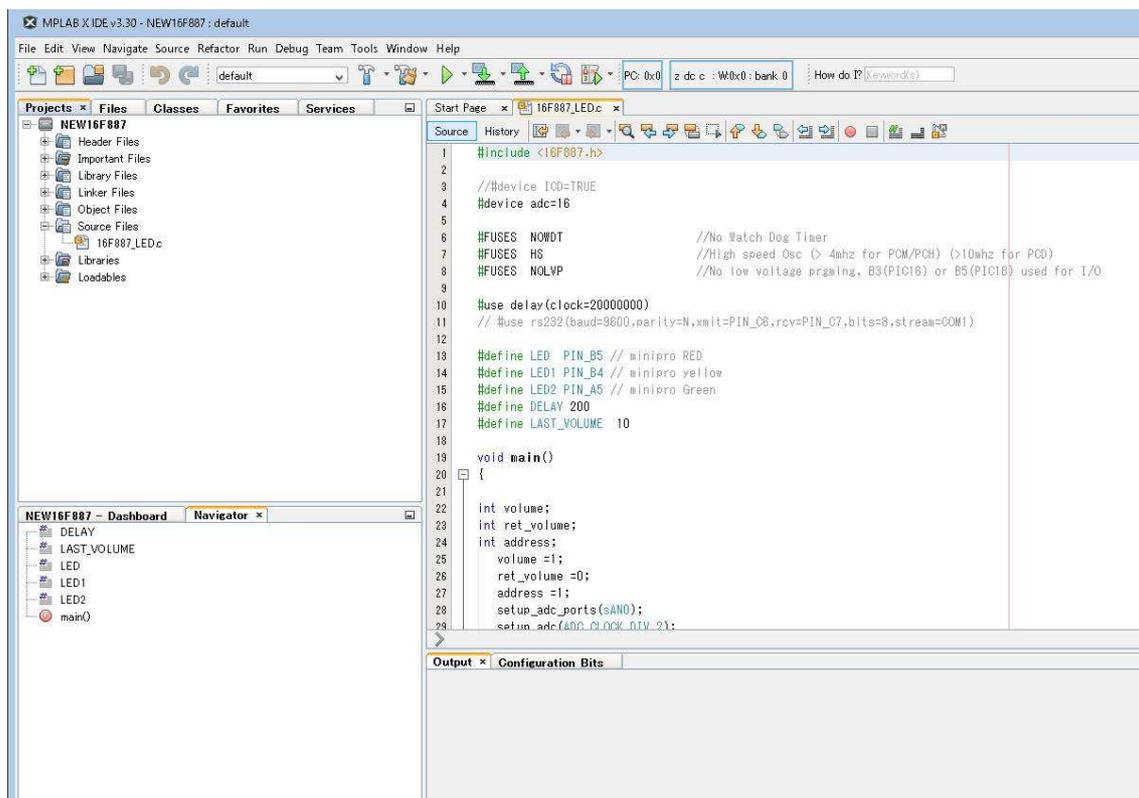


MPLAB-X の CCS インストールと組み込み方法

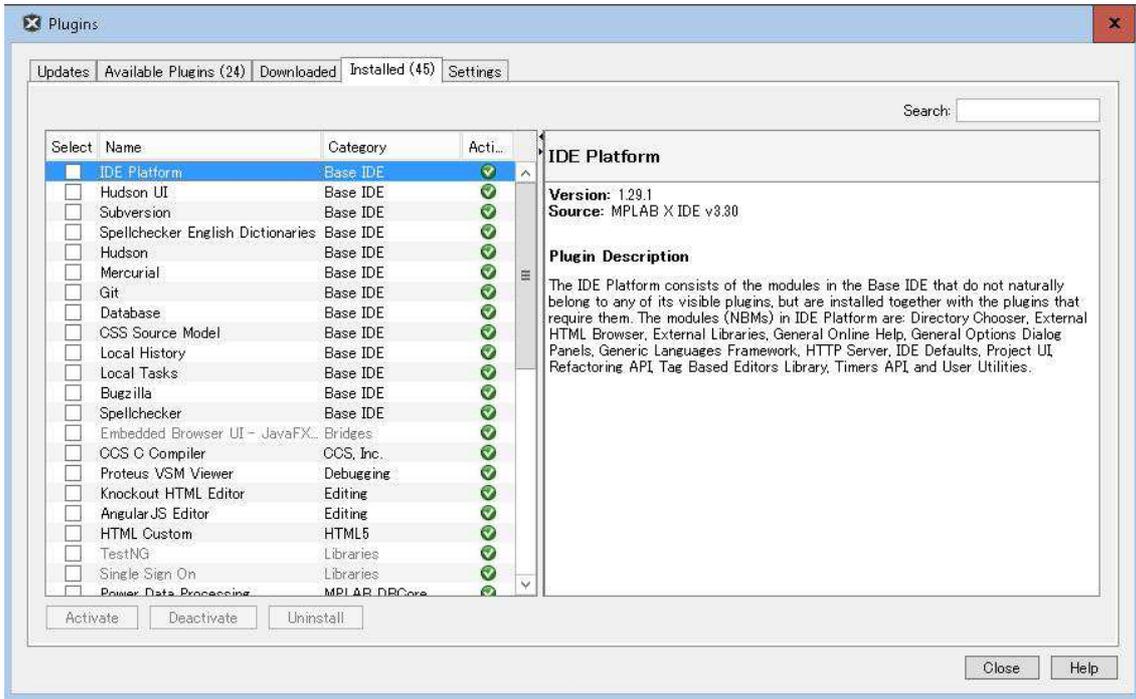
PICKIT3 等のライターを使用する場合は Device のプログラムは `ccsload` ではできませんので MPLAB-X を用いてください。

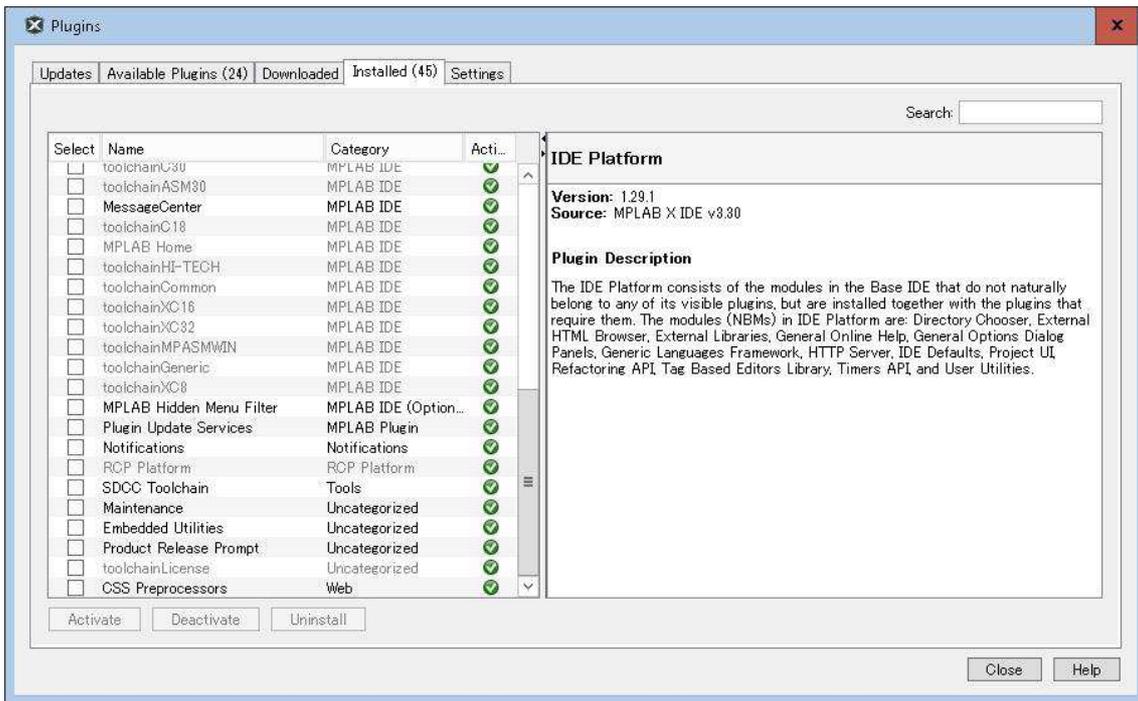
PCWH を用いる場合は書き込みとデバッグは可能です。

この場合 PICKIT3 はあらかじめ MPLAB-X で F/W を最新にしてください。

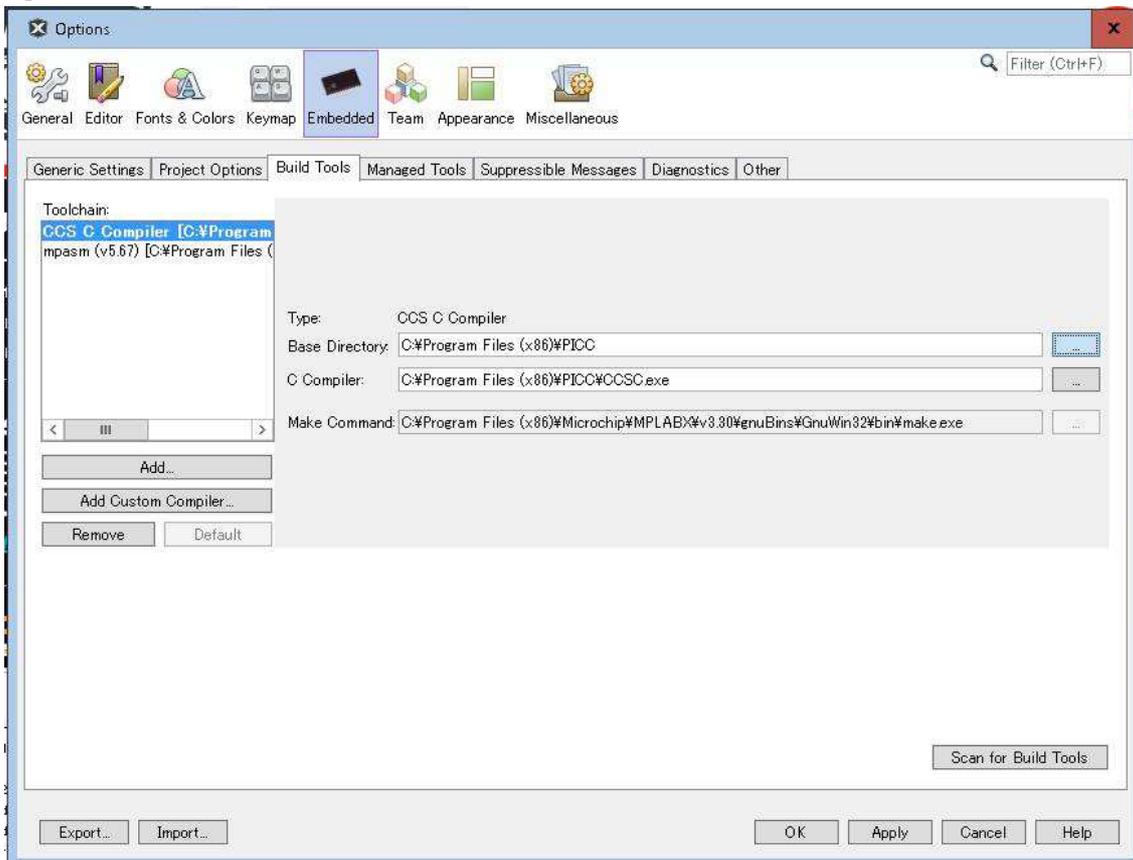


以下を参考に Plugin の確認をし、必要に応じて install ください

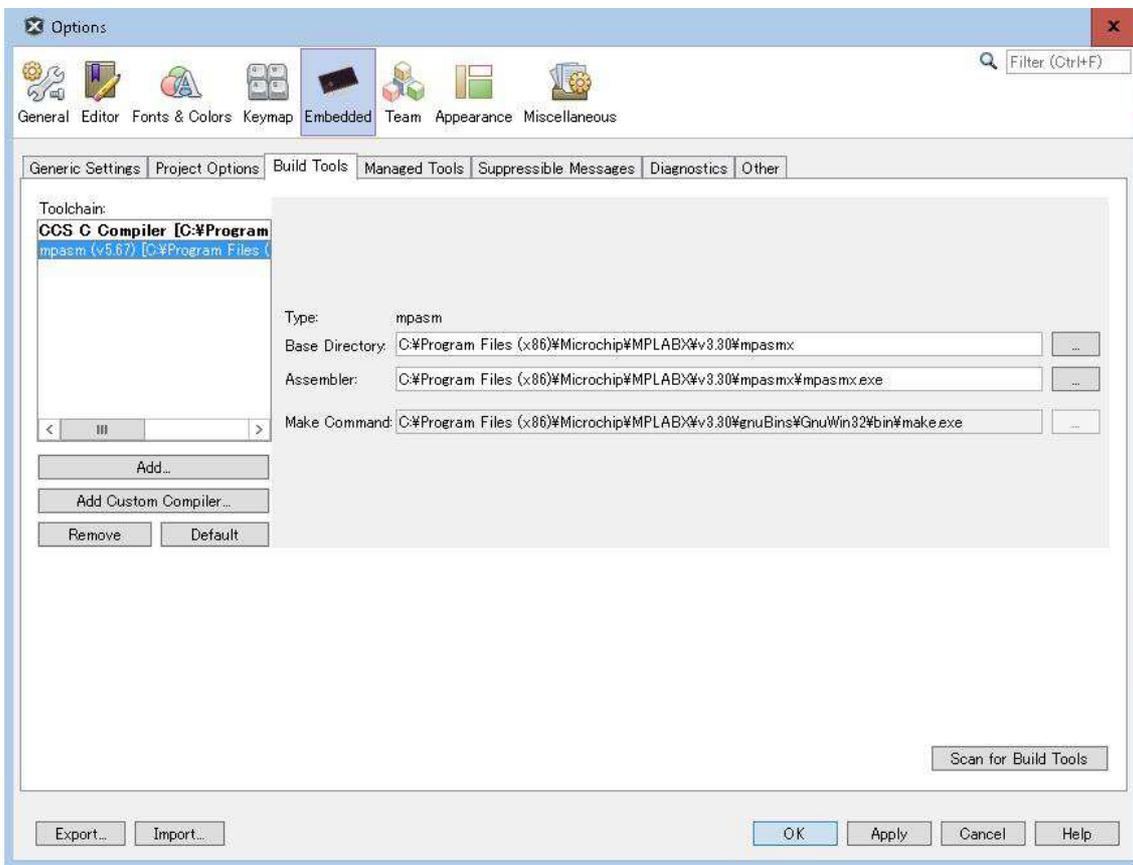




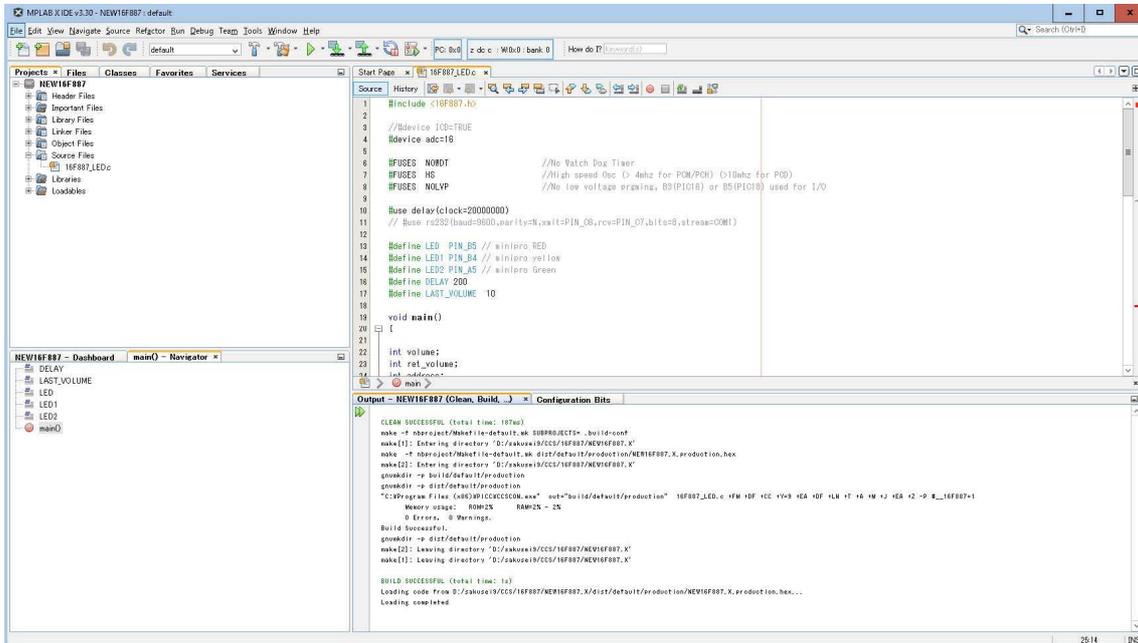
Options でディレクトリを CCS のコンパイラがある位置に変更ください。



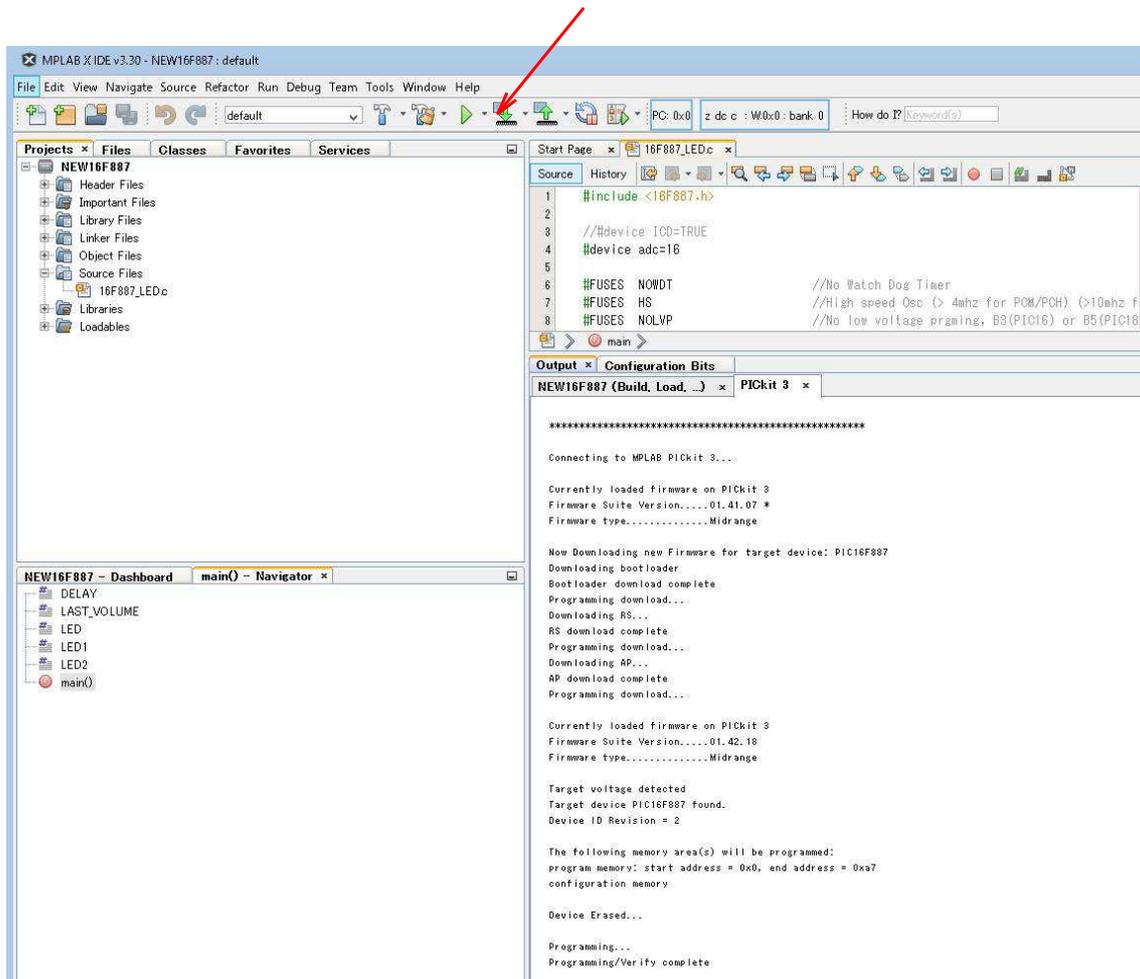
アセンブラは初期値で以下となっています。そのまま結構です



プロジェクトを作成しソースファイルを指定し、
Run - Clean and Build Main Project で以下の様に CCS コンパイラを呼び出し
コンパイルが完了します。



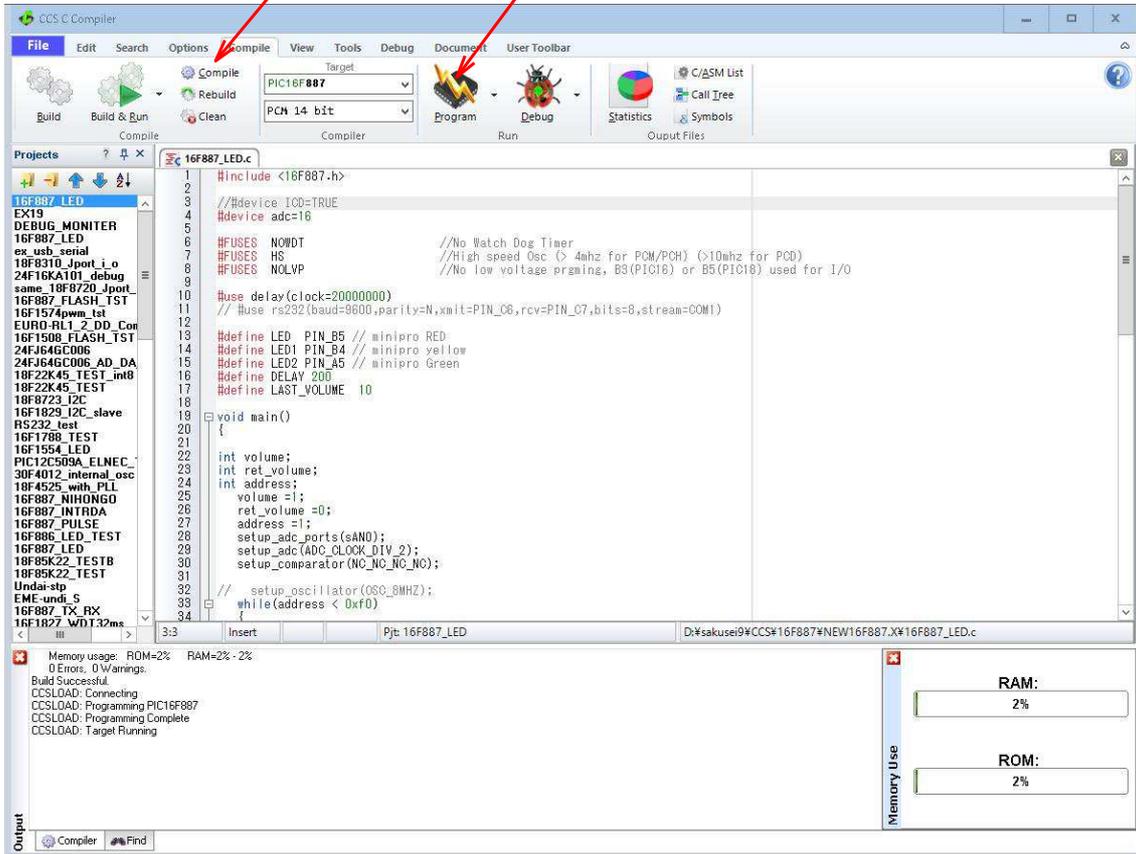
Programming は以下で行います



MPLAB-X はクローズしてください。

PCWH(D)を立ち上げます

PICKIT3 を用いています。 コンパイルしプログラムします。



デバッグの場合は以下の設定となります。

The screenshot shows the CCS C Compiler IDE with the following configuration for debugging:

- Target:** PIC16F887
- Compiler:** PCN 14 bit
- Code:** `#include <16F887.h>`
`#device ICD=TRUE`
`#device adc=18`
`#FUSES NOWDT //No Watch Dog Timer`
`#FUSES HS //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PIC16C7X/7E)`
`#FUSES NOLVP //No low voltage prgming, B3(PIC16) or B5(PIC18) for PIC16C7X/7E`
`#use delay(clock=20000000)`
`// #use rs232(baud=9600,parity=N,xmit=PIN_C8,rcv=PIN_C7,bits=8,stream=COM1)`
`#define LED_PIN_B5 // minipro RED`
`#define LED1_PIN_B4 // minipro yellow`
`#define LED2_PIN_A5 // minipro Green`
`#define DELAY 200`
`#define LAST_VOLUME 10`
`void main()`
`{`
`int volume;`
`int ret_volume;`
`int address;`
`volume =1;`
`ret_volume =0;`
`address =1;`
`setup_adc_ports(sANO);`
`setup_adc(ADC_CLOCK_DIV_2);`
`setup_comparator(NC_NC_NC_NC);`
`// setup_oscillator(OSC_8MHZ);`
`while(address < 0xf0)`
`{`
`output_low(LED);`
`delay_ms(DELAY);`
`output_high(LED);`
`delay_ms(DELAY);`
`output_low(LED1);`
`delay_ms(DELAY);`
`output_high(LED1);`
`delay_ms(DELAY);`
`output_low(LED2);`
`delay_ms(DELAY);`
`output_high(LED2);`
`delay_ms(DELAY);`
`}`
`}`

The **Debug** window configuration is as follows:

RAM	ROM	Data EE	Breaks	Stack	Watches	Peripherals
Eval	Monitor	Break Log	RTOS Tasks	SFR	Debug Configure	

Debugger: MPLABX - PICKIT 3

Compile Reload	True
Mouse over eval	True
Timeout Mouse over	True
Mouse over radix	Default
Monitor enabled	False
Echo on Monitor	True
Monitor Font Size	9
Processor	PIC16F887
Port Type	USB
Port Number	0

When TRUE the target will be reloaded after every compile

Buttons: Apply, Cancel

Status bar: PC=0070 W=00 Ready MCU at 0.00 MHz

ボードのスタンドアロン動作のためには忘れずに ICD=TRUE をコメントアウトして
コンパイルしておきましょう。

The screenshot displays the CCS C Compiler interface. The main window shows the source code for `16F887_LED.c`. The code includes headers, defines device and ADC settings, sets fuses, and defines LED pins. The `main` function initializes variables and enters a `while` loop that toggles an LED.

```
1 #include <16F887.h>
2
3 //device ICD=TRUE
4 #device adc=16
5
6 #FUSES NOWDT           //No Watch Dog Timer
7 #FUSES HS             //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
8 #FUSES NOLVP         //No low voltage prgming, B3(PIC18) or B5(PIC18) used for I/O
9
10 #use delay(clock=2000000)
11 // #use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,stream=COM1)
12
13 #define LED_PIN_B5 // minipro RED
14 #define LED1_PIN_B4 // minipro yellow
15 #define LED2_PIN_A5 // minipro Green
16 #define DELAY 200
17 #define LAST_VOLUME 10
18
19 void main()
20 {
21
22     int volume;
23     int ret_volume;
24     int address;
25     volume =1;
26     ret_volume =0;
27     address =1;
28     setup_adc_ports(sANO);
29     setup_adc(ADC_CLOCK_DIV_2);
30     setup_comparator(NC_NC_NC_NC);
31
32     // setup oscillator(OSC_8MHZ);
33     while(address < 0xf0)
34     {
```

The output window at the bottom shows the following information:

- Memory usage: ROM=2%, RAM=2%-2%
- 0 Errors, 0 Warnings
- Build Successful.

On the right side of the output window, there are two progress indicators for memory usage:

- RAM: 2%
- ROM: 2%