注意: この日本語版文書は参考資料としてご利用ください。
最新情報は必ずオリジナルの英語版をご参照願います。

MPLAB X IDEユーザガイド

MPLAB® X IDEユーザガイド

お客様へのご注意

重要:
どのような文書でも内容は時間が経つにつれ古くなります。本書も例外ではありません。Microchip社の製品は、お客様のニーズを満たすために常に改良を重ねており、実際のダイアログやツールが本書の内容とは異なる場合があります。最新文書はMicrochip社のウェブサイト(www.microchip.com)をご覧ください。

文書は「DS」番号によって識別します。この識別番号は各ページのフッタのページ番号の前に表記しています。DS番号「DSXXXXXA」の「XXXXX」は文書番号、「A」は文書のリビジョンレベルを表します。

開発ツールの最新情報はMPLAB® X IDEのオンラインヘルプでご覧になれます。[Help]メニューから[Help Content]を選択すると、オンラインヘルプ ファイルのリストが表示されます。

目次

1. MPLAB X IDEとは

MPLAB® X IDEは、Microchip社製マイクロコントローラ(MCU)とデジタルシグナル コントローラ(DSC)向けのアプリケーション開発用ソフトウェアです。

この開発ツールは統合開発環境(IDE)と呼ばれ、組み込みマイクロコントローラのコード開発向けに1つに統合された「環境」を提供します。MPLAB X IDEは組み込み回路設計の案出、設定、開発、デバッグ、認証に役立つ強力なツールを備えています。MPLAB X IDEはソフトウェアとツールのMPLAB開発エコシステムとシームレスに連携し、多くは無償で提供しています。

1.1 組み込みシステムの概要

一般的に組み込みシステムとは、Microchip社のPIC®やAVR®マイクロコントローラ(MCU)等の小型MCUの機能を利用したシステムを指します。このようなMCUは、マイクロプロセッサ ユニット(PCのCPUに相当)と「周辺モジュール」と呼ぶ付加回路で構成され、さらに、外部デバイスの数を減らすために、小規模の制御モジュール回路が同一チップ上に追加されています。このように1つのチップに集積されたデバイスを電子/機械装置に組み込む事により、低コストのデジタル制御が実現します。

1.1.1 組み込みコントローラとPCの違い

組み込みコントローラは、1つまたは複数の限定されたタスクだけを実行するという点で、PCとは大きく異なります。PCは不特定のプログラムを実行でき、各種の外部デバイスへ接続できます。組み込みコントローラは1つのプログラムだけを実行するため、そのタスクの実行に必要な処理能力とハードウェアだけを実装するだけで済み、結果として価格を低く抑える事ができます。

対してPCは、比較的高価な汎用中央演算装置(CPU)を核とし、多数の外部デバイス (メモリ、ディスクドライブ、ビデオ コントローラ、ネットワーク インターフェイス回路等)を備える必要があります。組み込みシステムは演算装置として低コストMCUを使い、同一チップに各種の周辺回路を含める事により、比較的少数の外部デバイスしか必要としません。

多くの場合、組み込みシステムは各種製品(コードレスドリル、冷蔵庫、車庫用のオートシャッター等)の内部製品またはサブモジュールとして使われます。このようなコントローラは、製品の全体機能の中のごく限られた部分だけに関与します。また、コントローラはデバイス内の重要サブシステムの一部に低コストのインテリジェンス機能を付加します。

組み込みシステムの例として煙感知器があります。煙感知器はセンサの信号を評価し、煙の発生を示す信号を検出すると警報を鳴らします。煙感知器が内蔵する小規模のプログラムは、無限ループを実行して煙センサの信号をサンプリングするか、あるいは普段は低消費電力の「スリープ」モードで待機し、センサからの信号によって復帰します。復帰したプログラムは、その時点で警報を鳴らします。通常、このようなプログラムは動作テストや電池残量アラーム等の機能も備えています。

センサとオーディオ出力を備えたPCにも同様の機能を持たせる事は可能ですが、9 Vのバッテリ1個だけで何年間も無人稼動させる事はできず、低コストにはなりません。煙感知器、カメラ、携帯電話、家電製品、自動車、スマートカード、セキュリティ システム等の日用品へのインテリジェンス機能の組み込みには、多くの場合安価なMCUが使われます。

1.1.2 MCUの構成要素

MCUは、プログラムを実行するためのファームウェアを格納するプログラムメモリ(図1-1、図1-2)またはコード化された命令(図1-3、図1-4)を内蔵しています。プログラムメモリのアドレス (リセットおよび割り込みアドレスを含む)指定にはプログラム カウンタ(PC)を使います。コード内のコールおよびリターン命令ではハードウェア スタックを使います。このスタックはプログラムメモリと連携して動作しますが、プログラムメモリの一部ではありません。プログラムメモリの動作の詳細(ベクタ、スタック等)はデバイス データシートに記載しています。

図1-1　PIC® MCUのデータシート – プログラムメモリとスタック

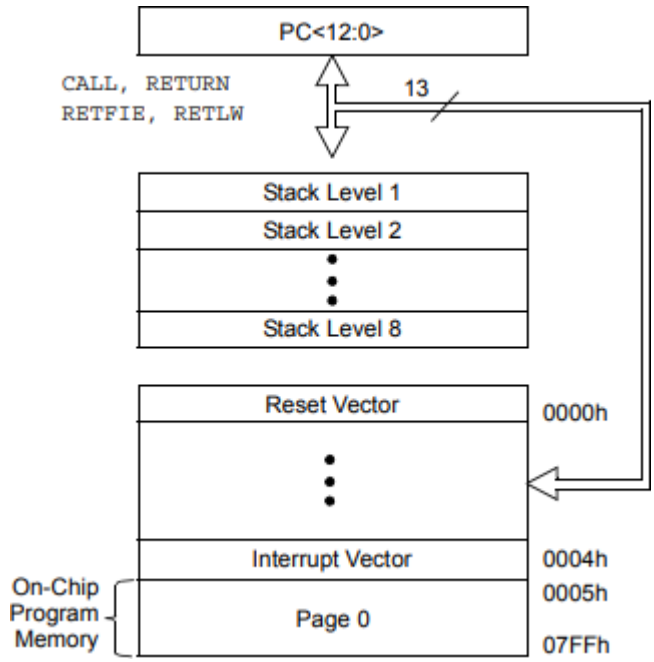


図1-2　AVR® MCUのデータシート – プログラムメモリ

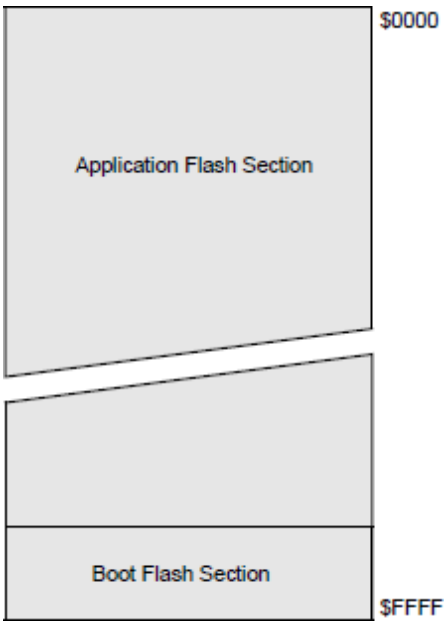


図1-3　PIC® MCUのデータシート – 命令セット(抜粋)

RRF	Rotate Right f through Carry
Syntax:	[*label*]　RRF　f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.

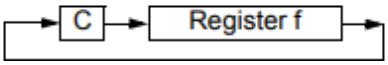


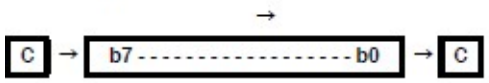
図1-4　AVR® MCUの命令セット(抜粋)

ROR – Rotate Right through Carry

Description:

Shifts all bits in Rd one place to the right. The C Flag is shifted into bit 7 of Rd. Bit 0 is shifted into the C Flag. This operation, combined with ASR, effectively divides multi-byte signed values by two. Combined with LSR it effectively divides multi-byte unsigned values by two. The Carry Flag can be used to round the result.

Operation:



	Syntax:	Operands:	Program Counter:
(i)	ROR Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	0111

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	⇔

S: N ⊕ V, For signed tests.

V: N ⊕ C (For N and C after the shift)

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; cleared otherwise.

C: Rd0
Set if, before the shift, the LSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
            lsr   r19          ; Divide r19:r18 by two
            ror   r18          ; r19:r18 is an unsigned two-byte integer
            brcc  zeroenc1     ; Branch if carry cleared
            asr   r17          ; Divide r17:r16 by two
            ror   r16          ; r17:r16 is a signed two-byte integer
            brcc  zeroenc2     ; Branch if carry cleared
            ...
zeroenc1:   nop                ; Branch destination (do nothing)
            ...

zeroenc1:   nop                ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

MCUはデータ(またはファイルレジスタ) メモリも内蔵しています。このメモリは特殊機能レジスタ(SFR)と汎用レジスタ(GPR)で構成されます。SFRはデバイスの動作を制御するためにCPUと周辺機能が使うレジスタです。GPRはプログラムの計算に必要な変数等の一時的なデータの保存用に使います。MCUにはデータEEPROMメモリを内蔵するものもあります。プログラムメモリと同様に、データメモリの使い方と動作の詳細もデバイス データシートに記載しています。

表1-1PIC® MCUのデータシート – ファイルレジスタの例

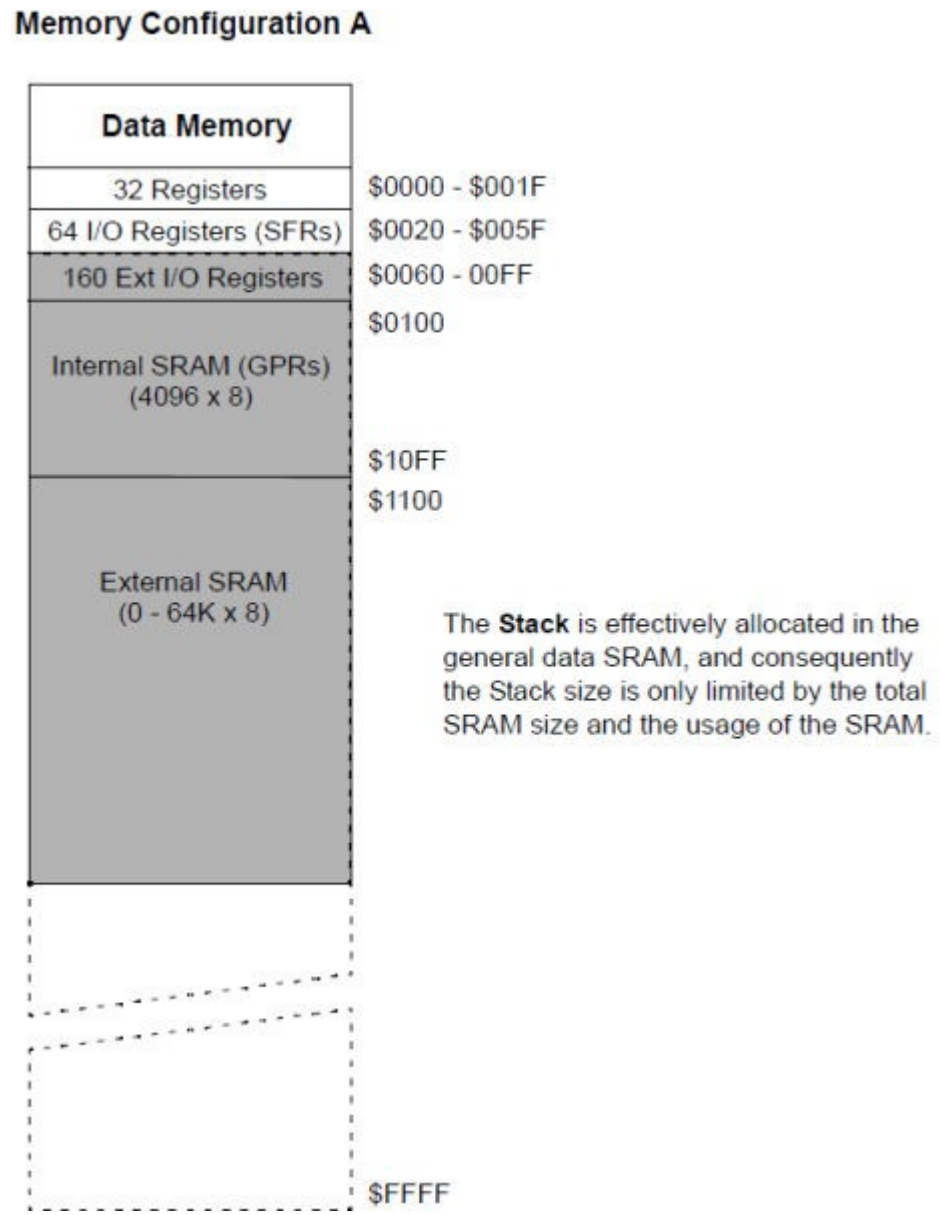
Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr. (1)	00h	Indirect addr.(1)	80h	Indirect addr. (1)	100h	Indirect addr.(1)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h:	PCL	182h:
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h:	FSR	84h:	FSR	104h	FSR	184h:
PORTA	05h:	TRISA	85h	PORTA	105h:	TRISA	185h
PORTB	06h:	TRISB	86h:	PORTB	106h:	TRISB	186h:
PORTC	07h	TRISC	87h:	PORTC	107h	TRISC	187h:
	08h		88h		108h		188h
	09h		89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah:
INTCON	0Bh	INTCON	8Bh:	INTCON	10Bh:	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDAT	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2(1)	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh:		18Eh
TMR1H	0Fh	OSCCON	8Fh	EEADRH	10Fh		18Fh
T1CON	10h	OSCTUNE	90h:		110h		190h
TMR2	11h		91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD(2)	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h	WPUA	95h	WPUB	115h		195h
CCPR1H	16h	IOCA	96h	IOCB	116h		196h
CCP1CON	17h	WDTCON	97h		117h		197h
RCSTA	18h	TXSTA	98h	VRCON	118h		198h
TX1REG	19h	SPBRG	99h	CM1CON0	119h		199h
RC1REG	1Ah	SPBRGH	9Ah	CM2CON0	11Ah		19Ah
	1Bh	BAUDCTL	9Bh	CM2CON1	11Bh		19Bh
PWM1CON	1Ch		9Ch		11Ch		19Ch
ECCPAS	1Dh		9Dh		11Dh	PSTRCON	19Dh

(続き)

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
ADRESH	1Eh:	ADRESL	9Eh	ANSEL	11Eh :	SRCON	19Eh
ADCON0	1Fh	ADCON1	9Fh	ANSELH	11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h		1A0h
			EFh		16Fh		
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h-7Fh	1F0h
	7Fh		FFh		17Fh		1FFh

表内の空白セル: 未実装のデータメモリ領域、「0」として読み出し
Note 1: 物理レジスタではありません。
Note 2: アドレス93hは、特定条件においてSSPマスク(SSPMSK)にもアクセスします。

図1-5　AVR® MCUのデータシート – SRAMデータメモリとスタック



メモリ以外にも、MCUは各種周辺モジュールを内蔵しています。周辺モジュールは入出力(I/Oポート)を含みます。I/Oポートとは出力として使えるMCUのピンであり、HighまたはLowに駆動する事で信号の送信、ランプの点滅、スピーカの駆動等を実行できます。多くの場合これらのピンは双方向であり、入力として構成する事もできます。入力として使う場合、外部スイッチまたはセンサからの入力に対する応答、外部デバイスとの通信が可能です。

図1-6　PIC® MCUのデータシート – ブロック図(抜粋)

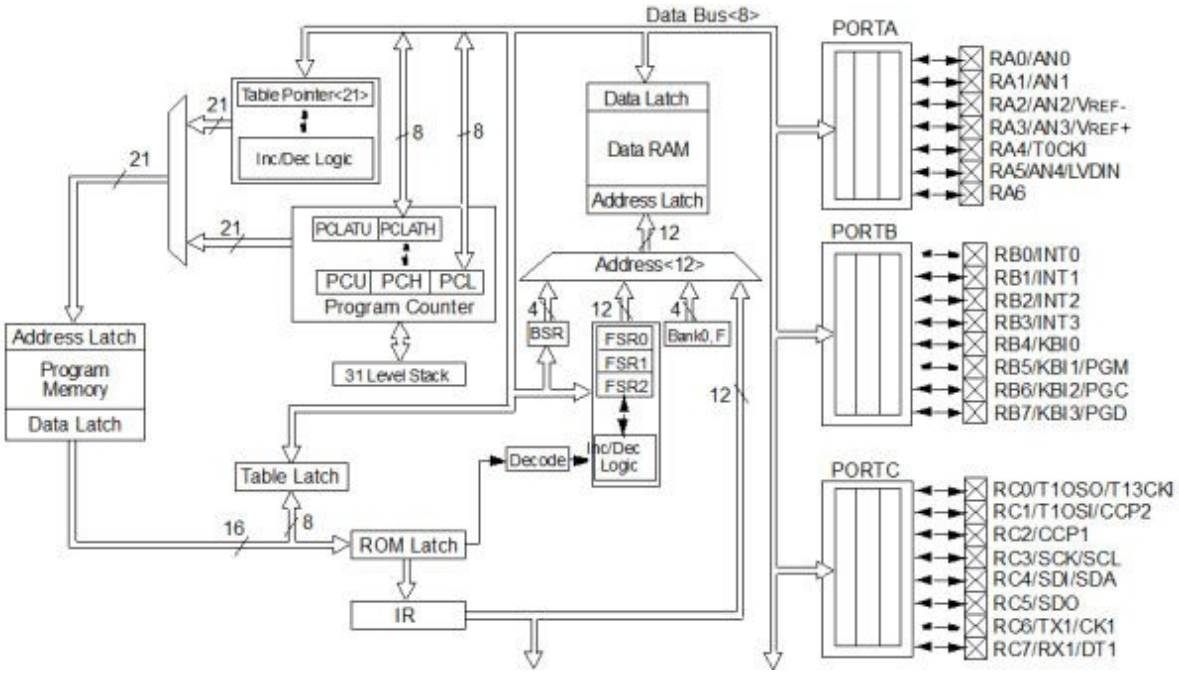
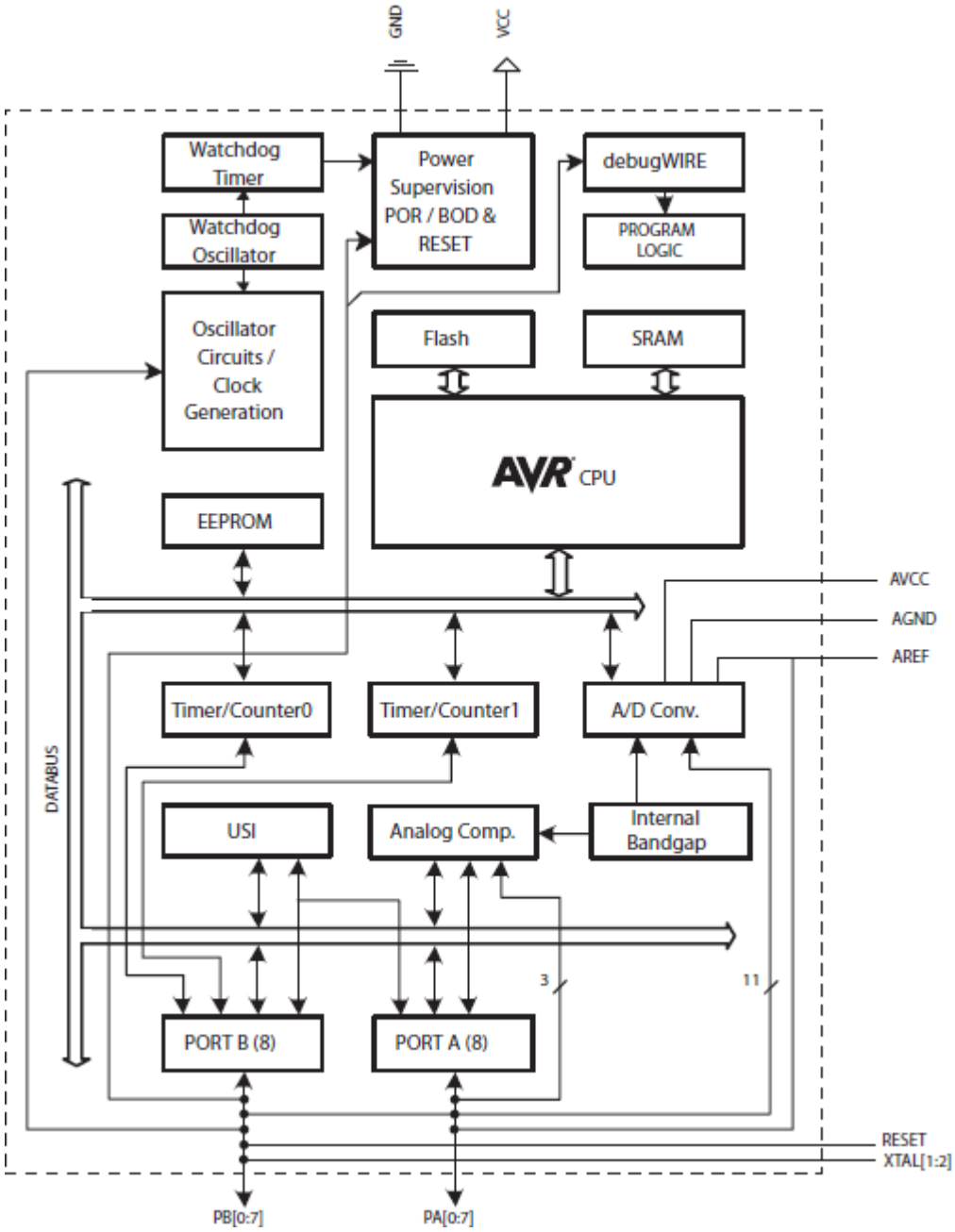


図1-7　AVR® MCUのデータシート – ブロック図(抜粋)



このようなシステムを設計する場合、アプリケーションに必要な周辺モジュールを選択します。

一般的な周辺モジュールは以下の通りです。

- A/Dコンバータ(ADC)を使うと、MCUに接続したセンサから電圧信号を取り込めます。
- シリアル通信モジュールを使うと、別のMCU、ローカル ネットワーク、インターネットと通信できます。
- PIC MCUのタイマモジュールを使うと、信号イベントの正確な計測、通信用信号の生成と受信、正確な波形の生成が可能な他、電源の瞬断やハードウェアの誤作動によるハングアップやフリーズが発生した場合に、MCUを自動的にリセットできます。

- その他に、外部電源が危険レベルまで低下した事を検出し、電力供給が完全に途絶える前に重要な情報を保存し、MCUを安全にシャットダウンするための周辺モジュールもあります。

使うべきPIC MCUは、アプリケーションが必要とする周辺モジュールとメモリ容量によってほぼ決まります。その他の選択基準としては、MCUの消費電力とサイズが挙げられます。

図1-8　PIC® MCUのパッケージ例

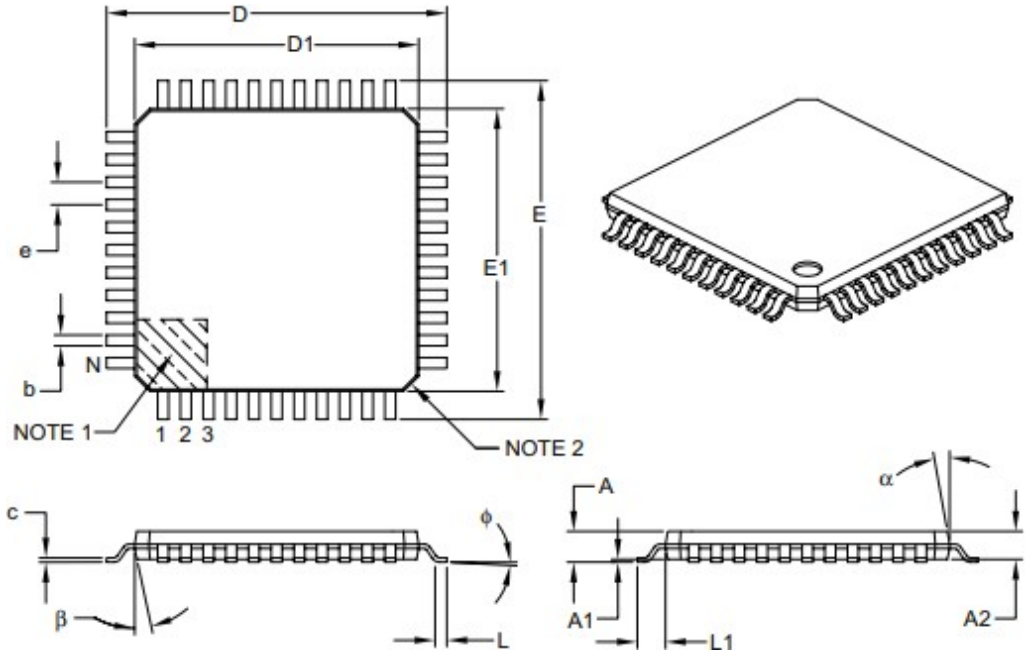
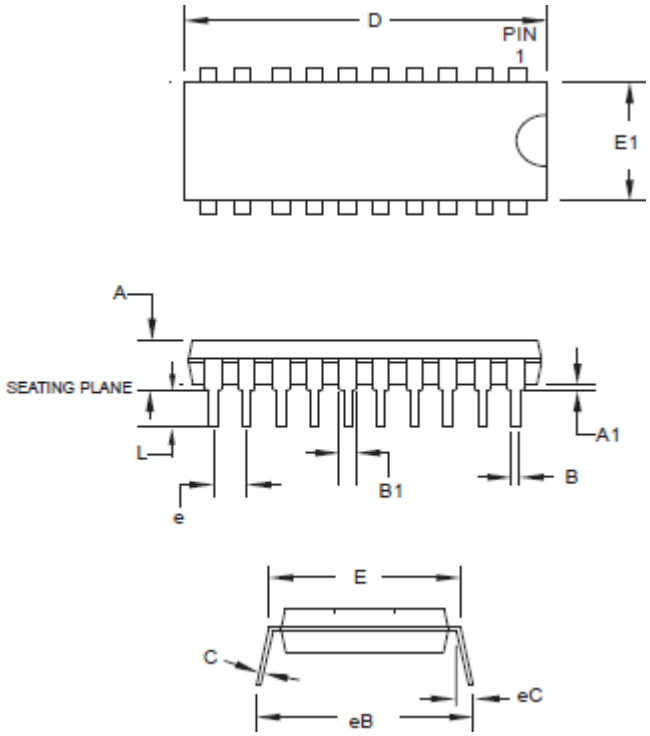


図1-9　AVR® MCUのパッケージ例



電力が供給され、オシレータがクロック信号を生成し始めるとMCUはアクティブになります。MCUによっては、複数の内部または外部オシレータ動作モードを備えているものがあります。

図1-10　PIC® MCUのデータシート – タイミング図(抜粋)

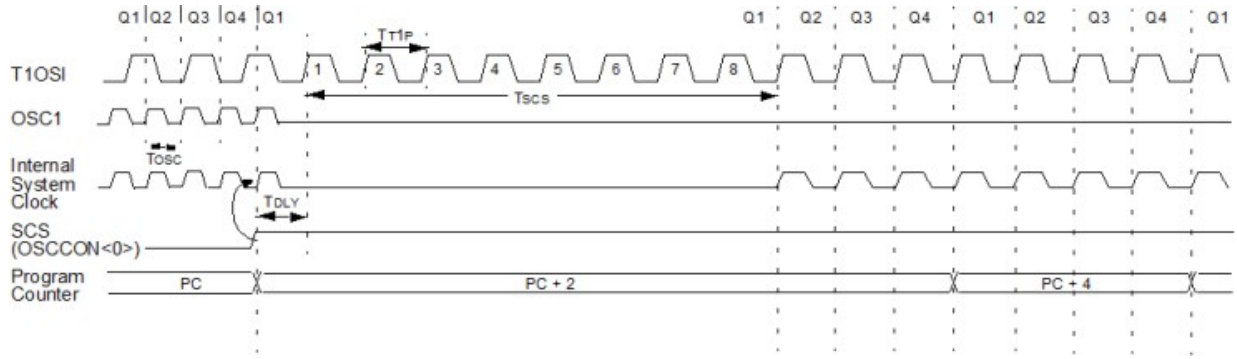
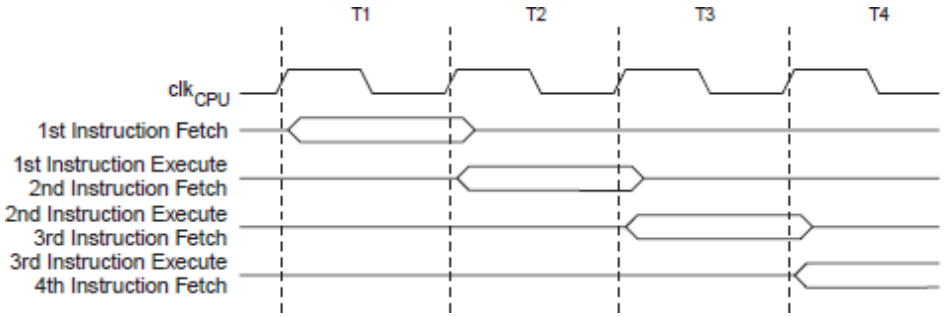


図1-11　AVR® MCUのデータシート – タイミング図(抜粋)



1.1.3 MPLAB X IDEによる組み込みシステム回路の実装

組み込みコントローラ用開発システムはPC上で動作し、組み込みシステム アプリケーションのコーディング、編集、デバッグと、MCUへの書き込みを支援するソフトウェアです。MPLAB X IDEは、組み込みシステム アプリケーションの設計と実装に必要な要素を備えています。

組み込みコントローラ アプリケーションの開発における代表的な作業手順は以下の通りです。

1. まず全体的な回路構成を設計します。次に、必要な機能と性能に基づいて、そのアプリケーションに最適なMCUを選択し、関連するハードウェア回路を設計します。ハードウェアを制御する周辺モジュールとピンを決定したら、ファームウェア(組み込みアプリケーションのハードウェアを制御するためのソフトウェア)を作成します。これには、マシンコードに直接変換可能なアセンブラや、高級プログラミング言語(C言語、BASIC等)用のコンパイラを使います。アセンブラやコンパイラでは、関数や変数にプログラム内の参照先ルーチンや用途に対応した名前を付けてコードを読みやすくできます。また、コードを構造化する事により、良好な保守性が得られます。
2. アセンブラ、コンパイラ、リンカを使ってソフトウェアをコンパイル、アセンブル、リンクします。これにより、コードを1と0だけで記述したマシンコード(MCUが実行できるコード)に変換します。このマシンコードがMCUに書き込まれることになるファームウェアです。
3. 作成したコードをMCUに書き込む前に、動作をテストします。複雑なプログラムでは、最初から期待通りに動作する事は稀であり、正しく動作させるにはプログラムのバグを取り除く必要があります。デバッガを使うと、マシンコードの実行状態をソースコード内の関数名と変数名に対応させて観察できます。デバッグ作業では、プログラムのシングルステップ実行による変数値の確認や、その値を変更してルーチンの動作を確認する「what if」チェックが可能です。
4. 最後に、マシンコードをMCUに書き込んで、正しく動作する事を確認します。

上の手順はいずれも非常に複雑な作業になりかねません。このような作業では、設計そのものに専念できる環境が重要です。MPLAB X IDEを使う事により、煩わしい操作の習得に時間を費やさずに設計作業に集中できます。

ステップ1は設計段階ですが、設計上の重要な判断を下すために、MPLAB X IDEを回路とコードのモデル化に役立てる事ができます。

MPLAB X IDEは、特にステップ2～4で真価を発揮します。コーディング用のエディタは各種言語ツールに対応しており、コーディングを支援します。このエディタは、アセンブラとコンパイラのプログラミング構造を認識して、自動的にソースコードを色分け表示します。このため、構文の誤り等を容易に見つけ出せます。プロジェクト マネージャにより、アプリケーションで使う各種ファイル (ソースファイル、プロセッサ記述ヘッダファイル、ライブラリファイル)を容易に管理できます。コードをビルドする際に、コンパイラによるコードの最適化レベル(実行速度優先またはバイナリサイズ優先)、変数とプログラムデータのメモリ格納領域を設定できます。MCUのメモリをアプリケーション用に最大限に活用するために、「メモリモデル」を指定する事もできます。アプリケーションのビルド中に言語ツールでエラーが発生すると、エラー発生行が表示されます。その行をダブルクリックしてソースファイルを即座に修正できます。コード修正後にアプリケーションを再ビルドして、エラーがないか確認します。複雑なコードでは、複数のサブセクションを作成してテストする必要があるため、記述 - コンパイル - 修正のループを繰り返すのが普通です。MPLAB X IDEを使うと、このループを最短時間で終わらせて次のステップへ進む事ができます。

コードのビルドでエラーが発生しなくなったら、次はコードの動作をテストします。MPLAB X IDEは、コードのテストを支援するために、全てのMCUに対応するデバッガと無償ソフトウェア シミュレータを備えています。ソフトウェア シミュレータを使うと、ハードウェアが完成する前にMCUの動作をシミュレートしてコードをテストできます。シミュレータは外部信号に対するファームウェアの応答をモデル化します。このため、シミュレートした外部信号(スティミュラス)を入力する必要があります。シミュレータでは実行時間の計測、コードのシングルステップ実行による変数と周辺モジュールの観察、コードのトレースによるプログラム実行状態の詳細な記録等が可能です。

プロトタイプのハードウェアをアプリケーションに組み込める段階になると、インサーキット エミュレータやインサーキット デバッガ等のハードウェア デバッガを使えます。これらのデバッグツールは、フラッシュメモリ内蔵デバイスの多くが備える特殊な回路を使って、実際のアプリケーション上でコードをリアルタイムに実行します。これらのツールを使うと、プログラムの実行を停止/再開して、ターゲットMCUのプログラムメモリとデータメモリの内容を観察できるため、アプリケーションにMCUを組み込んだ状態でコードをテストできます。

アプリケーションが正常に動作したら、Microchip社のデバイスプログラマまたは開発用プログラマを使って、MCUにプログラムを書き込みます。これらのプログラマは、コードがデバイスに正しく書き込まれたかベリファイします。

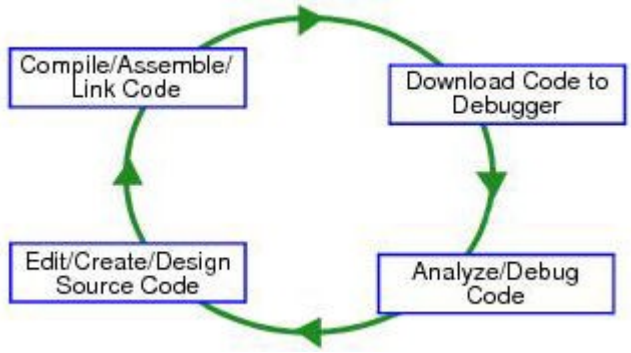
MPLAB X IDEはほとんどのPIC MCU、全てのdsPIC DSC、多くのAVRおよびSAM MCUをサポートしています。

1.2 開発サイクル

アプリケーション コードを作成するプロセスは、しばしば開発「サイクル」と呼ばれます。これは、設計から実装までの一連のステップを1回で問題なく完了できる事は稀なためです。正しく動作するアプリケーションを開発するには、コードの記述/テスト/修正を何度も繰り返す必要があります。

統合開発環境は、このような開発において複数のツールを何度も切り換える煩わしさからエンジニアを解放します。全ての機能を統合するMPLAB X IDEを使えば、ツールや動作モードの切り換えのたびに作業を中断する必要がなくなるため、アプリケーション開発に集中できます。

図1-12　設計サイクル



MPLAB X IDEは、共通のグラフィカル ユーザ インターフェイスを介して全てのツールを(通常は自動的に)連動させる「ラッパー」として機能します。例えば、ソースコードを記述し、これを実行可能なマシンコードに変換し、MCUに書き込み、その動作を確認するといった一連の作業を共通の環境内で実行できます。開発プロセスでは、コードを記述するためのエディタ、ファイルと設定を管理するためのプロジェクト マネージャ、ソースコードをマシンコードに変換するためのコンパイラまたはアセンブラ、MCUと接続またはMCUの動作をシミュレートするための各種ハードウェア/ソフトウェア等、複数のツールが必要です。

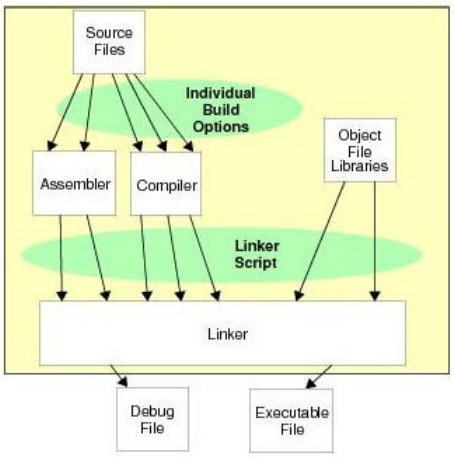
1.3 プロジェクト マネージャ

プロジェクト マネージャは、編集したファイルを他の関連ファイルと一緒にまとめて、アセンブルまたはコンパイル用の言語ツールに渡します。これらのファイルは最終的にリンカへ渡されます。

リンカは、アセンブラ/コンパイラ/ライブラリから受け取った複数のオブジェクト コードを組み込みコントローラのメモリ領域に適切に配置して、各モジュールが相互に機能できるようにリンクさせます。

このようなアセンブル/コンパイルからリンクまでのプロセス全体を、プロジェクトの「ビルド」と呼びます。必要に応じて、ファイルごとに異なる言語ツール向けのプロパティを呼び出す事ができます。このような場合でも、ビルドプロセスが全ての言語ツールの動作を統合します。

図1-13　MPLAB® X IDEのプロパティ マネージャ



ソースファイルは、アセンブラまたはコンパイラの規則に従って書かれたテキストファイルです。アセンブラとコンパイラは、これらのソースファイルを複数の中間的なマシンコード モジュールに変換します。この際、関数とデータを参照するためにプレースホルダも生成されます。

リンカはこれらのプレースホルダを解決し、全てのモジュールから実行可能なマシンコード ファイルを生成します。リンカはデバッグファイルも生成します。MPLAB X IDEは、このデバッグファイルに基づいて、実行用マシンコードを元のソースファイルに関連付けます。

コードの作成にはテキストエディタを使います。このエディタはテキストの構造を認識し、命令ニーモニック、C言語の構造体、コメント等の各種エレメントを色分け表示します。エディタは、ソースコードの作成に必要な一般的機能を備えています。コード作成後、エディタは他のツールと連携してデバッガ内でのコードの実行状態を表示します。エディタでブレークポイント (コードの実行を一時停止する位置)を設定してプログラムを実行し、変数名の上にマウスポインタを置くと変数値を表示できます。変数名をソーステキスト ウィンドウからウォッチ ウィンドウにドラッグ&ドロップすると、各ブレークポイントで停止した時またはコード実行中に変数値の変化を観察できます。

1.4　言語ツール

「言語ツール」とは、クロスアセンブラやクロスコンパイラ等のプログラムを指します。Visual BasicやCコンパイラのように、PC上で動作する言語ツールが一般的によく知られています。

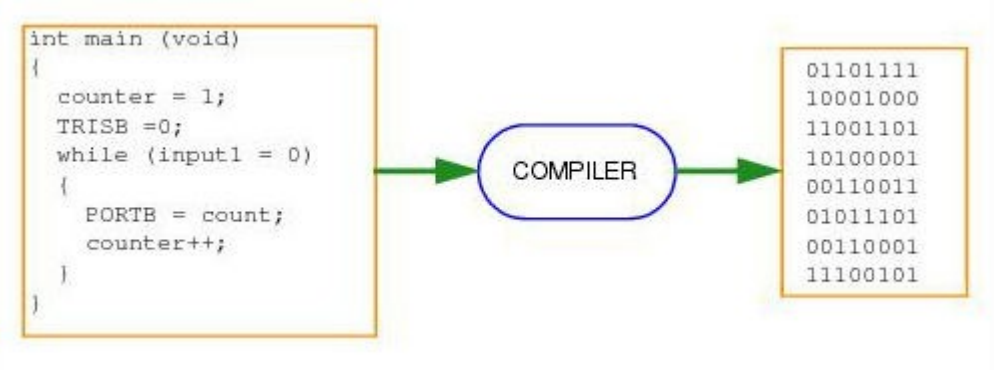
組み込みシステムでは「クロスアセンブラ」または「クロスコンパイラ」を言語ツールとして使います。これらのツールもPC上で動作しますが、そのPCではなく別のマイクロプロセッサ(またはMCU)上で動作するコードを生成します。

これらの言語ツールはデバッグファイルも生成します。MPLAB X IDEは、デバッグファイルを使ってマシン語命令とメモリ領域をソースコードに関連付けます。この関連付けにより、MPLAB X IDEエディタによるブレークポイントの設定とウォッチ ウィンドウによる変数の表示が可能となり、ソースコードをシングルステップ実行してアプリケーションの動作を観察できるようになります。

組み込みシステムの言語ツールは、コードサイズが非常に重要視されるという点でも、一般的なコンパイラと異なります。これは、コードサイズを小さくできればMCUのメモリサイズを小さくでき、それだけコストを削減できるからです。このため、ターゲット デバイスを熟知してコードを最適化するテクニックが求められます。

一般的なPC用プログラムの場合、ある程度複雑なプログラムになると、そのサイズは数MBに達します。これに対し、単純な組み込みシステムのプログラムであれば、1 KB以下に収める事ができます。比較的複雑な機能を実行する中規模の組み込みシステムになると、32Kまたは64Kのメモリサイズが必要になる場合もあります。組み込みシステムの中には、大規模なテーブル、ユーザテキスト メッセージ、データのロギングに何MBものストレージを使うものもあります。

図1-14　コンパイラによるソースコードからマシンコードへの変換



1.5　デバッグ

統合開発環境(IDE)では、コードの動作をテストするためにデバッガを使います。デバッガには、MCUの動作をPC上でシミュレートするソフトウェア プログラム (シミュレータ)と、実際のアプリケーション内でプログラムの動作を解析できるハードウェア デバイス (ハードウェア デバッガ)があります。

IDEとデバッグ

通常、プロジェクトの設計サイクルが最終段階に近づくと、デバッグは緊急の課題になります。製品化の最終ステップではアプリケーションを当初の設計通りに機能させる必要がありますが、多くの場合この段階が製品の完成に遅れを生じる最大の要因となります。このような製品開発において、統合開発環境が極めて重要な役割を果たします。コードの「微調整」、リコンパイル、ダウンロード、テストは全てそれなりに時間を要します。全て同じ環境内のツールを使う事で、サイクル時間を短縮できます。致命的なバグを全て取り除く必要がある最終ステップでは、組み込みエンジニアの腕が試されます。適切なツールはエンジニアを支援して開発期間を短縮します。MPLAB X IDEでは、各種ツールを共通のインターフェイスで使えるため、ソフトウェア シミュレータから低コストのインサーキット デバッガへ、さらには強力なインサーキット エミュレータへとツールをアップグレードする際にも習熟が容易です。

ソフトウェア デバッガ

MPLAB X IDEはシミュレータを内蔵しているため、ハードウェアを追加しなくても、PC上でプログラムをテストできます。シミュレータはソフトウェア デバッガとも呼ばれ、その機能はハードウェア デバッガとほぼ同じです。このためシミュレータの使い方を習得すれば、ハードウェア デバッガも容易に使えます。シミュレータはPCのCPUを使ってMCUの動作をシミュレートするため、通常は実際のMCUよりも低速で動作します。

ハードウェア デバッガ

MPLAB X IDEで使えるハードウェアには、プログラマとハードウェア デバッガの2種類があります。プログラマは、PC上で作成したマシンコードをMCUの内部メモリに書き込みます。これにより、アプリケーションに組み込んだMCU上でプログラムを実行できるようになります。

しかし、最初からコードが期待通りに機能する事はきわめて稀です。そこで実際のアプリケーションでプログラムの動作をよく観察して、期待通りの動作が得られるようにソースコードを修正する必要があります。このプロセスをデバッグと呼びます。既に説明した通り、シミュレータはPC上でコードの動作をテストできますが、MCUにプログラムを書き込んで実際のアプリケーションで動作させると、シミュレータでは見つからなかった問題が多数発生します。プログラマだけでコードを変更してMCUにプログラミングし直し、ターゲットにプラグインして再テストできますが、コードが複雑な場合、長い時間と労力を要するサイクルになる可能性があります。また、ハードウェア内の問題を正確に理解する事は困難です。

このような場合、ハードウェア デバッガが有効です。ハードウェア デバッガには、特殊なデバッグ用回路を内蔵したMCUを使うインサーキット エミュレータとインサーキット デバッガがあります。ハードウェア デバッガは、この特殊な回路を使う事によりシミュレータと同様にコード内の任意ステップでの変数の観察や、シングルステップ実行を可能にします。

1.6 デバイスのプログラミング

アプリケーションのデバッグが終わって、開発環境で完全に動作する事を確認できたら、今度は実際のデバイス上で動作をテストする必要があります。デバイスのプログラミングにはインサーキット エミュレータ、インサーキット デバッガ、開発用プログラマ、デバイス プログラマを使えます。MPLAB X IDをプログラマ機能向けに設定して、デバイスにプログラムを書き込む事ができます。これにより、アプリケーションを完成品とほぼ同じ状態で観察できるようになります。エンジニアリング プロトタイプ プログラマを使うと、プロトタイプを素早く作成して評価できます。アプリケーションによっては、デバイスをプリント基板にはんだ実装後でもプログラミングできます。In-Circuit Serial Programming™ (ICSP™)プログラミング機能を使うと、製造時にファームウェアを書き込み、さらに後から書き直す事もできます。インサーキット デバッグをサポートしているデバイスなら、完成品をインサーキット デバッガに接続して製造後の品質検査とファームウェアの改良開発が可能です。

量産時のプログラミングには量産プログラマとMPLAB IPEを使います。MPLAB IPEはMPLAB X IDEと共にインストールされます。

1.7 MPLAB X IDEの構成要素

MPLAB X IDEは以下の要素で構成されています。

- プログラマ用テキストエディタ: このエディタはデバッガ用のウィンドウとしても機能します。
- プロジェクト マネージャ (プロジェクト ウィンドウとして表示): IDEと言語ツール間の統合と通信を提供します。
- 各種アセンブラ/リンカ パッケージ: ターゲット デバイスのファームウェア開発に使います。
- デバッガエンジン: ブレークポイント、シングルステップ実行、ウォッチ ウィンドウ等、デバッガが備えるべき全ての機能を提供します。デバッガは、ソフトウェアおよびハードウェア デバッグツールと連携して機能します。
- ソフトウェア シミュレータ: 全てのPIC/dsPICデバイス、多くのAVR/SAMデバイスに対応します。シミュレータの実行ファイルは、デバイス別に用意しています。MPLAB X IDEは、それらのファイルの中から、ターゲット デバイスに適合するファイルを選択します。

以下のMPLAB X IDE用オプション コンポーネントもご利用またはご購入頂けます。

コンパイラ言語ツール

Microchip社のMPLAB XC Cコンパイラは、MCU向けに完全に統合および最適化したコードを提供します。コミュニティベース コンパイラ、サードパーティ コンパイラをMPLAB X IDEのプロジェクト マネージャから起動してコードをコンパイルする事もできます。コンパイルしたコードは自動的にターゲット デバッガに読み込まれ、テストと検証が可能です。

プログラミング環境/フレームワーク

MPLAB Code Configurator (MCC)は、無償のグラフィカルなプログラミング環境です。MCCで設計を行うと、シームレスで分かりやすいC言語コードが生成され、プロジェクトに挿入できます。[Tools] > [Plugins] > [Available Plugins]を選択すると、MCCプラグインをMPLAB X IDEにインストールできます。

MPLAB® Harmonyは、移植可能な周辺モジュール ライブラリをベースに構築されたシステムサービス、デバイスドライバ、その他ライブラリのフレームワークであり、組み込みPIC32アプリケーションの開発に使える柔軟性、移植性、一貫性のあるソフトウェア「構成要素」を提供します。MPLAB Harmony Configurator (MHC)は、MPLAB Harmony向けの便利なハードウェア設定ユーティリティであり、[Tools] > [Plugins] > [Available Plugins]からプラグインとして入手できます。

プログラマ

MPLAB ICD 4インサーキット デバッガ等の量産プログラマを使うと、量産環境でターゲット デバイスにコードをプログラミングできます。プログラマとMPLAB X IDEまたはMPLAB IPEを組み合わせて使う事で、コードとデータだけでなく、ターゲットMCUまたはDSCの各種動作モードを設定するコンフィグレーション ビットのプログラミングも制御できます。

インサーキット デバッガ/エミュレータ

インサーキット デバッガおよびエミュレータを使うと、ターゲット デバイス上でアプリケーション コードをデバッグできます。これらのツールは、一部のデバイスが内蔵する特殊なリソースを使って、アプリケーションに実装済みのターゲットMCUにコードをダウンロードし、ブレークポイントの設定、コードのシングルステップ実行、レジスタと変数の監視を可能とします。エミュレータでは、その他トレース等のデバッグ機能も利用できます。

プラグインツール

MPLAB X IDEに機能を追加するために、各種プラグインを利用できます。例えばMPLAB Data Visualizerを使うと、アプリケーションの出力をリアルタイムでグラフィカルに表示する事ができます。

1.8 MPLAB X IDEヘルプ

MPLAB X IDEは、NetBeansプラットフォームをベースにしています。従って、多くのNetBeans機能をMPLAB X IDEの機能として利用できます。NetBeansヘルプトピックの詳細は、以下を参照してください。

https://docs.oracle.com/cd/E50453_01/doc.80/e50452/toc.htm

ヘルプ

MPLAB X IDEの挙動を完全に理解するには、全てのヘルプファイルを参照してください。ヘルプを開くには[Help] > [Help Contents]を選択します。この操作により全ヘルプファイルが1つに結合されます。そのため開くのに少し時間がかかります。

ツールのヘルプ

[Help] > [Tool Help Contents]で選択したツールの個別のヘルプファイルを参照する事もできます。個別のヘルプファイルを開く事で、素早く範囲を絞って内容を検索できます。

開発者ヘルプ

Microchip Developer Helpは先進機能のヒントを探すのに便利です。

http://microchipdeveloper.com/

Developer Helpでトピックを検索するには[How do I?]の横のボックスにテキストを入力します。

図1-15 Microchip Developer Help

1.9 その他のMPLAB X IDE関連文書

ヘルプ以外の文書、ビデオとフォーラムへのリンク、Wikiは[Start Page]から利用できます。

図1-16　MPLAB X IDEの[Start Page]

1.10　Microchip社ウェブサイト

Microchip社は以下のウェブサイトで開発ツールのオンライン サポートを提供しています。

http://www.microchip.com/development-tools/

1.11　Microchip社オンラインストア

MPLAB X IDE v3.xx以降、開発ツール ソフトウェア/ハードウェア製品を入手するにはMicrochip社のオンラインストアにIDE経由でアクセスします。

	ストアのアイコン

- [MPLAB X Store]タブ – [Start Page]タブの隣がこのタブです。このタブは各種製品へのリンクを提供します。
- [Help] > [MPLAB X Store] – [Help]メニューの下に[MPLAB X Store]タブを開くリンクがあります。
- [MPLAB X Store]ツールバー アイコン – [MPLAB X IDE]ツールバー上のアイコンで[MPLAB X Store]タブが開きます。

1.12　プログラミング センター

Microchip社プログラミング センターを使うと、デバイスへのコードの書き込みまたはプログラミング済み製品の無償サンプル請求が可能です。以下のアイコンをクリックするとmicrochipDIRECTのサイトが開きます。

	プログラミング センターのアイコン

1.13 MPLAB X IDEの更新

MPLAB X IDEはユーザの皆様にお使い頂くだけでなく、Microchip社内でも新機能を採用した新型MCUの開発に使っています。MPLAB X IDEの新機能の多くは、お客様からのご要望とMicrochip社内での使用経験を基に開発しています。新設計のMCUのリリースが続く限り、MPLAB X IDEも進化し続けます。

新しいデバイスのサポートと新機能を追加するために、MPLAB X IDEのバージョンはほぼ数ヶ月に1回のペースで更新されます。

開発プロジェクトの途中でMPLAB X IDEの新バージョンがリリースされた場合、よほどの理由(例: 現在の作業を阻害しているバグの修正)がない限り、その時点での新バージョンへの更新は推奨しません。新しいプロジェクトの開始時に更新する事を推奨します。

新バージョンへの更新のたびに、MPLAB X IDEソフトウェアに新機能が追加されるため、マニュアル等の印刷文書の更新はオンラインヘルプよりも遅れます。このため、MPLAB X IDEで疑問が生じた場合、オンラインヘルプが最良の参照資料です。

MPLAB X IDEと関連要素の変更通知サービスをご希望のお客様は、以下のmyMICROCHIP Personalized Notification Serviceで開発ツールのセクションにご登録ください。

http://www.microchip.com/pcn

1.14 IDE外での作業手順

MPLAB X IDEは、組み込みシステム向けアプリケーションの書き込み、デバッグ、リリースを支援するように設計されています。しかし、IDE外でコード開発が必要な場合があります。

MPLAB X IDE外でコードをビルドする方法は、以下を参照してください。

http://microchipdeveloper.com/mplabx:work-outside

2. 使用前の準備

MPLAB X IDEを使うには以下の作業を行います。

- readme/リリースノートでのインストール要件の確認
- MPLAB X IDE(とJRE)のインストール
- ハードウェア ツール用USBドライバのインストール
- ターゲットへの接続
- 言語ツール (アセンブリ、コンパイラ)のインストール
- IDEの起動
- Microchip社オンラインストアでの購入

さらに、IDEの複数インスタンスの起動方法、複数バージョンの起動方法、起動パラメータの使い方を学習します。

2.1 インストール要件の確認

インストール要件とインストールに関する問題は「Readme for MPLAB X IDE」に記載されています。このHTMLファイルへは、デスクトップ上の[Learn & Discover] > [Getting Started] > [Users Guide & Release Notes]からアクセスできます。

2.2 JREとMPLAB X IDEのインストール

MPLAB X IDE (NetBeansプラットフォーム ベース)をインストールする際、Java Runtime Environment (JRE) 8もインストールされます。このJREはMPLAB X IDE専用であり、他の用途ではPCに登録されていません。

Windows XPの場合、JRE 7をインストールする必要があります。MPLAB X IDEインストーラはシステムでJRE 7を検索します。見つからなかった場合、推奨ダウンロードサイトを表示します。

Mac OS XにおけるJRE 8のサポートは10.8 (Mountain Lion)から始まりました。従って、MPLAB X IDEにはMac OS X 10.8以上が必要です。

2.3 ハードウェア ツール用USBデバイスドライバのインストール

MPLAB X IDEインストーラは、サポートされているハードウェア ツールのUSBデバイスドライバのプリインストーラを含んでいます。MPLAB X IDEをインストールした後でハードウェア ツールをPCに接続すると、オペレーティング システム(OS)がドライバとツールを関連付けます。しかし、一部のOSとツールでは正しい関連付けに追加ステップが必要です。

macOS®またはLinux®用USBドライバのインストール

MPLAB X IDEをMacまたはLinux PCにインストールする場合、追加ステップは不要です。

Windows®用USBドライバのインストール

MPLAB X IDEをWindows PCにインストールする場合、以下の手順を実行します。

Note: MPLAB IDE v8用のUSBハードウェア ツール ドライバは、MPLAB X IDE用とは異なります。

下表に、手順が必要なツールと不要なツールを示します。

手順が必要なツール	不要なツール
MPLAB REAL ICEインサーキット エミュレータ	PICkit 2またはPICkit 3インサーキット デバッガ
MPLAB ICD 3、MPLAB ICD 4、 MPLAB PICkit 4インサーキット デバッガ	その他のMPLABスタータキット
MPLAB PM3デバイス プログラマ	

(続き)	
手順が必要なツール	**不要なツール**
PIC32スタータキット	

以下のセクションに記載する手順を実行し、インストール方法を決定します。

2.3.1 MPLAB X IDEをインストールする前に

既存のWinUSBドライバのWindowsのバージョンがIDEプリインストーラのバージョンより古いと、システムドライバは新しいものに置き換えられます。既存のWinUSBドライバを残しておきたい場合、MPLAB X IDEのインストール前にファイル名を変えておきます。

WinUSBドライバファイルは以下の場所に保存されています。

32ビットWindowsの場合:

`C:\Windows\system32\WinUSB.dll` **(32ビット)**
`C:\Windows\system32\drivers\WinUSB.sys` **(64ビット)**

64ビットWindowsの場合:

`C:\Windows\SysWOW64\WinUSB.dll` **(32ビット)**
`C:\Windows\system32\WinUSB.dll` **(64ビット)**
`C:\Windows\system32\drivers\WinUSB.sys` **(64ビット)**

2.3.2 MPLAB IDE v8.xxがシステムにインストールされていない場合

特別な操作は必要ありません。MPLAB X IDEのインストール時にUSBドライバもインストールされます。PCのUSBポートにツールを接続すると、[新しいハードウェアが見つかりました]というバルーン メッセージが表示されます。その後、ドライバが自動的にインストールされます。またはウィザードに従い[自動的にドライバを選択する]ように設定します。しかし、どちらの手順も失敗した(プリインストールされたドライバとツールを関連付けられなかった)場合、手動でインストール/関連付けを行う必要があります。その手順とドライバの保存場所は2.3.3.4「ドライバの手動切り換えが必要な場合」を参照してください。

2.3.3 MPLAB IDE v8.xxがインストール済みの場合

コンピュータにMPLAB IDE v8.xx以前のバージョンがインストール済みの場合、MPLAB Device Driver Switcherを実行してMPLAB IDE v8ドライバからMPLAB X IDEドライバに切り換える必要があります。

ツールに関連付けられているドライバの判定は以下の手順で行います。

1. ツール ハードウェアをPCのUSBコネクタに接続します。
2. PCの[デバイス マネージャ]ウィンドウを開きます。
   2.1. MPLAB IDE v8.xxの場合、ドライバは[Custom USB Device] > [Microchip Custom USB Device]に保存されています。
   2.2. MPLAB X IDEの場合、ドライバは[Microchip Tools] > [Microchip WinUSB Device]に保存されています。
3. MPLAB IDE v8.xxドライバをSwitcherでMPLAB X IDEドライバに切り換えます。

MPLAB X IDEのドライバに切り換えるには、以下を参照してください。

1. 手動切り換え(Windows XP 64の場合)
2. 管理者モードの使用(Windows 7または8の場合)
3. ツール用USBデバイスドライバの切り換え

Switcherがドライバの切り換えに失敗した場合、以下を参照してください。

1. ドライバの手動インストールが必要な場合
2. ツールの通信問題

2.3.3.1 手動切り換え(Windows XP 64の場合)

Note: Microsoft Windows XP Professional SP3のサポートは終了しました。ただし、サポート外ですがほとんどの場合MPLAB X IDEは動作します。

64ビット版Windows XPを使っている場合、MPLAB Device Driver Switcherで自動的にドライバを切り換える事はできません。この場合、手動で切り換える必要があります。2.3.3.4「ドライバの手動切り換えが必要な場合」を参照してください。

2.3.3.2 管理者モードの使用(Windows 7または8の場合)

MPLAB Device Driver Switcherを実行するには、管理者モードでログインする必要があります。

Switcherユーティリティを管理者として実行するには、実行ファイルを右クリックして[管理者として実行]を選択します。実行ファイルは以下の場所に保存されています。

- 32ビット版の場合: `C:\Program Files\Microchip\MPLABX\vx.xx\Switcher\32Bit\MPDDSwitch.exe`
- 64ビット版の場合: `C:\Program Files(x86)\Microchip\MPLABX\vx.xx\Switcher\64Bit\MPDDSwitch64.exe`

ドライバの切り換えにはSwitcher GUIユーティリティを推奨します。このユーティリティがうまく動作しない場合のみ、コマンドライン アプリケーションを使います。

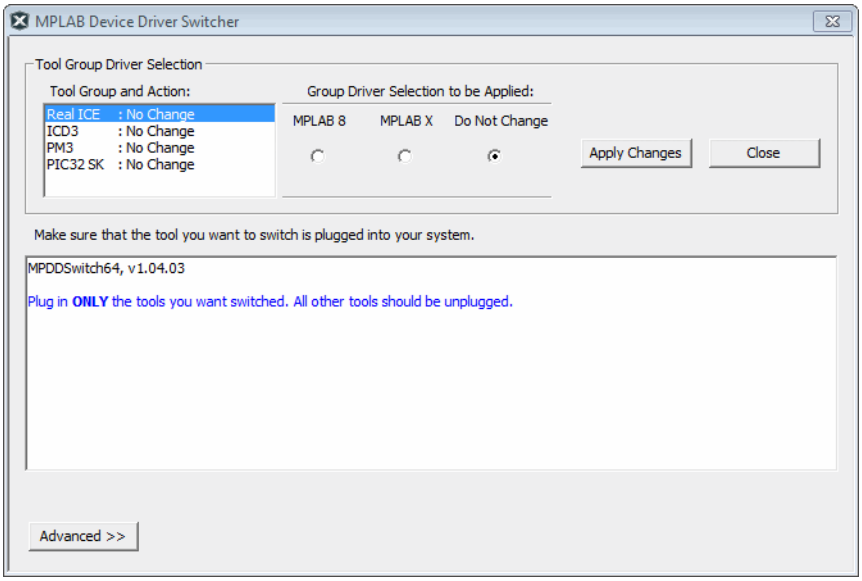
コマンドライン アプリケーション(mchpdds32.exeまたはmchpdds64.exe)を管理者として実行するには、コマンドプロンプトを管理者モードで起動します。[スタート] > [全てのプログラム] > [アクセサリ] > [コマンド プロンプト]を右クリックして[管理者として実行]を選択します。管理者用コマンドプロンプト画面が開きます。ここで、ReadMe32.txtまたはReadMe64.txtファイルに記載されているコマンドを入力してドライバを切り換えます。

2.3.3.3 ハードウェア ツール用USBドライバの切り換え

MPLAB Device Driver Switcherは、MPLAB IDE v8ドライバとMPLAB X IDEドライバを切り換えるためのGUIアプリケーションです。デスクトップのアイコンをクリックすると、Switcherユーティリティが起動します。

図2-1 MPLAB Driver Switcherのアイコン

図2-2 Swithcerユーティリティ



Switcherの操作

USBドライバの切り換えは以下の手順で行います。

1. [Tool Group and Action]の下で、ドライバを切り換えツールをクリックして選択します。
   Note: Switcher実行時にツールを接続しておかないと、そのツールに対応したドライバのインストールと切り換えができません。

2. ラジオボタンで[MPLAB 8]または[MPLAB X]を選択します。
3. [**Apply Changes**]ボタンをクリックします。切り換えの進捗状況がテキスト ウィンドウに表示されます。切り換えには少し時間がかかります。
4. 切り換えが完了したら、[デバイス マネージャ]ウィンドウでドライバ名を確認します。MPLAB X IEの場合、ドライバは[Microchip Tools] > [Microchip WinUSB Device]で確認できます。MPLAB IDE v8.xxの場合、ドライバは[Custom USB Device] > [Microchip Custom USB Device]で確認できます。

Switcherのトラブルシュート

1. GUIによるドライバの切り換えに失敗した場合、[**Advanced**]ボタンをクリックしてドライバファイルへのパスを確認した上でSwitcherを再度実行します。
2. それでもGUIからドライバを切り換えられない場合、手動で切り換えます。その手順とドライバの保存場所は、2.3.3.4「ドライバの手動切り換えが必要な場合」を参照してください。

Switcher実行ファイルの保存場所

Switcher実行ファイルは、MPLAB X IDEインストール フォルダ、Switcherサブフォルダの以下の場所に保存されています(既定値)。

- 32ビット版の場合: `C:\Program Files\Microchip\MPLABX\vx.xx\Switcher`
- 64ビット版の場合: `C:\Program Files (x86)\Microchip\MPLABX\vx.xx\Switcher`

vx.xxはMPLAB X IDEのバージョンです。

2.3.3.4 ドライバの手動切り換えが必要な場合

USBドライバの手動切り換えまたは関連付けが必要な場合、以下の手順を実行します。

1. [コントロール パネル]の[デバイス マネージャ]を開きます。[Microchip Tools]でツールを探します。見つからない場合、[その他のデバイス]の下の[不明なデバイス]で探します。
2. ツール名または[不明なデバイス]を右クリックして、[ドライバ ソフトウェアの更新]を選択します。
3. [ドライバ ソフトウェアの更新]ダイアログで[コンピュータを参照してドライバ ソフトウェアを検索します]を選択します。
   Note: [ドライバ ソフトウェアの最新版を自動検索します]は選択しないでください。Windowsの既定値ドライバがツールに関連付けられてしまい、ツールが機能しなくなる場合があります。誤って選択した場合、元に戻るかダイアログを終了し、上の手順を再度実行して正しいドライバに切り換えます。
4. デバイスドライバの正しい保存場所を参照します。
   MPLAB X IDEの場合、デバイスドライバの既定値での保存場所は以下の通りです。

   ```
   C:\Program
   Files\Microchip\MPLABX\vx.xx\Switcher\32Bit\ winusb\x86\MCHPWinUSBDevice.inf
   ```

   または

   ```
   C:\Program Files
   (x86)\Microchip\MPLABX\vx.xx\Switcher\64Bit\ winusb\amd64\MCHPWinUSBDevice.inf
   ```

   vx.xxはMPLAB IDEMPLAB X IDEのバージョンです。
5. Windowsのセキュリティ ダイアログが表示された場合、[このドライバ ソフトウェアをインストールします]を選択し、選択したドライバに切り換えます。

2.3.3.5 ツールの通信問題

1. ドッキング ステーションまたはハブを介してツールを接続するとうまく行かない場合があります。その場合PCのUSBポートに直接ツールを接続します。
2. デバイスドライバを手動で切り換える場合、「32Bit」または「64Bit」フォルダ内のINFファイルを指定する必要があります。詳細は2.3.3.4「ドライバの手動切り換えが必要な場合」を参照してください。

2.4 ターゲットとハードウェア ツールの接続

インサーキット デバッガやエミュレータを使う場合、以下を参照してそれらのツールをターゲットに接続します。

- 開発ツールの設計アドバイザリ
- ヘッダの仕様書(ヘッダを使う場合)

- ツールのマニュアル

プログラマ専用機の場合、プログラマのマニュアルを参照してください。

Microchip社のデモボード、評価用キット、リファレンス デザインをターゲットとして使う場合、セットアップに関する情報は、それらに付属する文書を参照してください。

2.5 言語ツールのインストール

MPLAB X IDEをインストールする際に、言語ツールのMPASMツールチェーンも一緒にインストールされます。

さらに、MPLAB X IDEでは以下のCコンパイラ ツールチェーン (コンパイラ、アセンブラ、リンカ等)を使えます。コンパイラ ツールチェーンを選ぶ際は、設計に使うデバイスがサポートされているか注意します。

- MPLAB XC Cコンパイラ
- AVRおよびArm Cコンパイラ
- AVRASM2 - Atmel Studioと一緒にインストールされる

MPLAB XCコンパイラは、FreeとPRO(機能制限なし、コード最適化使用可能)のライセンスがあります。これらのコンパイラのインストールとライセンス取得の方法は『MPLAB® XC Cコンパイラのインストールとライセンス認証』(DS50002059)を参照してください。MPLAB X IDEは、インストール済みコンパイラのライセンス取得とネットワーク ライセンスの取得/返却機能を備えています。[Tools] > [Licenses]で行います。

AVRとArm GNU Cコンパイラ(GCC)は無償です。インストーラの画面に従ってインストールします。

Note: これらのコンパイラは、MPLAB X IDEがアクセスできる場所にインストールする必要があります(例: MPLAB XC Cコンパイラの場合はC:\Program Files\Microchip)。そうでない場合、MPLAB X IDEでその場所を指定します(4.7「言語ツールのパスの指定」参照)。

2.6 MPLAB IDEの起動とデスクトップの表示

[MPLAB X IDE]アイコンをダブルクリックするとプログラムが起動します。

図2-3MPLAB X IDEのアイコン

MPLAB X IDEは、NetBeansプラットフォームに基づいて構築されています。NetBeans IDEを使い慣れていれば、MPLAB X IDEデスクトップはすぐに使えます。しかし、以下のタブはMPLAB X IDE特有のものです。

- [Start Page]
- [MPLAB X Store]

[Start Page]は、各種のリンクを収めた3つのタブで構成されています。起動時に[Start Page]を表示したくない場合、各タブの下部の[Show on Startup]のチェックを外します。

図2-4　MPLAB X IDEのデスクトップ

2.7　[MPLAB X Store]での購入

[MPLAB X Store]タブをクリックするとMicrochip社オンラインストアからツールを購入できます。このタブではMPLAB X IDEに関する最新の商品を紹介します。各種カテゴリを参照して他の商品を見つける事もできます。

このタブは[Help]メニューまたはツールバー アイコンから直接アクセスする事もできます。

図2-5　MPLAB X Store

追加の機能が予定されています。[Start Page]の[My MPLAB IDE]からmicrochipDIRECTへのログインが可能になる予定です。その場合、

- ソフトウェアに関するアラート(HPAの失効等)を確認できます。
- mySoftwareアカウントにアクセスできます。mySoftwareアカウントでは各種ライセンスを登録およびダウンロードし、HPAを更新できます。

2.8 複数インスタンスのIDEを起動する方法

MPLAB X IDEの各インスタンスでハードウェア ツール(MPLAB PICkit 4等)を使うには、事前に設定が必要です。ハードウェア ツールの準備が完了したら、IDEのインスタンスをIDEと同じフォルダから起動します。

2.8.1 複数インスタンスを使うためのハードウェア ツールの設定方法

既定値ではIDEのインスタンスを最大5つ使えます。それ以上のインスタンスを使う場合、「mchpdefport」ファイルを手動で編集する必要があります。

「mchpdefport」ファイルの使い方とフォーマット

「mchpdefport」ファイルは、IDEとローレベルUSBライブラリ(DLL、so、dylibファイル)にツールのホットプラグに必要な情報を提供します。このファイルのフォーマットを以下に示します。

```
localhost
30000
1/30002
1/30004
1/30006
1/30008
```

1行目はIDEを実行するホスト名を示します。

以降の行は、ローレベルのライブラリがハイレベルのIDEと通信するポートまたはソケットの番号を表します。IDEの各インスタンスは、別々のポートまたはソケットに割り当てられます。IDEのインスタンス間の通信はユーザからは見えません。

MPLAB X IDEのインスタンスが5つまでの場合、このファイルを編集する必要はありません。6つ以上のインスタンスを使う場合、このファイルを編集してポートまたはソケットの数を追加します。

「mchpdefport」ファイルの位置

IDEの既定値インストールでは、「mchpdefport」ファイルはオペレーティング システムに応じて以下の場所に保存されています。

OS	場所

Windows (64ビット)	`C:\Windows\system32`
	と
	`C:\Windows\SysWOW64`
	Note: 両方の「mchpdefport」を編集する必要があります。
	Note: 64ビットシステムで32ビットエディタを使って`C:\Windows\system32`の「mchpdefport」を編集すると、実際には`SysWOW64`の「mchpdefport」を編集しています。`system32`のファイルを編集するには64ビットエディタを使う必要があります。

Windows (32ビット)	`C:\Windows\system32`
Linux	`/etc/.mplab_ide`
Mac (OS X)	`/etc/.mplab_ide`

2.8.2 IDEのインスタンスの呼び出し方

MPLAB X IDEでは、各インスタンスが独自のユーザフォルダを持つ必要があります。従って、あるインスタンスに対するプリファレンスの設定またはプラグインの追加は、他のインスタンスには反映されません。

複数のインスタンスを使い分けるには、[`--userdir`]オプションでフォルダを指定してIDEを起動します。

Windows

[`--userdir`]オプションを使ってショートカットを作成します。例: Windows 7:

1. デスクトップで右クリックし、[新規作成] > [ショートカット]を選択します。

2. [参照]をクリックして、インストールされているMPLAB X IDEの実行ファイルを指定します(以下は既定値)。
   `C:\Program Files (x86)\Microchip\MPLABX\vx.xx\mplab_platform\ bin\mplab_ide.exe`
   (`vx.xx`はMPLAB X IDEのバージョン)
3. 行の末尾にユーザフォルダの場所を入力します。このタイプのMPLAB X IDE情報の場合、既定値の場所の下にユーザフォルダを置く事ができます。
   `--userdir "C:\Users\MyFiles\AppData\Roaming\.mplab_ide\ dev\anydir"`
   または、任意の場所に置いてもかまいません。
   `--userdir anydir`
4. [OK]をクリックします。

Linux

インストールしたバージョンを、パラメータを指定せず起動(デスクトップ アイコンをクリック)した場合、ユーザフォルダとして「`$ (HOME)/.mplab_ide`」を使います。ユーザフォルダを変更するには、シェルスクリプト「`$InstallationDir/mplab_platform/bin/ mplab_ide`」に引数「`--userid anydir`」を渡して実行します。以下は、MPLAB X IDEのインスタンスを2つ実行する場合の例です。

```
$ /opt/microchip/mplabx/vx.xx/mplab_platform/bin/mplab_ide --userdir ~/.anydir1 &
$ /opt/microchip/mplabx/vx.xx/mplab_platform/bin/mplab_ide --userdir ~/.anydir2 &
```

vx.xxはMPLAB X IDEのバージョン

ユーザIDを埋め込んだデスクトップ アイコンを作成する事もできます。

Mac OS

シェルウィンドウを開き、以下のコマンド行を入力して、インストールされているMPLAB X IDEを別のユーザフォルダで実行します。このタイプのMPLAB X IDE情報の場合、既定値の場所の下にユーザフォルダを置く事ができます。

```
$/bin/sh /Applications/microchip/mplabx/vx.xx/mplab_ide.app/Contents/
Resources/mplab_platform/bin/mplab_ide --userdir "${HOME}/Library/
Application Support/mplab_ide/dev/anydir"
```

または、任意の場所に置いてもかまいません。

```
$/bin/sh /Applications/microchip/mplabx/vx.xx/mplab_ide.app/Contents/
Resources/mplab_platform/bin/mplab_ide --userdir anydir
```

2.9 複数バージョンのIDEを使う方法

MPLAB X IDE v3.00がリリースされる前は、異なるバージョンのMPLAB X IDEをインストールする場合、異なるフォルダにインストールする必要がありました。MPLAB X IDE v3.00からは、これは既定値で行われます。例を以下に示します。

```
C:\Program Files (x86)\Microchip\MPLABX\v3.00
```

複数のバージョンを同時に起動および実行できます。各バージョンは独自のユーザフォルダを持ちます。

任意のバージョンのMPLAB X IDEとMPLAB IDE (v8以前)を同一PC上で実行できます。しかし、「**USBデバイスドライバ (ハードウェア ツール用)のインストール**」に記載したUSBハードウェア ドライバの制約に留意する必要があります。

2.10 起動パラメータによる起動

コマンドライン オプションの多くはNetBeansプラットフォーム

からのものです。以下を参照してください。

http://wiki.netbeans.org/FaqStartupParameters

さらに、`--help`オプションを使うと、MPLAB® X IDEオプションのリストを参照できます。例えば64ビットWindows® 7システムの場合:

```
>C:\Program Files (x86)\Microchip\MPLABX\vx.xx\mplab_platform\bin\mplab_ide --help
```

3. チュートリアル

チュートリアルでは、例を示しながらMPLAB X IDEの使い方を説明します。

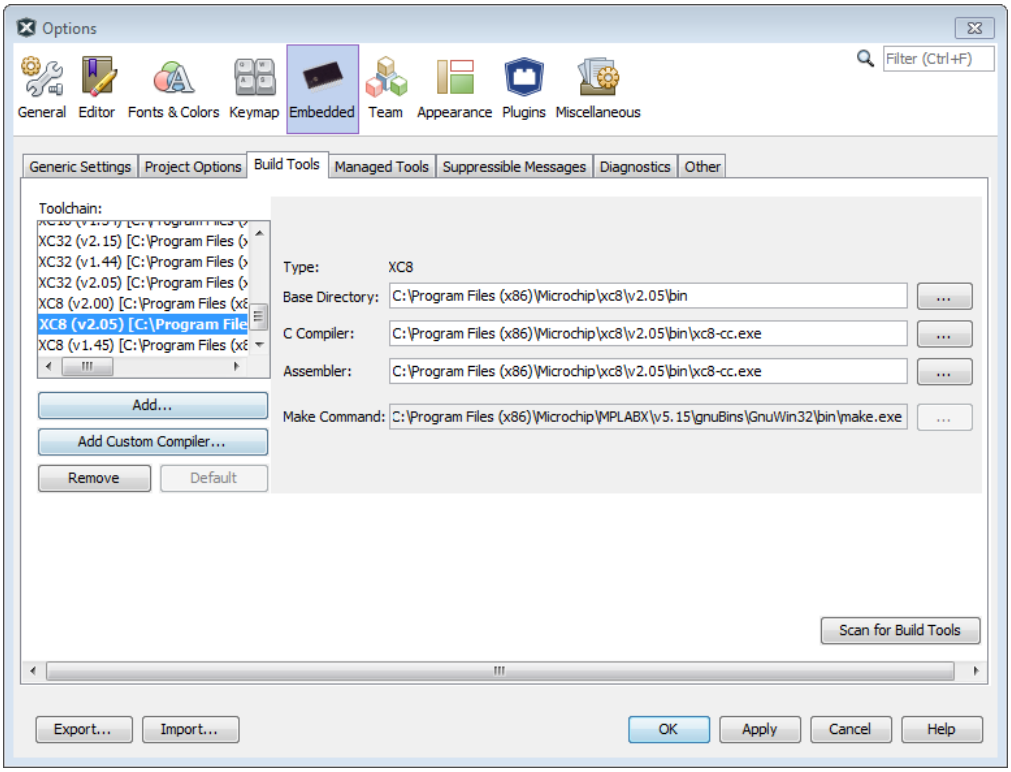
サンプル プロジェクトではポテンショメータの値を4つのLEDに表示します。このプロジェクトをMPLAB X IDEでダウンロードして開くと、プロジェクト プロパティの設定/変更方法とプロジェクトの実行方法が説明されます。ブレークポイントの設定、コードのステップ実行、変数値の表示、レジスタおよびメモリ値の表示/変更等のデバッグ機能が示されます。

3.1 ソフトウェアのインストールとセットアップ

以下のソフトウェアをダウンロードしてインストールします。インストールの詳細は第2章「使用前の準備」を参照してください。

- MPLAB X IDEをインストールします。http://www.microchip.com/mplabxで無償IDEをダウンロードします。
- MPLAB XC8 Cコンパイラをインストールします。http://www.microchip.com/xcで無償MPLAB XCコンパイラをダウンロードします。

MPLAB X IDEを起動します。IDEウィンドウの[Tools] > [Options] > [Embedded] > [Build Tools]でコンパイラが表示されている事を確認します。表示されていない場合、[**Add**]ボタンをクリックし、コンパイラがインストールされた場所を手動で探します。

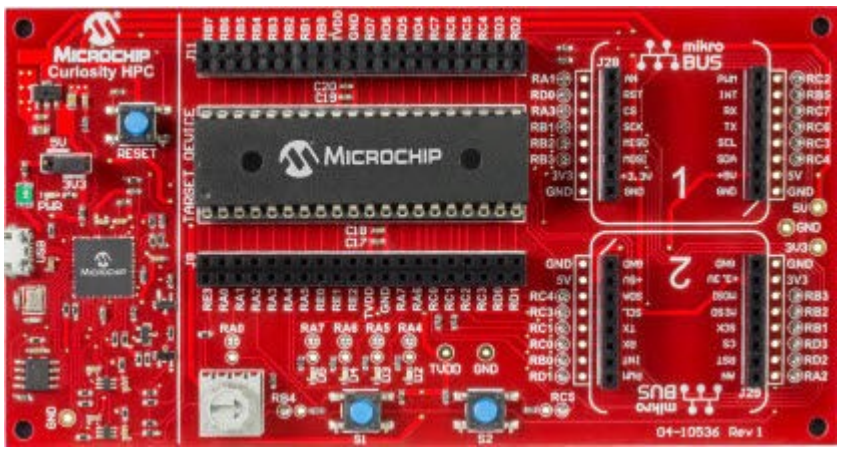


3.2 ハードウェアの接続

Curiosity多ピン(HPC)開発ボード(DM164136)とPIC16F18875を使います。詳細は以下の製品ページを参照してください。

- DM164136
- PIC16F18875

評価用ボードとPCをUSBケーブルで接続します。ドライバが自動的にインストールされます。



3.3 サンプルコードのダウンロード

[MPLAB Xpress Code Examples]を開

きます。以下のように選択します。

- [**Tags**]の下で「#Getting Started」にチェックを入れます。
- [**Board**]の下で「Curiosity HPC Board」にチェックを入れます。
- [**Title**]の下で「CuriosityHPC-Basics」をクリックします。

MPLAB Xpress Code Examples

Title	Author	Like	Watch	Import	Tags	Board	Device
Search		From	From	From	#Getting Started	Curiosity Board	Search
		To	To	To	ADC	Curiosity HPC Board	
					ADCC	Curiosity Nano	
CuriosityHPC-Basics		0	0	282	#Getting Started, ADCC,...	Curiosity HPC B...	PIC16F18875

「CuriosityHPC-Basics」ページに、ボードとデバイスの情報へのリンクが表示されています。[**Download**]をクリックしてサンプル プロジェクトをダウンロードします。ファイルを解凍します。

ヒント: `Users>UserName>MPLABXProjects`フォルダにダウンロードします。

CuriosityHPC-Basics

Download 85 Open 298

3.4 MPLAB X IDEでサンプル プロジェクトを開く

MPLAB X IDEで[Open Project]アイコン  をクリックします。リストをスクロールして「CuriosityHPC-Basics」プロジェクトを選択し、[**Open Project**]をクリックします。

[Projects]および[Files]ウィンドウの表示

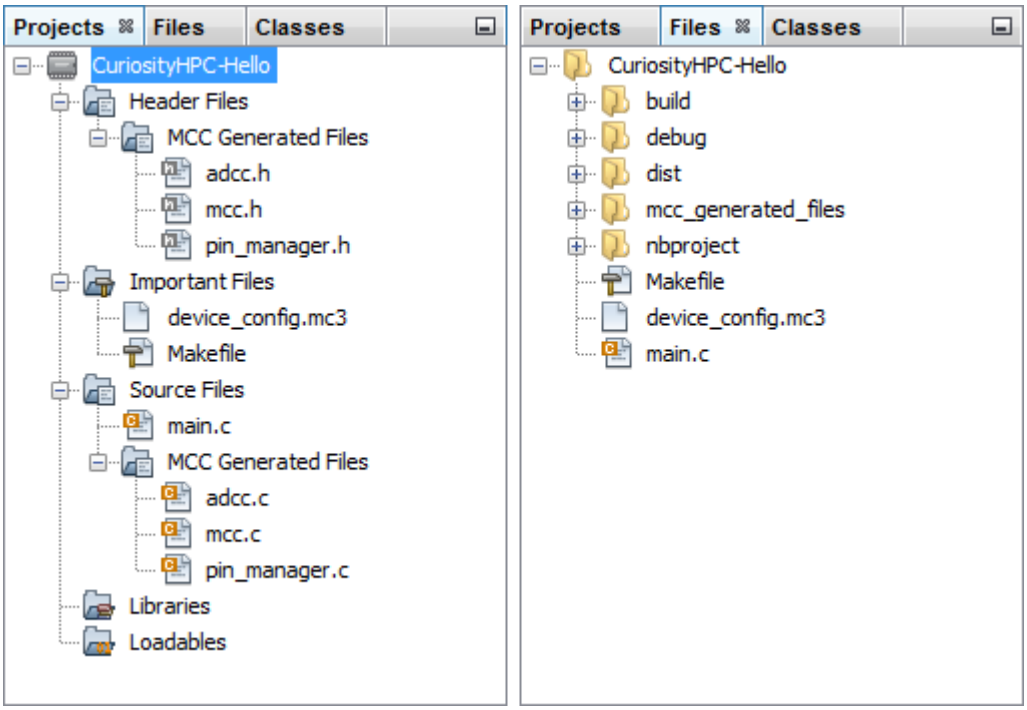
[Projects]ウィンドウには、ファイルがカテゴリごとにグループ分けされたプロジェクト ツリーが表示されます。カテゴリの詳細は8.1「[Projects]ウィンドウの表示」を参照してください。

[Files]ウィンドウをクリックすると、プロジェクト ファイルのファイル マネージャ画面が表示されます。8.2「[Files]ウィンドウの表示」を参照してください。

どちらのウィンドウでもファイル名をダブルクリックすると、そのファイルの内容が[Editor]ウィンドウに表示されます。タブを閉じるには、[x]をクリックします。

[Projects]ウィンドウ内でプロジェクト名を右クリックすると、コンテクスト メニューが開きます。この操作は、プロジェクトのサブフォルダとFilesやClasses等のツリーでも同様です。

図3-1　[Projects]および[Files]ウィンドウの表示

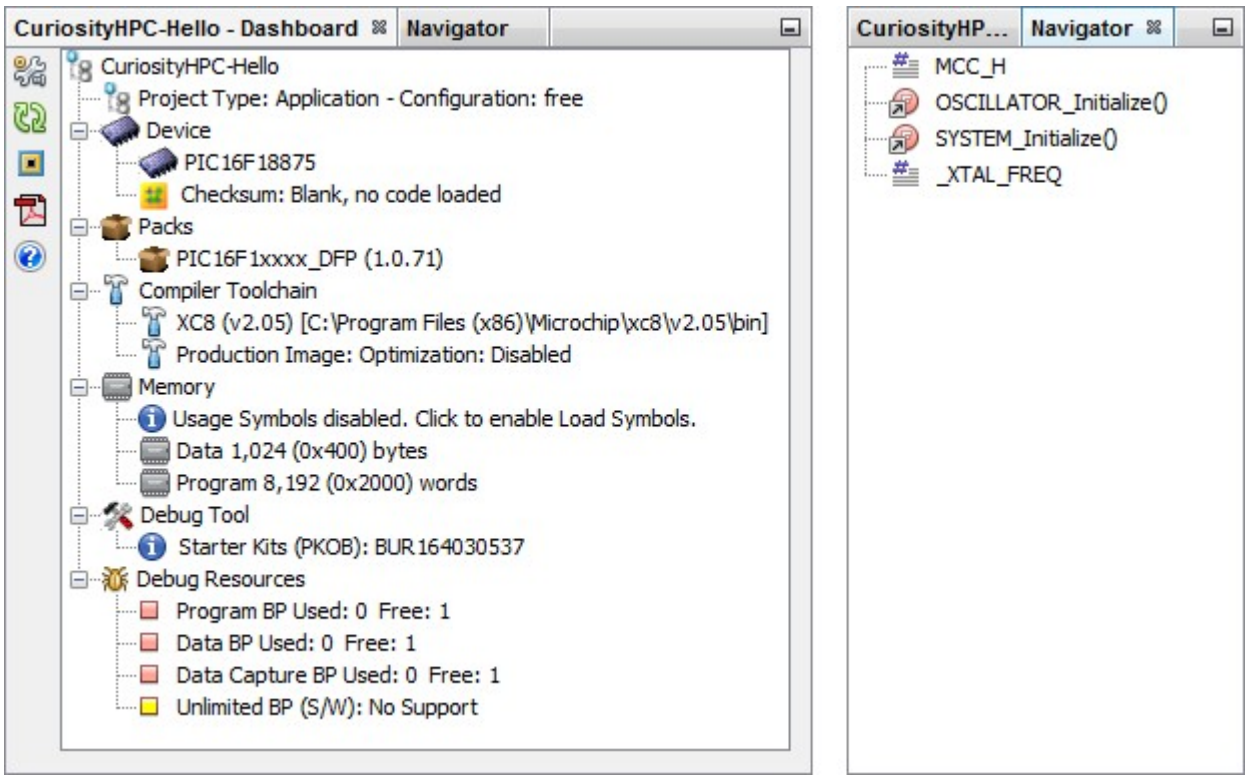


[Dashboard]および[Navigator]ウィンドウの表示

[Dashboard]ウィンドウはプロジェクトの詳細を表示します。詳細は12.6「[Dashboard]ウィンドウ」を参照してください。

[Navigator]ウィンドウは選択中のファイルをコンパクトに表示し、ファイル各部への移動を簡単にします。

図3-2　プロジェクトを開いている時の[Dashboard]ウィンドウ

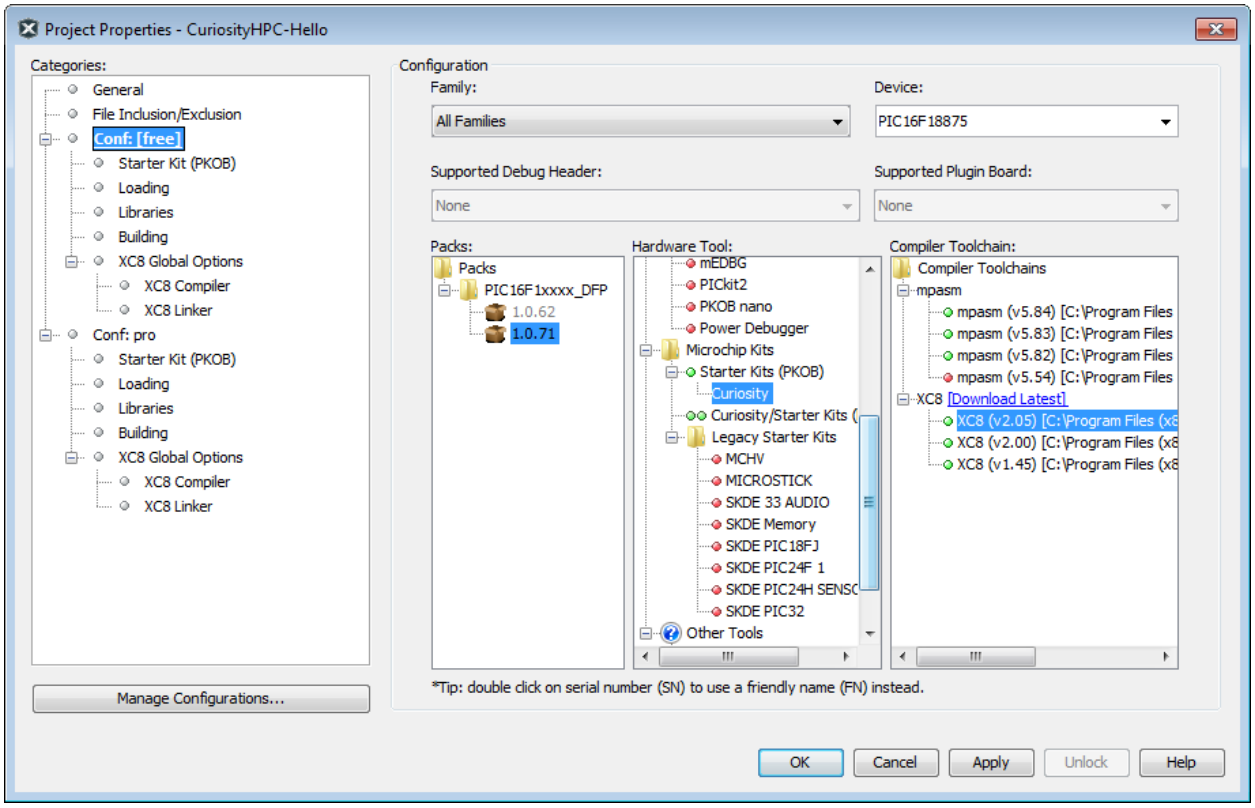


3.5　プロジェクト プロパティの設定

[Projects]ウィンドウでプロジェクト名を右クリックすると、ドロップダウン メニューが表示されます。[Properties]を選択します。[Project Properties]ウィンドウを確認します。[**OK**]をクリックすると、変更が保存されます。

Categories	このプロジェクトは無償版(free)コンパイラを使うように設定されています。ライセンス版を使っている場合、「pro」を選択できます。
Device	ボード上のデバイスと一致しているか確認します。
Packs	下図で選択されているパックバージョンはMPLAB X IDE v5.20用です。
Hardware	[Starter Kits (PKOB)](内蔵デバッガ)として「Curiosity」を選択します。
Compiler Toolchain	チュートリアルのプロジェクトはMPLAB XC8 v1.37で作成しました。図のように最新バージョンのコンパイラに変更するか、以下からプロジェクトで使用したバージョンをダウンロードします。 Downloads Archive

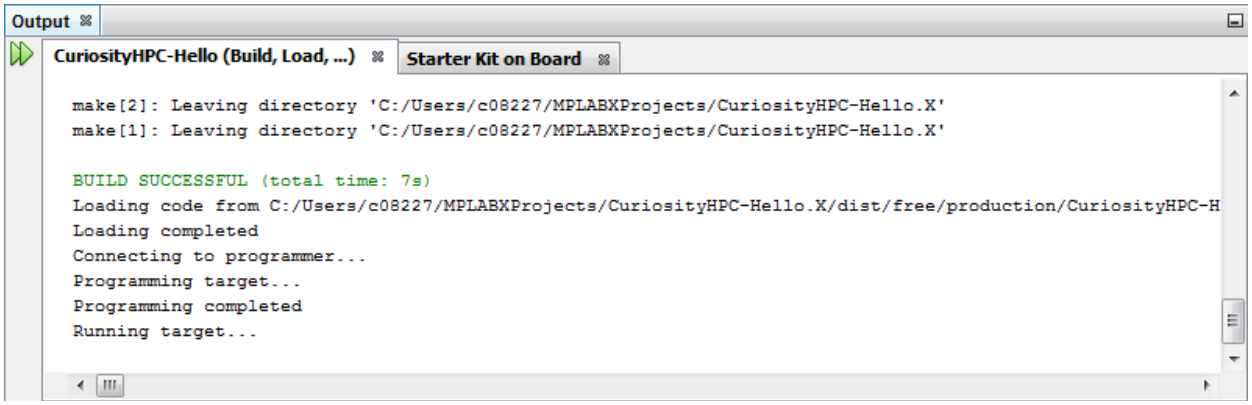
図3-3　[Project Properties]ウィンドウ



3.6　コードの実行

[Run Project]アイコン  をクリックすると、プログラムが実行されます。実行の進捗状況は[Output]ウィンドウに表示されます(下図参照)。[Hold in Reset]ボタン  を使うと、デバイスリセットと実行をトグルできます。

図3-4　[Output]ウィンドウ - コード実行



ボード上のポテンショメータを回すと、値に従ってLEDが点灯します。

3.7 コードのデバッグ

本チュートリアルで使うコードの動作はテスト済みです。しかし、実際にアプリケーションを開発する際にはコードをデバッグする必要があります。

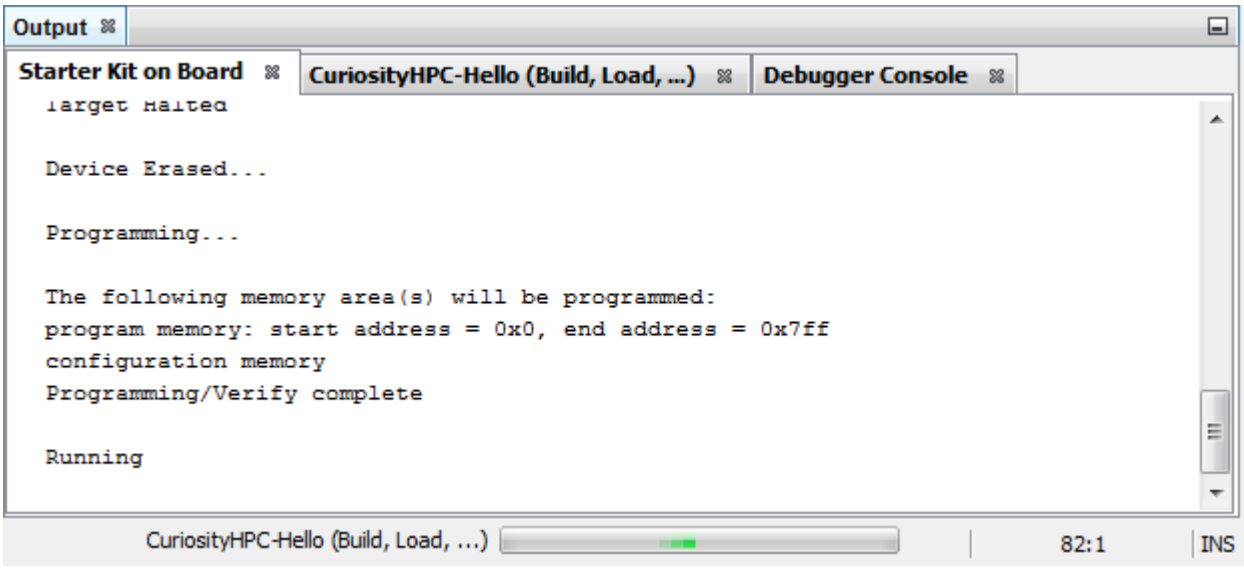
[Debug Project]アイコン  をクリックすると、デバッグ セッションが開始します。デバッグの進捗状況は[Output]ウィンドウに表示されます。

コードの実行を一時停止する		Pause
コードの実行を再開する		Continue
コードの実行を終了する		Finish Debugger Session

C/C++コード プロジェクトのデバッグに関する詳細は以下のNetBeansヘルプも参照してください。

Debugging C/C++ Projects Tutorial

図3-5 [Output]ウィンドウ - デバッグモードでのコードの実行



3.8 ブレークポイントの設定

コードをデバッグする際に、コードの特定位置で実行を一時停止して変数値を調べる事ができると便利です。そのためにはブレークポイントを使います。

ブレークポイントの詳細は以下のNetBeansヘルプも参照してください。

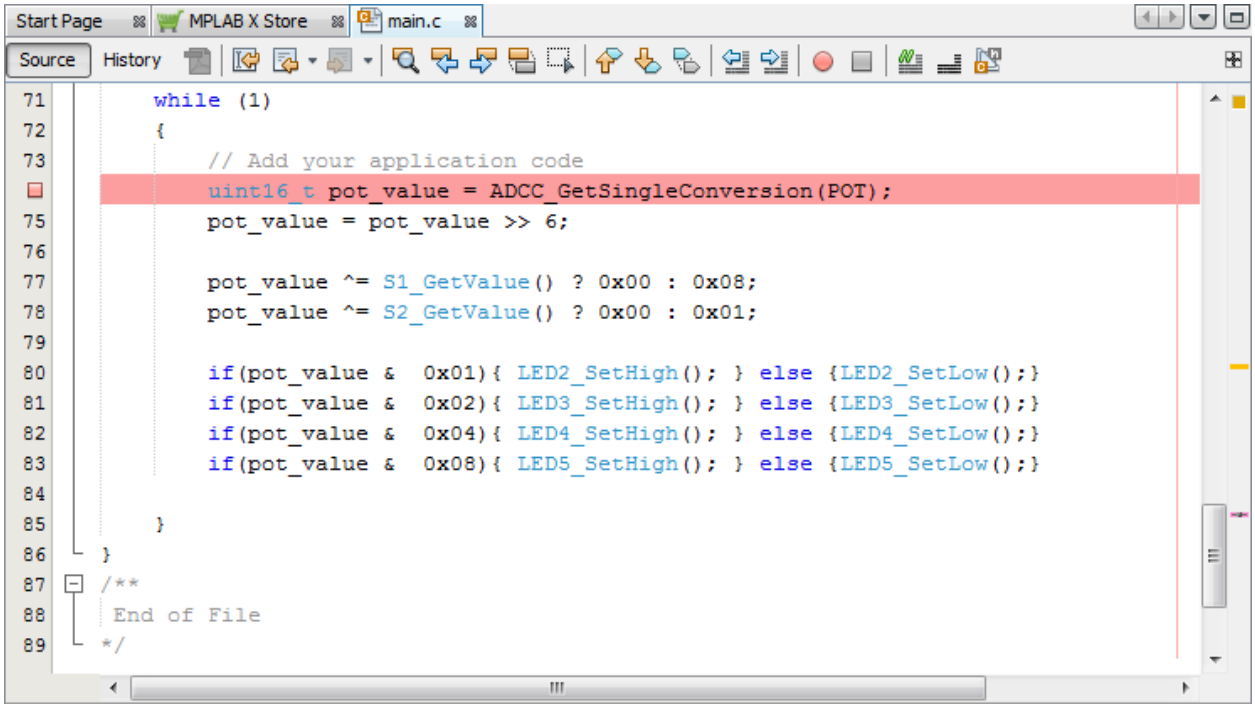
Debugging C/C++ Projects Tutorial>Creating Breakpoints

3.8.1 ブレークポイントの設定

コード実行中であれば、[Finish Debugger Session]アイコン  をクリックして実行を停止します。

[Projects]ウィンドウの[Source Files]でファイルmain.cをダブルクリックすると、このファイルが[Editor]ウィンドウで開きます。下図に示す行の左側の余白をクリックし、この行にブレークポイントを設定します。

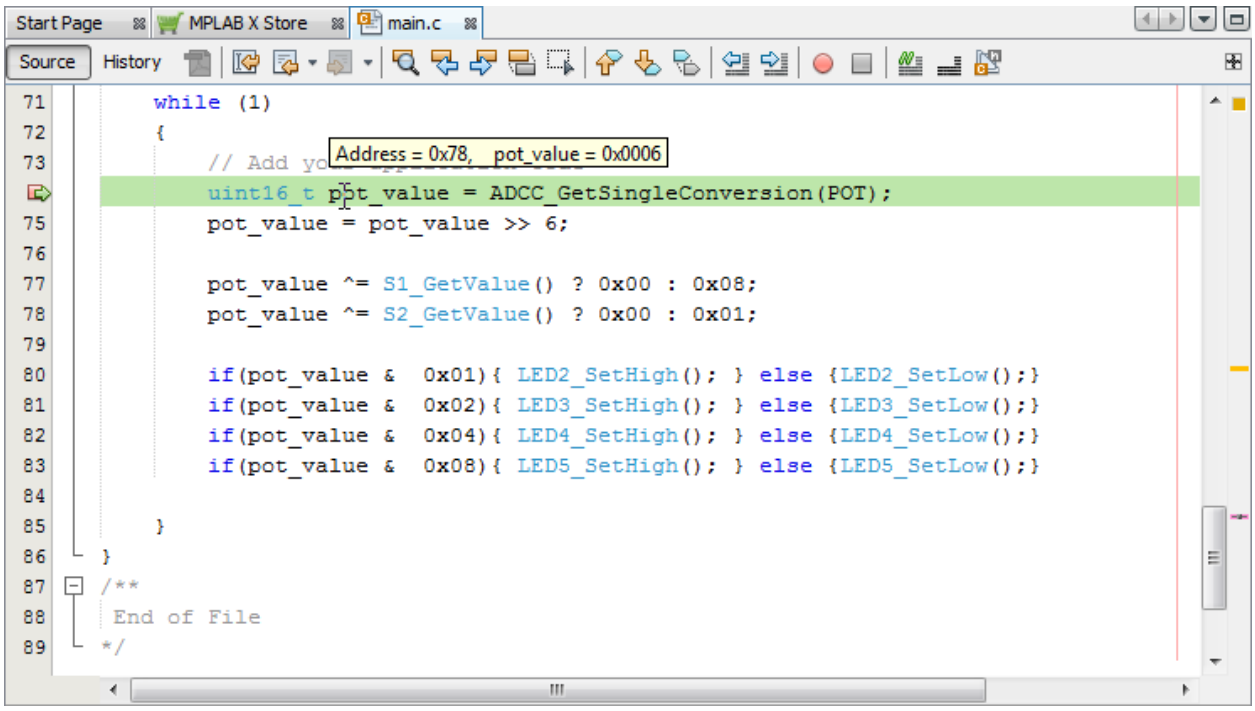
図3-6 コードのブレークポイントの設定



3.8.2 ブレークポイントでの一時停止

[Debug Project]アイコン  をクリックすると、コードを再び実行されます。コード実行はブレークポイントの位置で一時停止します。`pot_value`変数の上にマウスカーソルを置くと、現在値が表示されます。

図3-7　ブレークポイントで一時停止したコード実行



3.8.3　トラブルシュート

エディタでコード行の横に電球アイコン(警告)が表示される場合、インクルード ファイルのパス設定が必要です。以下を参照してください。

Set The Include Directory Path

Note:　MPLAB XC8 v2.xxではAVR MCUとPIC MCUをサポートしているため、コンパイラのインストール フォルダ構造が変更されました。

3.9　コードのステップ実行

Step Intoを使うと、ブレークポイントで停止した状態からコードをステップ実行できます。ステップ実行すると、変数値の変化を調べたり、プログラムフローが正しいかどうかを判断できます。

Step Over – コードを1行実行します。その行が関数コールだった場合、呼び出した関数全体を実行して停止します。

Step Into – コードを1行実行します。その行が関数コールだった場合、呼び出した関数の先頭の命令文を実行して停止します。

Step Out – コードを1行実行します。その行が関数コールだった場合、実行後に呼び出し元の制御に戻ります。

Run to Cursor – コードをカーソル位置まで実行して停止します。

図3-8　ステップ実行中のコード

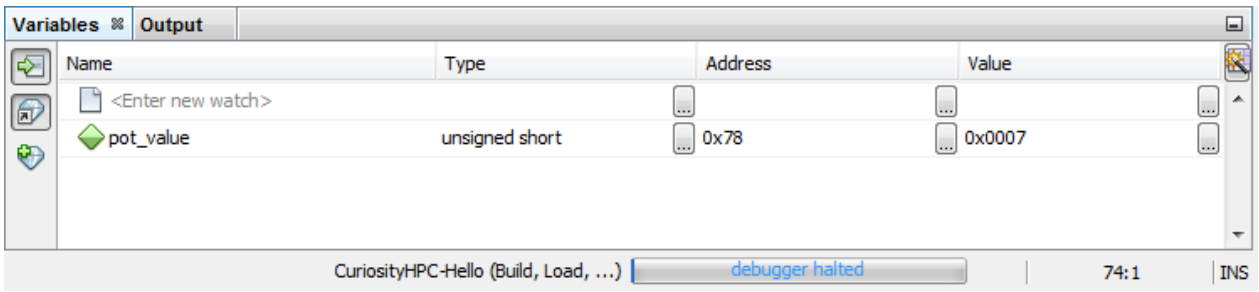
```
71      while (1)
72      {
73          // Add your application code
            uint16_t pot_value = ADCC_GetSingleConversion(POT);
75          pot_value = pot_value >> 6;
76
            pot_value ^= S1_GetValue() ? 0x00 : 0x08;
78          pot_value ^= S2_GetValue() ? 0x00 : 0x01;
79
80          if(pot_value &  0x01){ LED2_SetHigh(); } else {LED2_SetLow();}
81          if(pot_value &  0x02){ LED3_SetHigh(); } else {LED3_SetLow();}
82          if(pot_value &  0x04){ LED4_SetHigh(); } else {LED4_SetLow();}
83          if(pot_value &  0x08){ LED5_SetHigh(); } else {LED5_SetLow();}
84
85      }
86  }
87  /**
88   End of File
89  */
```

3.10　変数値の表示

変数値の変化は[Variables]ウィンドウで観察できます。このウィンドウを開くには[Window] > [Debugging] > [Variables]を選択します。このウィンドウの詳細は4.17「ローカル変数値の変化の観察」を参照してください。

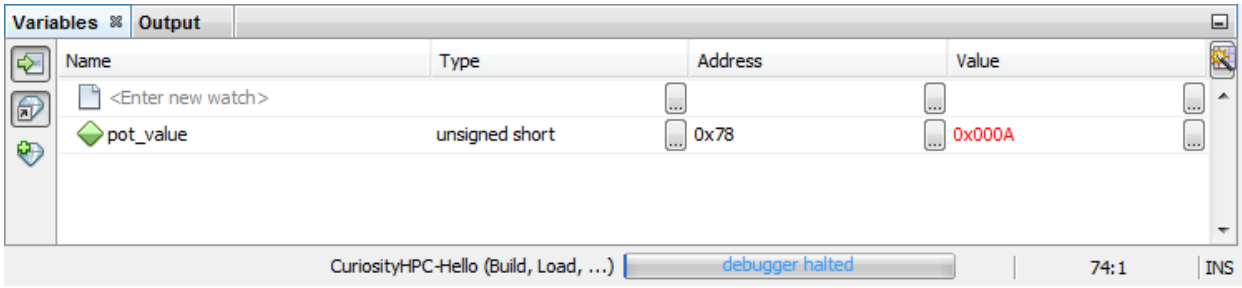
[Debug Project]アイコン  をクリックします。コードが実行され、ブレークポイントで停止します。`pot_value`の情報が[Variables]ウィンドウに表示されます。

図3-9　[Variables]ウィンドウ - 1番目のブレークポイントでの停止



ボード上のポテンショメータを回して値を変化させます。[Continue]アイコン  をクリックします。コードが実行され、再びブレークポイントで停止します。[Variables]ウィンドウを表示し、値が変化した事を確認します(図中の赤字)。

図3-10　[Variables]ウィンドウ - 2番目のブレークポイントでの停止



3.11 シンボル値の変化の観察

グローバル シンボルまたはSFRの値の変化を[Watches]ウィンドウで観察します。これらの値がプログラム実行中に予測通りに変化するか確認する事でコードのデバッグに役立てます。

3.11.1 コードの編集

[Finish Debugger Session]アイコン  をクリックします。

現在pot_valueはローカルシンボルです。これをグローバル シンボルにするためには、main()の前に宣言します。

```
#include "mcc_generated_files/mcc.h"
uint16_t pot_value;
/*                         Main application
 */
void main(void)
```

ローカル変数宣言を解除するためuint16_tを削除します。

```
    while (1)
    {
        // Add your application code
        pot_value = ADCC_GetSingleConversion(POT);
```

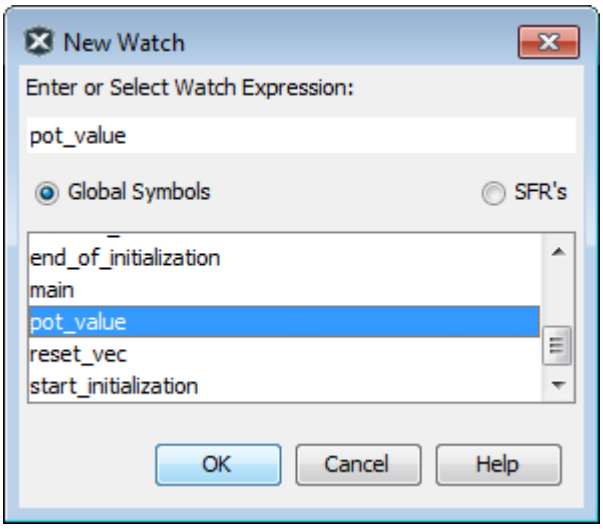
この行にブレークポイントを作り、再び[Debug Project]アイコン  をクリックします。コード実行はこの行で停止します。

3.11.2 新規ウォッチの作成

ウォッチの新規作成方法

1. [Debug] > [New Watch]を選択すると、[New Watch]ダイアログが開きます。
2. ウォッチ式(この例では「pot_value」)を入力し、**[OK]**をクリックします。デスクトップに[Watches]ウィンドウが開き、指定したシンボルが表示されます。

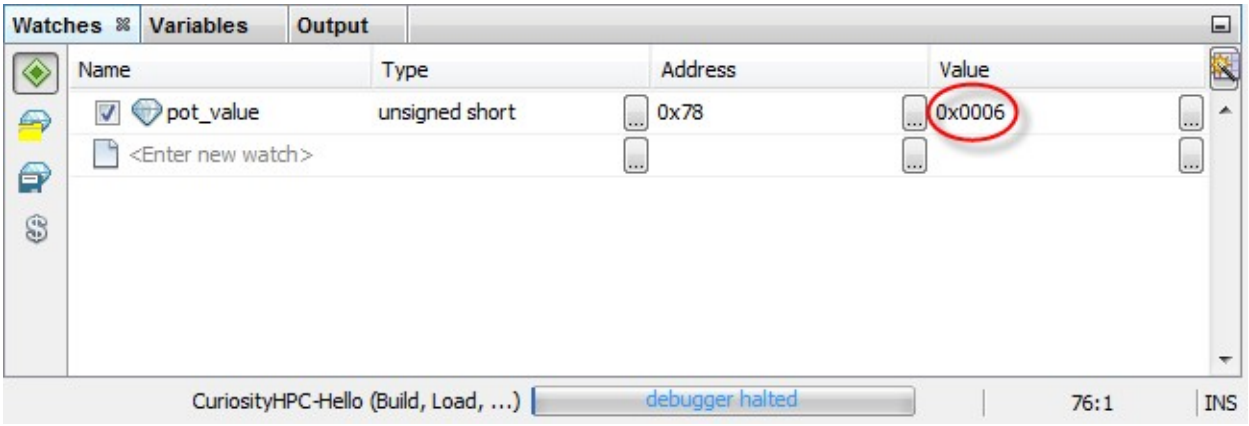
図3-11 [New Watch]シンボル



3.11.3 [Watches]ウィンドウでの値の変化の表示

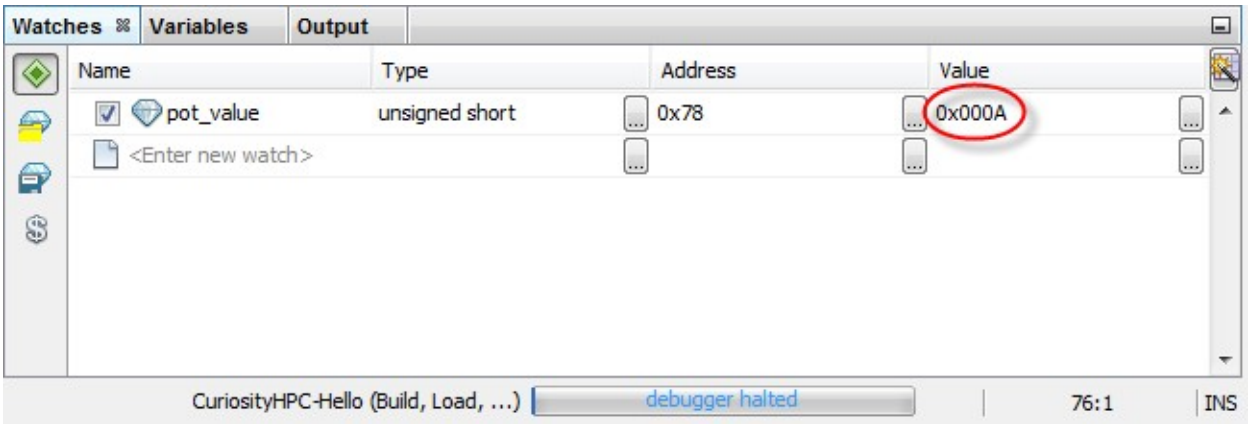
[Debug Project]アイコン  をクリックします。コードが実行され、ブレークポイントで停止します。[Watches]ウィンドウにpot_valueの情報が表示されます。

図3-12　[Watches]ウィンドウ - 1番目のブレークポイントでの停止



ボード上のポテンショメータを回して値を変化させます。[Continue]アイコン  をクリックします。コードが実行され、再びブレークポイントで停止します。[Variables]ウィンドウを表示し、値が変化した事を確認します。

図3-13　[Watches]ウィンドウ - 2番目のブレークポイントでの停止



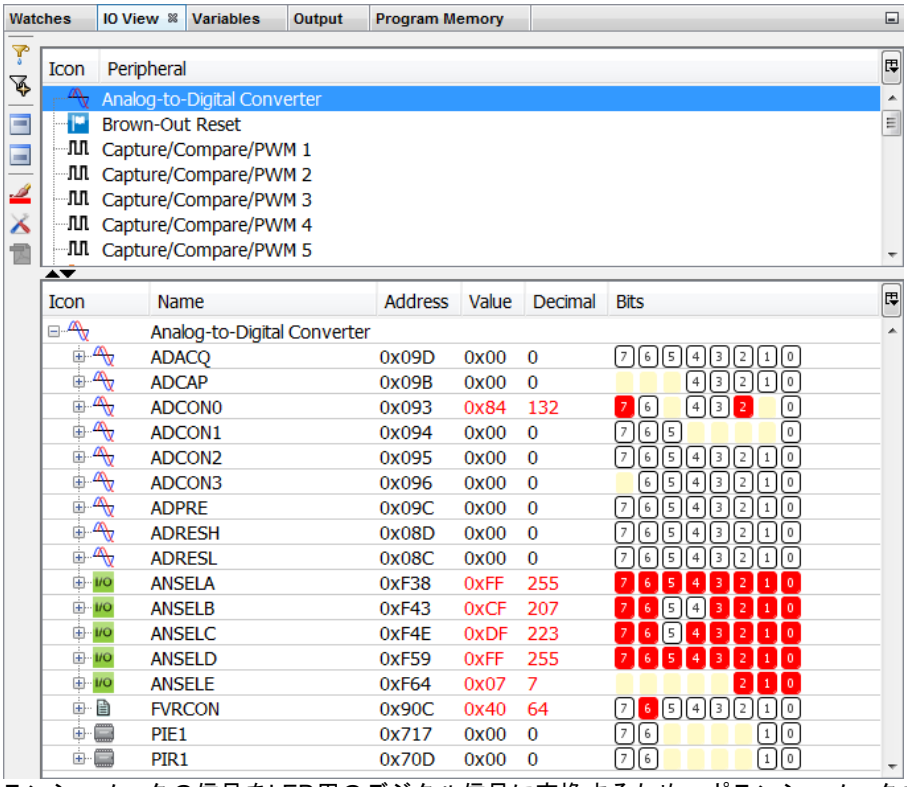
このウィンドウの詳細は12.19「[Watches]ウィンドウ」を参照してください。

3.12　I/Oレジスタの表示

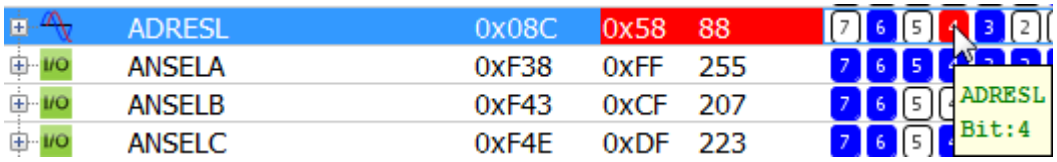
[I/O View]ウィンドウは、アクティブなプロジェクトに関連付けられたデバイスのI/Oメモリマップをグラフィカルに表示します。このデバッグツールはデバッグ時にレジスタが格納している値を表示するため、周辺モジュールの設定を確認できます。再コンパイルしないでレジスタの値を変更する事もできます。

[Window] > [Debugging] > [IO View]を選択して[I/O View]ウィンドウを開きます。ウィンドウの[Peripheral](上部)で[Analog-to-Digital Converter]をクリックします。このモジュールに関連付けられているレジスタが[Register](下部)に表示されます。

図3-14　[IO View]ウィンドウ



ADCを使ってポテンショメータの信号をLED用のデジタル信号に変換するため、ポテンショメータを回すと一部の値が変わります。ポテンショメータを回してから[Continue]アイコン  をクリックします。
これらをクリックして直接レジスタの値を変更する事もできます。



このウィンドウの詳細は、以下のセクションを参照してください。

- 5.20「プロジェクト レジスタの表示([I/O View])」
- 12.8「[I/O View]ウィンドウ」

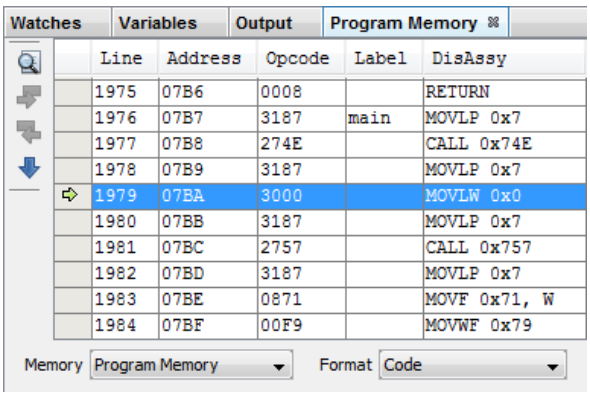
3.13　デバイスメモリ (コンフィグレーション ビットを含む)の表示

MPLAB X IDEは柔軟で見やすい[Memory]ウィンドウを備えています。このウィンドウは、各種デバイスメモリに合わせて、表示を柔軟にカスタマイズできます。[Window] > [Target Memory Views]を選択すると、デバイス固有のメモリウィンドウの一覧が表示されます。

例えばフラッシュメモリを表示するには

1. [Window] > [Target Memory Views] > [Program Memory]を選択します。
2. [Program Memory]ウィンドウが開き、直前の一時停止位置が表示されます。

図3-15　ブレークポイントで停止時の[Memory]ウィンドウの内容



[Memory]ウィンドウのオプションを設定するには、[Memory]ウィンドウ内で右クリックし、各種オプション(表示オプション、メモリの書き込み、テーブルのインポート/エクスポート、ファイルへの出力等)を含むコンテクスト メニューを開きます。メニュー内容はウィンドウによって異なります。12.「MPLAB X IDEのウィンドウとダイアログ」を参照してください。

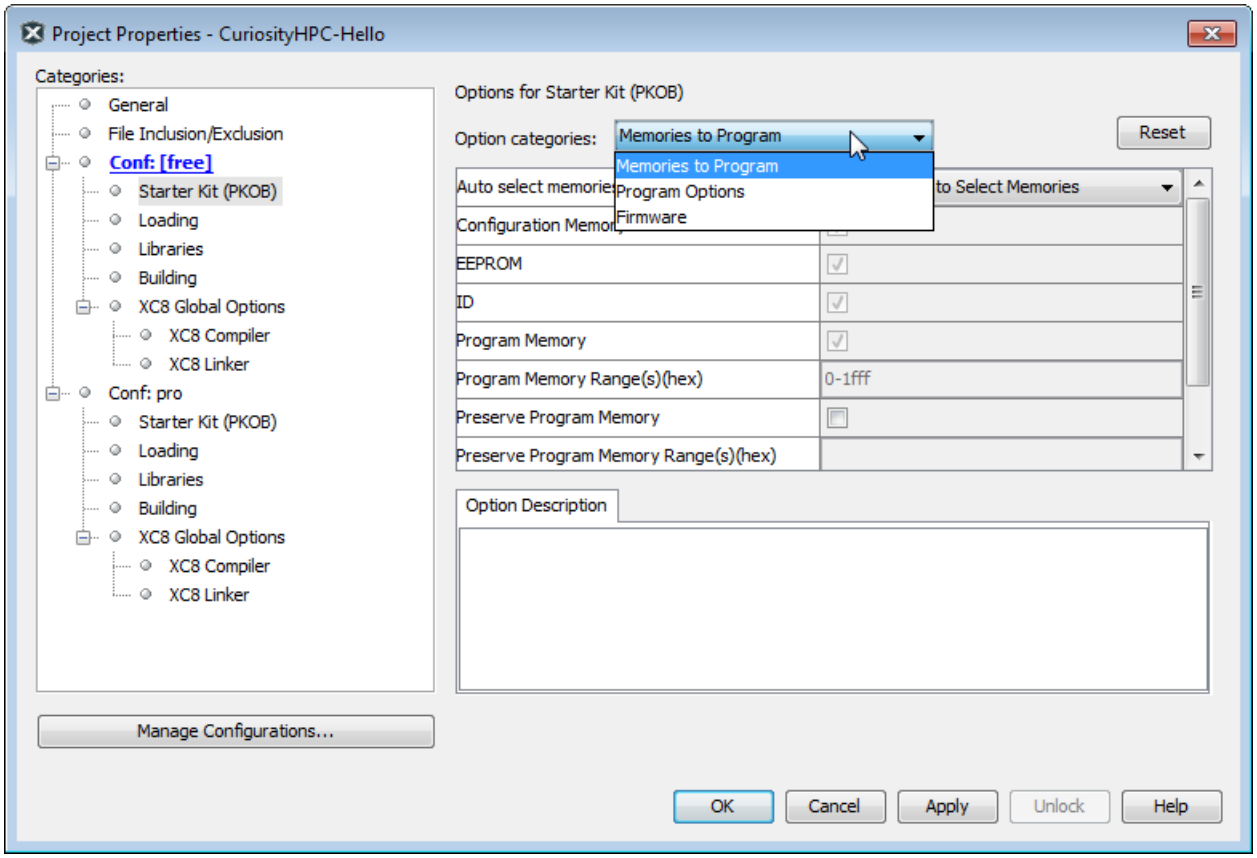
[Flash Memory]ウィンドウは、下の方法で更新できます。

1. コード実行を停止(デバッガ セッションを終了)します。
2. [Read Device Memory]アイコン  をクリックします。

3.14　デバイスのプログラミング

コードをデバッグしたら、ターゲット デバイスにプログラミングします。

まず、[Project Properties]ウィンドウでプログラミング オプションを確認します。本チュートリアルではオプションを変更する必要はありません。



最後に、[Make and Program]アイコン  をクリックしてデバイスをプログラミングします。

Note: MPLAB X IDEは、全てのプログラミング機能を備えている訳ではありません。その他のプログラミング機能のサポートについては、MPLAB X IDEと一緒にインストールされるMPLAB IPEを参照してください。

4. 基本作業

MPLAB® X IDEにおけるプロジェクトの作業手順は以下の通りです。

表4-1　基本作業手順

1 準備	1. 使用前の準備として、MPLAB X IDEをインストールし、ハードウェア ツールを設定し(USBドライバをインストールしてターゲットを正しく接続し)、言語ツールをインストールします。 2. MPLAB X IDEを起動します。
2 プロジェクトの作成とビルド	1. [New Project]ウィザードを使ってプロジェクトを新規作成します(新規プロジェクトの作成)。次に、デスクトップ ペインで変更を確認します(プロジェクト作成後のデスクトップ ペイン)。 2. [Project Properties]ウィンドウを開き([Project Properties]ウィンドウを開く)、プロジェクトを確認または編集し(プロジェクト プロパティの表示と変更)、デバッガ/プログラマ、言語ツールのオプションを設定します(デバッガ/プログラマツール オプションを設定または変更する方法、言語ツールオプションを設定または変更する方法)。 3. [Tools Options]ダイアログで、言語ツールのパスと、その他のツールオプションを設定します(言語ツールのパスの指定、その他のツールオプションの設定)。 4. ファイルを新規作成してプロジェクトに追加します(ファイルの新規作成)。または、既存のファイルをプロジェクトに追加します(既存ファイルのプロジェクトへの追加)。追加したファイルは[File]ウィンドウに表示され、エディタを使ってアプリケーション コードを入力または編集できます。 5. エディタのその他の機能は「エディタの使い方」を参照してください。 6. ライブラリとオブジェクト ファイルを追加、設定します(ライブラリやオブジェクト ファイルの追加と設定)。 7. 各ファイルまたはフォルダ全体のプロパティ (ビルドに含めるかどうか)を設定します(ファイルおよびフォルダ プロパティの設定)。 8. プロジェクトのビルドプロパティを設定します(ビルドプロパティの設定)。プロジェクトのコンフィグレーション タイプ、ビルド前後のステップ、ユーザIDとしてのチェックサムの使用、ビルド完了時の代替HEXファイルのロードに関するビルドプロパティを設定します。 9. プロジェクトをビルドします(プロジェクトのビルド)。
3 コードの実行	1. コードを実行します(コードの実行)。 2. デバッグ セッションでコードをデバッグします(コードのデバッグ)。
4 コードのデバッグ	1. ブレークポイントによりプログラム実行を制御します(ブレークポイントによるプログラム実行の制御)。ブレークポイントは、エディタ内で行を選択して設定するか、[Breakpoint]ウィンドウを使って設定します。 2. プログラム実行時にコードをステップ実行します(コードのステップ実行)。 3. [Watches]および[Variables]ウィンドウでシンボルと変数の値の変化を観察します(シンボル値の変化の観察)。 4. デバイスメモリを表示または変更します(デバイスメモリの表示と変更)。メモリタイプはデバイスによって異なります。「[Configuration Bits]ウィンドウでのコンフィグレーション値の設定」」も参照してください。
5 デバイスのプログラミング	1. ツールバーのボタンを使ってデバイスをプログラミングします(デバイスのプログラミング)。

4.1 新規プロジェクトの作成

プロジェクトには複数のソースファイルと、これらのソースファイルをコンパイルしてリンクし、生成されたプログラムを実行する方法に関する情報が含まれます。IDEは、makefileとメタデータ ファイルを保存するプロジェクト フォルダにプロジェクト情報を保存します。このプロジェクト フォルダにソースフォルダを置く必要はありませんが、プロジェクトの移植性という点では推奨します。

IDEでは常にプロジェクト内で作業を行います。プログラムが1つのソースファイルに格納されている場合も同様です。IDEがプロジェクトをビルドできるように、各プロジェクトは固有のmakefileを持つ必要があります。プロジェクトのmakefileはIDEで生成できますが、IDE外で作成したmakefileを使う事もできます。

IDEが生成したmakefileを含むプロジェクトは**Managed Projects**(管理プロジェクト)と呼び、いくつかの種類があります。MPLAB X IDEで[New Project]ウィザードを使ってプロジェクトを作成する際に、利用可能なプロジェクト タイプから選択できます。

IDE外で作成したmakefileを使うプロジェクトは**User Makefile Projects**(ユーザMakefileプロジェクト)と呼びます。このタイプのmakefileの作成に関する情報は「MPLAB® X IDE外部でのMakefileの作成」を参照してください。

プロジェクトはIDEの[Projects]ウィンドウで操作します(**12.15「[Projects]ウィンドウ」**)。

コンフィグレーション タイプを変更する事でスタンドアロン (アプリケーション) プロジェクトをライブラリ プロジェクトに変更できます。詳細は4.10.1「プロジェクト コンフィグレーション タイプの変更」を参照してください。

4.1.1 MPLAB X IDE v5新プロジェクト フォーマット

v5.00以降の新プロジェクト フォーマットでは、バージョン別にデバイス情報を格納するパックをサポートしています。パックは以下に保存されています。

```
<MPLAB X IDE install directory>\v5.xx\packs
```

プロジェクトの管理

v5.00以降で作成または更新されたプロジェクトは、v4.xxに対して下位互換ではありません。

v5.00以前に作成されたプロジェクトを開くと、現行バージョンにアップグレードされる事を通知するメッセージが表示されます。

- [**Yes**]を選択するとプロジェクトは現行バージョンにアップグレードされ、以前のバージョンで開く事はできなくなります。
- [**No**]を選択すると、プロジェクトに対する変更をv5.00以降に保存する事はできません。変更はv4.xxで行う必要があります。

プロジェクトをv5.xxに更新後v4.xxに戻す場合、MPLAB X IDEの[Tools] > [Plugins] > [Available Plugins] > [Save As v4.xx Project]からプラグインをインストールします。プラグイン インストール後は、[Tools] > [Embedded] > [Save as MPLAB X v4.xx Project]を選択して古いバージョンでプロジェクトを保存できます。

デバイスパックについて

v4.20以前のバージョンでは、サポートデバイス(*.PIC)ファイルはそれぞれのMPLAB X IDEバージョンに組み込まれていました。そのバージョンでサポートされているデバイスを更新する事はできず、次のMPLAB X IDEバージョンまで待つ必要がありました。

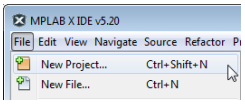
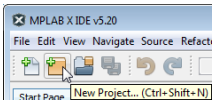
v5.00以降のバージョンでは、デバイスファイルはバージョン管理されたデバイスファミリ パックにグループ化されています。各MPLAB X IDEバージョンにはデバイスパックが付属していますが、このデバイスパックをアップデートして新しいデバイス、新しいデバイス機能サポート、デバイスバグ修正サポートを含める事ができます。

デバイスパックの詳細は以下を参照してください。

5.1「デバイスパックを使う方法」

4.1.2 [New Project]ウィザードの起動

プロジェクトはいくつかの方法で新規作成できます。

	[File] > [New Project]を選択する(または<Ctrl>+<Shift>+<N>)
	[New Project]アイコン([File]ツールバー上)
	**[Start Page]**で、**[Learn & Discover]**または**[My MPLAB X IDE]**タブを開き、[Projects] > [Create New]を選択する

4.1.3 Step 1: Choose Project

プロジェクト カテゴリを選択します。ほとんどの場合、以下を含む[Microchip Embedded]のプロジェクト タイプを選択します。

以下のオプションを使ってプロジェクト カテゴリを選択できます。

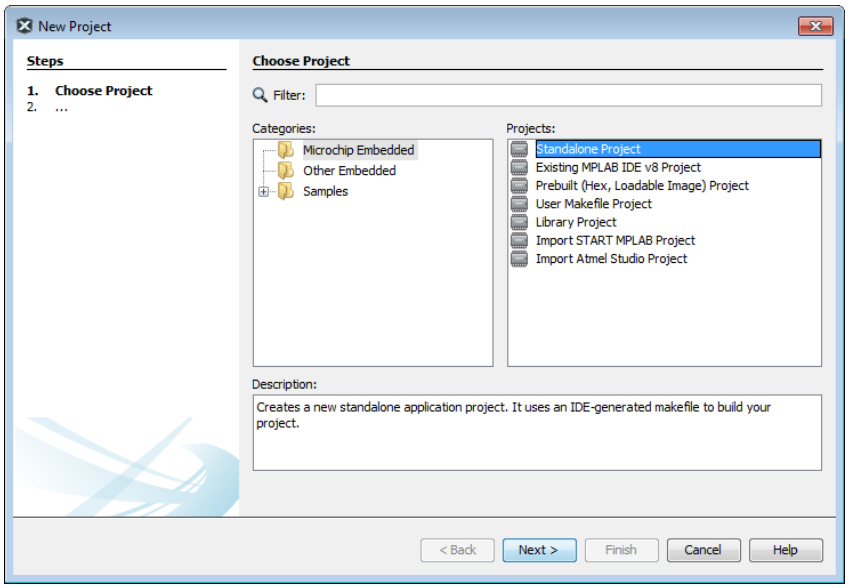
- Standalone Project – 新しいプロジェクト(C言語またはアセンブリコード)を作成します。本セクションでは、このプロジェクト タイプを解説します。
- Existing MPLAB IDE v8 Project – 既存のMPLAB IDE v8用プロジェクトをMPLAB X IDE用に変換します。詳細は5.3「MPLAB IDE v8プロジェクトのインポート」を参照してください。
- Prebuilt (Hex, Loadable Image) Project – 既存のプロジェクト イメージをMPLAB X IDEにロードします。詳細は5.4「ビルド済みのプロジェクト」を参照してください。
- User Makefile Project – 外部makeファイルを使うプロジェクトを作成します。詳細は6.6「ユーザMakefileプロジェクトの作成」を参照してください。
- Library Project – 実行用HEXファイルではなくライブラリをビルドする新しいプロジェクト(C言語またはアセンブリコード)を作成します。詳細は5.7「ライブラリ プロジェクト」を参照してください。
- Import START MPLAB Project / Import Atmel Studio Project - これらのAtmelプロジェクトをインポートします。詳細は5.8「Atmel Studio 7またはAtmel STARTプロジェクトのインポート」を参照してください。

その他に以下を選択できます。

- 5.9「その他の組み込みプロジェクト」– 他社製プロジェクト
- 5.10「サンプル プロジェクト」– 各種デバイスファミリ向けの既製のプロジェクトやプロジェクト テンプレートを選ぶ事ができます。

項目を選択したら、**[Next>]**をクリックして次のダイアログへ進みます。

図4-1　プロジェクト ウィザード – プロジェクトの選択



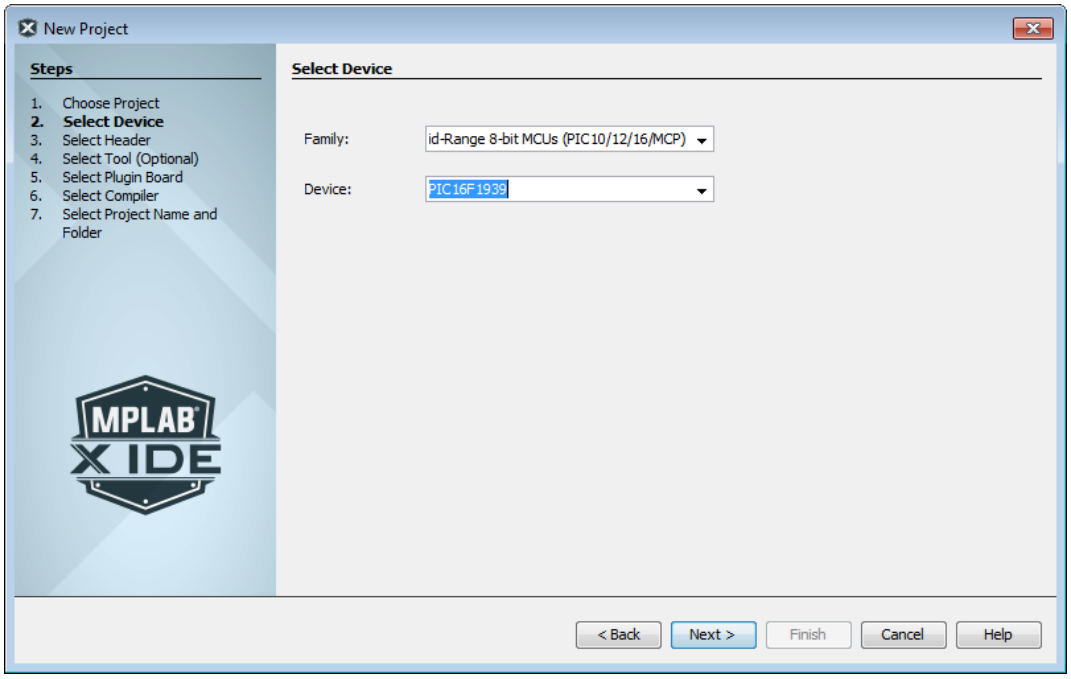
4.1.4 Step 2: Select Device

アプリケーションで使う予定のデバイスを[Device]ドロップダウン リストから選択します。選択肢を絞り込むために、最初にデバイスファミリを選択します。

LFデバイスの場合、デバイスにFタイプより低い電圧(通常は5.0 Vの代わりに3.3 V)を供給する必要があります。過電圧でデバイスを損傷する可能性がある場合、警告が表示されます。

[Next>]をクリックします。

図4-2　プロジェクト ウィザード – デバイスの選択



4.1.5　Step 3: Select Header

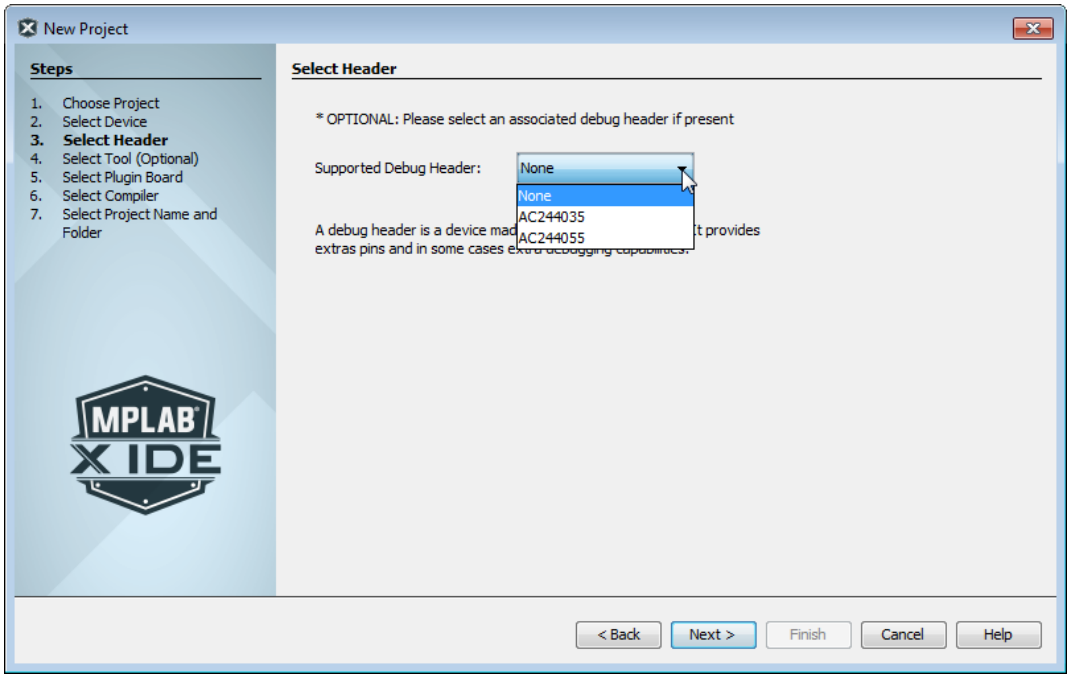
プロジェクト デバイスのヘッダを選択します。利用可能なヘッダがない場合、MPLAB X IDEはこのステップを表示しません。利用可能なヘッダがある場合、以下のヘルプトピックを参照してください。

- プロセッサ拡張パックとデバッグヘッダ
- エミュレーション拡張パックとエミュレーション ヘッダ

完了したら、**[Next>]**をクリックします。

Note:　[Project Properties]ウィンドウで後からヘッダを選択する事もできます。

図4-3　プロジェクト ウィザード – ヘッダの選択



4.1.6　Step 4: Select Tool

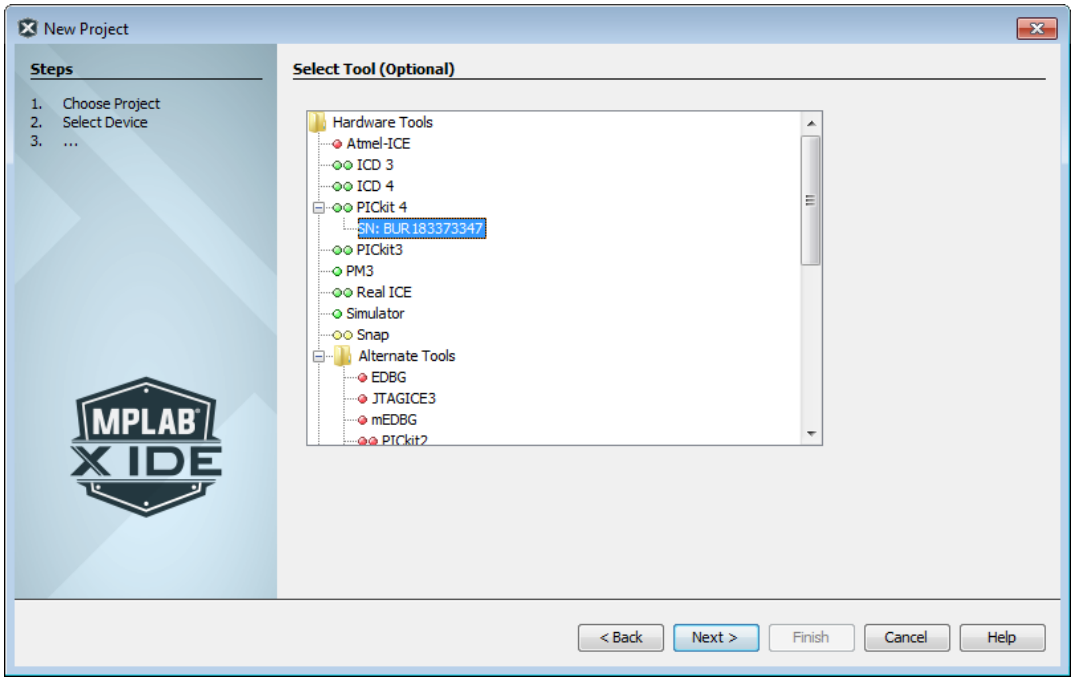
デバッグツールを選択します。

ツール名の左横の丸い色付きマークの詳細は4.1.6.1「プロジェクト ツールサポート」を参照してください。

ハードウェア ツール接続時、ツール名の下にシリアル番号(SN)が表示されます。これにより、同じハードウェア ツールを複数接続しても個別のツールを識別できます。

ツールを選択して**[Next>]**をクリックします。

図4-4　プロジェクト ウィザード – ツールの選択

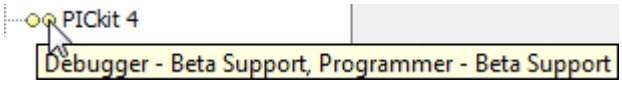


4.1.6.1 プロジェクト ツールサポート

プロジェクト デバイスのプロジェクト ツールサポートは、ツール名の左側の丸いマーク(信号灯)の色で識別できます。

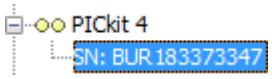
マーク	色	サポート
●	緑	完全サポート(実装と完全なテスト済み)
●	黄	ベータ版サポート(実装済み、ただし完全なテストは未完)
●	赤	未サポート

色を識別できない場合、マウスカーソルをマークの上に置くと、サポートに関する情報を表示できます。



デバッガまたはプログラマとして使えるハードウェア ツールの場合、ツール名の横に2つの信号灯が表示されます。左側の信号灯はデバッグサポート、右側の信号灯はプログラマ サポートを意味します。

ハードウェア ツール接続時も、ツール名の下にシリアル番号(SN)が表示されます。これにより、同じハードウェア ツールを複数接続していても個別のツールを識別できます。



お使いのサードパーティ製ハードウェア ツールがここに表示されない場合、そのツールが適切にインストールされているかどうか確認します。詳細は6.8「サードパーティ製ハードウェア ツール」を参照してください。

4.1.7 Step 5: Select Plugin Board

MPLAB REAL ICEインサーキット エミュレータのプラグインボードを選択します。このエミュレータをデバッグツールとして選択していない場合、MPLAB X IDEはこのステップを表示しません。

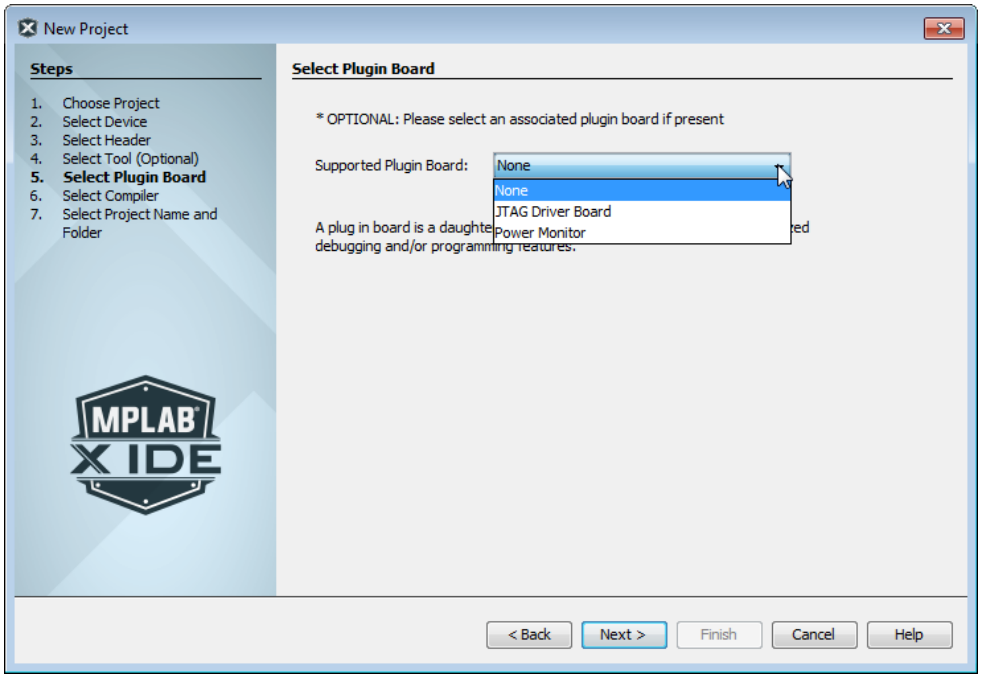
プラグインボードとはエミュレータのドライバボード スロットに挿入する基板で、デバッグ機能を追加できます。

表4-2 エミュレータ プラグインボード

Supported Plugin Board	ボードの説明
None	標準通信ドライバボード
None	高速通信ドライバボード
JTAG Driver Board	JTAGアダプタボード
Power Monitor Board	電源モニタボード (ロジック プローブ コネクタにも挿入する必要があります)

ツールを選択してから**[Next>]**をクリックします。

図4-5　プロジェクト ウィザード – プラグインボードの選択



4.1.8　Step 6: Select Compiler

言語ツール(Cコンパイラまたはアセンブラ)を選択します。選択可能な言語ツールは以下のように表示されます。

ツール略称(ツールバージョン) [ツール実行パス]

ツール名の左横の丸い色付きマーク(信号灯)の詳細は4.1.6.1「プロジェクト ツールサポート」を参照してください。

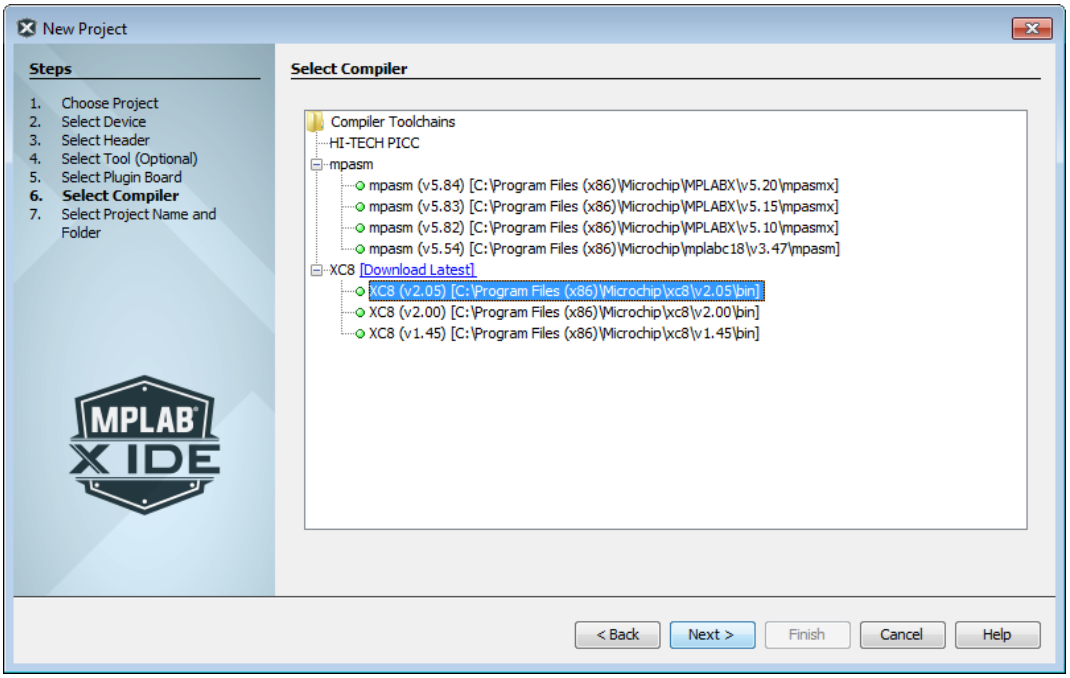
ここに必要な言語ツールが表示されない場合、またはサポートツールが1つも表示されない場合、ツールがインストール済みであるかどうか確認します。2.5「言語ツールのインストール」を参照してください。

Note:　AVR GNUおよびArm GNUツールチェーンの場合、MPLAB X IDEが検索可能な場所にコンパイラをインストールします(例：　Windowsの場合、`C:\Program Files (x86)\Microchip`)。または、パスを入力する必要があります。4.7「言語ツールのパスの指定」を参照してください。

ツールは[Build Tools]の下に表示されるが、サポートが表示されない場合、そのプロジェクト デバイスはツールまたはツールバージョンによってサポートされていない可能性があります。そのデバイスをサポートする他の言語ツールを選ぶかインストールする必要があります。または、[Download Latest］でMPLAB XCコンパイラの最新バージョンをダウンロードしてインストールする事もできます。ただし、このバージョンがそのデバイスをサポートしている必要があります。

ツールを選択して**[Next>]**をクリックします。

図4-6　プロジェクト ウィザード – コンパイラ(またはその他の言語ツール)の選択



4.1.8.1　サポートされている言語ツール

下表に、言語ツールに対する現行およびレガシーMPLAB X IDEサポートを示します。現行の言語ツールは新規デバイスをサポートします。レガシーツールは新規デバイスをサポートしません。

表4-3　Microchip社製言語ツール – 現行

ツールチェーン	正式名	サポートするデバイス
XC8	MPLAB XC8 C Compiler for PIC MCUs	8ビットPIC® MCU
	MPLAB XC8 C Compiler for AVR MCUs	8ビットAVR® MCU
XC16	MPLAB XC16 C Compiler	16ビットPIC MCU、dsPIC® DSC
XC32	MPLAB XC32 C/C++ Compiler for PIC32M MCUs	32ビットPIC32M MCU
	MPLAB XC32 C/C++ Compiler for PIC32C/SAM MCUs	32ビットPIC32C/SAM MCU
Arm® GNU	Arm GNU C Compiler	32ビットArm MCU
AVR GNU	AVR GNU C Compiler	8/32ビットAVR MCU
MPASM	MPASM Assembler, MPLINK Object Linker and Utilities	8ビットPIC MCU

表4-4　Microchip社製言語ツール – レガシー

ツールチェーン	正式名
8ビットPIC MCU向け言語ツール	
C18*	MPLAB C Compiler for PIC18 MCUs
HI-TECH PICC	HI-TECH C Compiler for PIC10/12/16 MCUs
HI-TECH PICC18	HI-TECH C Compiler for PIC18 MCUs
16ビットPIC MCU、dsPIC DSC向け言語ツール	
ASM30**	MPLAB Assembler, Object Linker and Utilities for PIC24 MCUs and dsPIC DSCs

(続き)

ツールチェーン	正式名
C30	PIC24 MCUおよびdsPIC DSC向けMPLAB Cコンパイラ
C24	MPLAB C Compiler for PIC24 MCUs (C30のサブセット)
dsPIC	MPLAB C Compiler for dsPIC DSCs (C30のサブセット)
HI-TECH DSPICC	HI-TECH C Compiler for PIC24 MCUs and dsPIC DSCs
32ビットPIC MCU向け言語ツール	
C32	MPLAB C Compiler for PIC32 MCUs
HI-TECH PICC32	HI-TECH C Compiler for PIC32 MCUs
* ほとんどのコンパイラには、アセンブラ、リンカ、ユーティリティが付属しています。しかし、MPLAB C18はMPASMツールチェーンでサポートされています。 ** MPLAB X IDEのv1.30以降には付属していません。16ビットコンパイラに付属するアセンブラをお使いください。	

各言語ツールの詳細はそれぞれの文書を参照してください。

サードパーティ製言語ツールチェーン(CCS等)については、[Start Page]の「Release Notes and Support Documentaion」より[Readme for Third Party Tools.htm]ファイルを参照してください。

4.1.8.2 最新MPLAB XCコンパイラのダウンロード

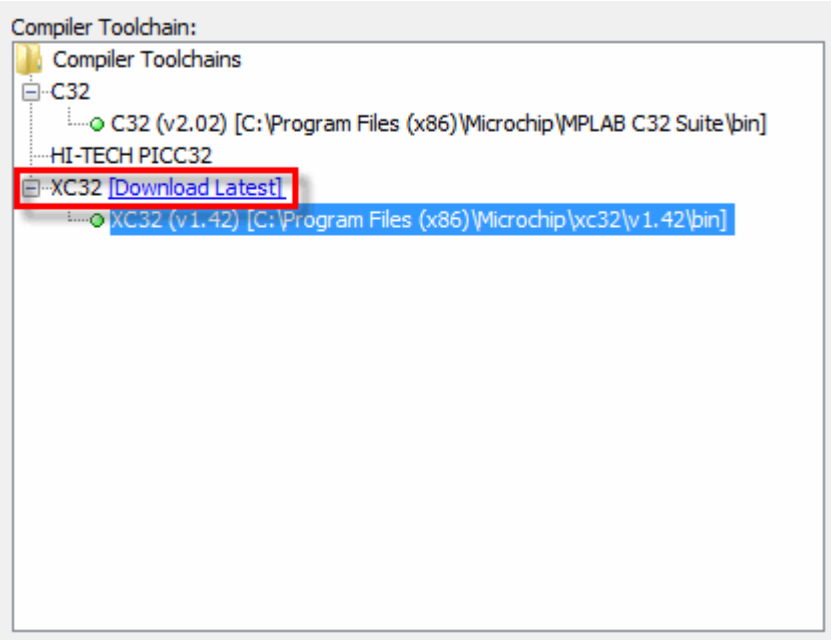
[Download Latest]リンクをクリックすると、MPLAB XC Cコンパイラの最新バージョンをダウンロードできます。OSによっては、コンパイラのインストールとコンパイラ選択ツリーの更新も行います。

[Download Latest]リンクはStep 6.[Select Compiler]、[Compiler Toolchains] > [XCn] (n = 8、16、32)の下の[New Project]ウィザードにも表示されます。

Note: インターネットに接続していない場合、このリンクは表示されません。

プロジェクトが作成されると、このリンクは[Project Properties]ウィンドウに表示されます。

図4-7 [Download Latest]



[Download Latest]リンクをクリックすると、以下の手順が実行されます。

1. メッセージ ダイアログが開き、実行内容を表示します。この時点ではダウンロードとインストールをキャンセルする事ができます。
2. 最新コンパイラが既にインストールされている場合、メッセージ ダイアログにより続行するか確認します。
3. コンパイラ インストーラの保存先を選択します。
4. ダウンロード中は、ダウンロードの残り量を示す進捗ウィンドウが表示されます。
5. WindowsおよびMac OS: インストーラのダウンロードが完了したらコンパイラのインストールが始まります。コンパイラがインストールされたら、MPLAB X IDEはそのコンパイラを選択ツリーへ追加します。追加できない場合、MPLAB X IDEにコンパイラを手動で追加する必要がある旨の通知が表示されます(4.7「言語ツールのパスの指定」)。
6. Linux: ダウンロードしたコンパイラを手動でインストールする必要があります。追加されると、MPLAB X IDEが認識するはずです。認識しない場合、MPLAB X IDEにコンパイラを手動で追加する必要があります(4.7「言語ツールのパスの指定」)。

4.1.9 Step 7: Select Project Name and Folder

プロジェクトの名前と保存場所、その他のプロジェクト設定を選択します。

プロジェクト名を入力します。拡張子として.X が使われます。[Project Folder]テキストボックスを参照してください。フォルダの場所を参照します。プロジェクト フォルダを新規作成する事もできます。既定値では、プロジェクトは以下のパスに保存されます。

- Windows 7以降 – `C:\Users\UserName\MPLABXProjects`
- Linux – `/home/UserName/MPLABXProjects`
- Mac – `/Users/UserName/MPLABXProjects`

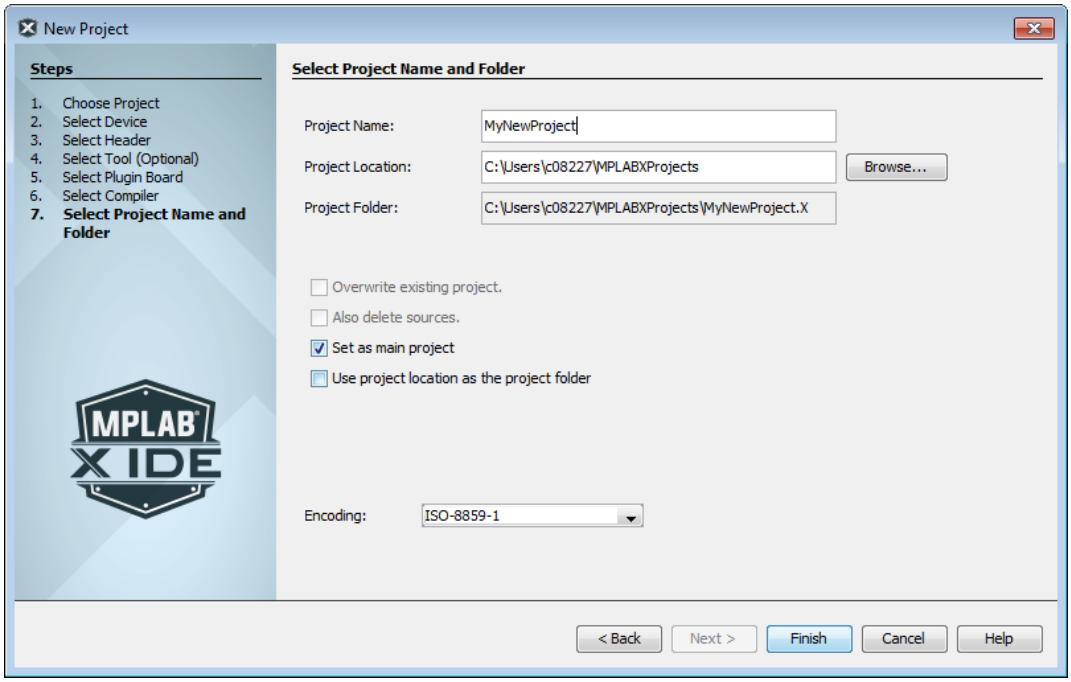
表4-5　その他の機能

Set as main project	このプロジェクトをメイン プロジェクトに指定します。
Use project location as project folder	プロジェクトの保存場所と同じフォルダ内にプロジェクトを作成します。 これは、MPLAB IDE v8プロジェクトをインポートする際、MPLAB X IDEのプロジェクトを同じフォルダにする場合(ビルドも同様にする場合)に便利です。 MPLAB IDE v8が`.mcp`ファイルを使ってプロジェクト情報を保存するのに対し、MPLAB X IDEはフォルダを使います。従って、MPLAB IDE v8プロジェクト(`.mcp`ファイル)とフォルダを共有させる事ができるMPLAB X IDEプロジェクトは1つだけです。
Encoding	プロジェクトのエンコードを選択します。既定値の文字セットはISO-8859-1 (Latin 1)です。この選択によって、コード構文の色分け設定が決まります。色分けは[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択し、**[Fonts and Colors]**ボタンをクリックして、**[Syntax]**タブより編集できます。

[Finish]をクリックすると、プロジェクトの新規作成が完了します。

Note: コンフィグレーション タイプを変更する事でスタンドアロン (アプリケーション) プロジェクトをライブラリ プロジェクトに変更できます。詳細は4.10.1「プロジェクト コンフィグレーション タイプの変更」を参照してください。

図4-8　プロジェクト ウィザード – プロジェクト名と保存先フォルダの選択



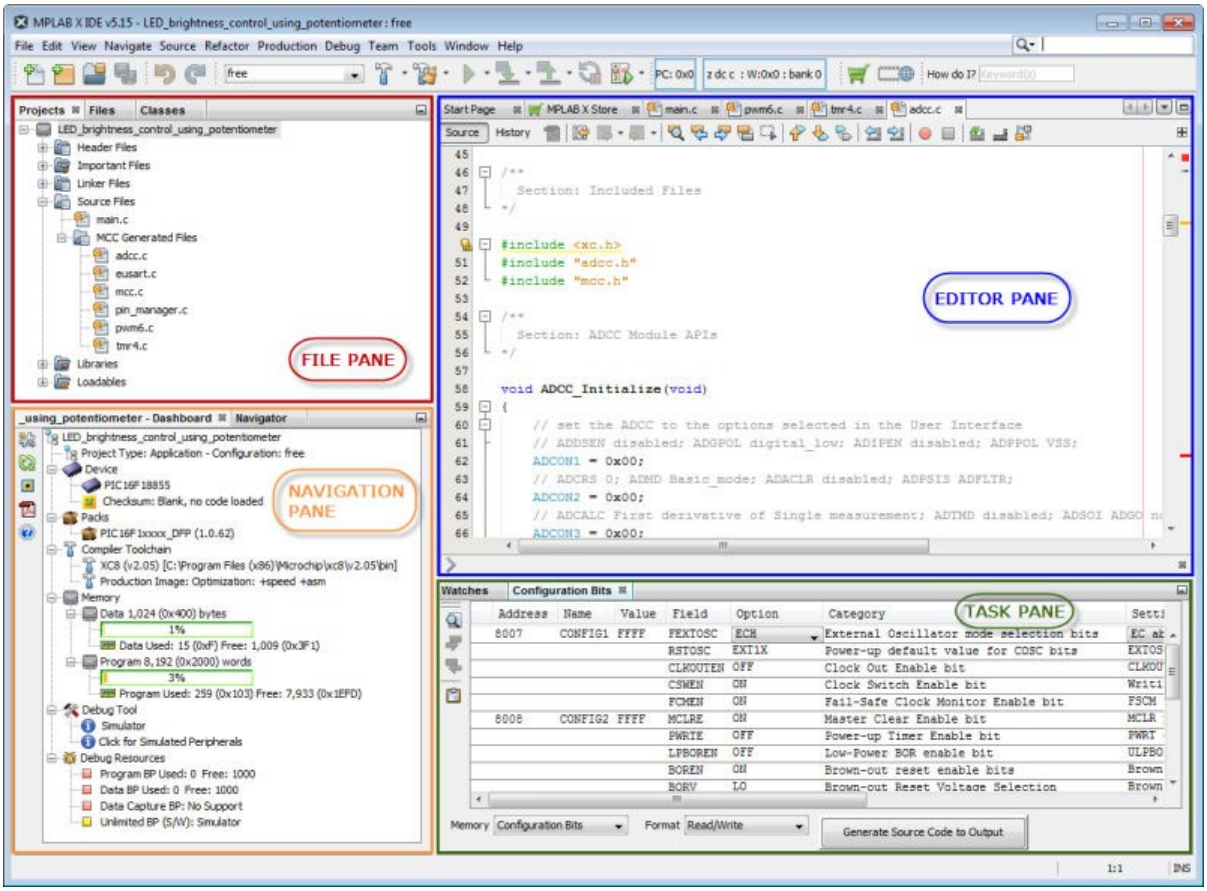
4.2　プロジェクト作成後のデスクトップ画面

MPLAB X IDEのデスクトップは、4つの**ペイン**に分割されます。プロジェクトの作成が完了すると、これらのペインにいくつかの**ウィンドウ**が表示されます。

デスクトップ ペイン

- **[File]ペイン** – ファイルに関する情報を表示します。
  - [Projects]ウィンドウは、ファイルがカテゴリごとにグループ分けされたプロジェクト ツリーを表示します。
  - [Files]ウィンドウは、コンピュータ上のフォルダ構成に従って、プロジェクト ファイルを表示します。
  - [Classes]ウィンドウは、コード内の全てのクラスと、それらの関数、変数、定数を表示します。その宣言部分を見るには、項目をダブルクリックします。
- **[Navigator]ペイン** – [File]ペイン内の項目に関する情報を表示します。
  - [Dashboard]ウィンドウは、選択されているプロジェクトに関する情報を表示します。
  - [Navigator]ウィンドウは、選択されているプロジェクト ファイル内のシンボルと変数に関する情報を表示します。
- **[Editor]ペイン** – プロジェクト ファイルの内容を表示して編集します。ここには[Start Page]と[MPLAB X Store]も表示されます。
- **[Task]ペイン** – アプリケーションのビルド、デバッグ、実行からのタスク出力を表示します。

図4-9　MPLAB X IDEのデスクトップ - ペイン



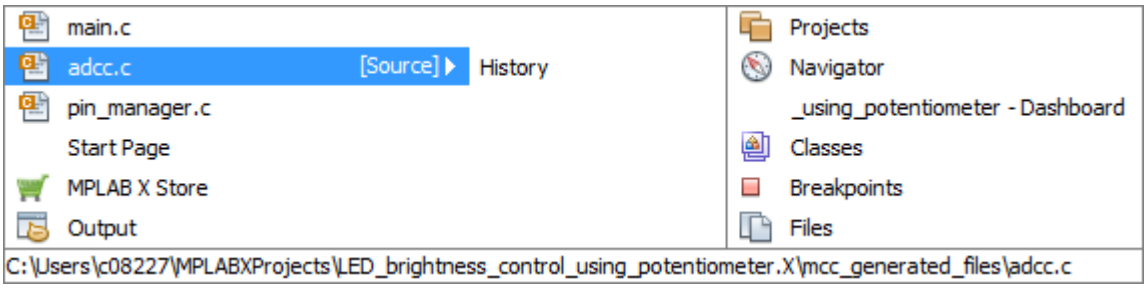
デスクトップ ウィンドウ

ペインはIDEインターフェイスの各セクションを説明するために便利ですが、ウィンドウはペイン間で移動できます。そのため、ペインが必ずしもウィンドウの内容を表しているとは限りません。従って、ほとんどの文書はウィンドウのみを説明しています。

デスクトップでのウィンドウ間の移動は以下のように行います。

- ウィンドウをクリックしてフォーカスします。
- 1つのウィンドウで<Ctrl>を押しながら<Tab>を押すと、開いている他のウィンドウにフォーカスを変更できます。ポップアップ リストからウィンドウを選択します。

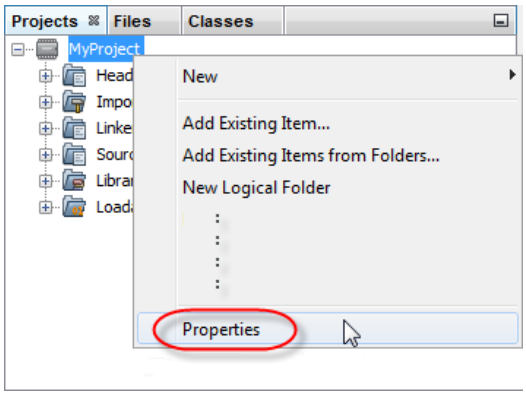
図4-10　ウィンドウ フォーカスの切り換え



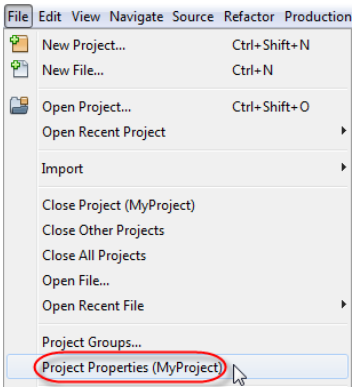
4.3　[Project Properties]ウィンドウを開く

プロジェクト作成後は、[Project Properties]ウィンドウでプロパティを表示、編集できます。このウィンドウは以下のいずれかの方法で開く事ができます。

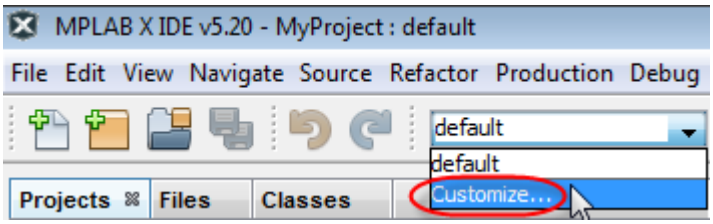
1. [File]ペインの[Projects]ウィンドウ内でプロジェクト名を右クリックし、[Properties]を選択します。



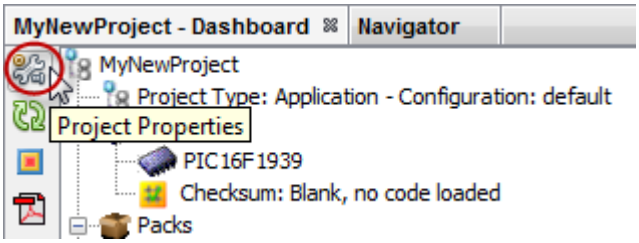
2. [File]ペインの[Projects]ウィンドウ内でプロジェクト名を左クリックし、メニューから[File] > [Project Properties]を選択します。



3. [Projects]ウィンドウ内でプロジェクト名をクリックし、[Set Project Configuration]ドロップダウン メニューをクリックして[Customize]を選択します。



4. [Dashboard]ウィンドウで[Project Properties]アイコンをクリックします。



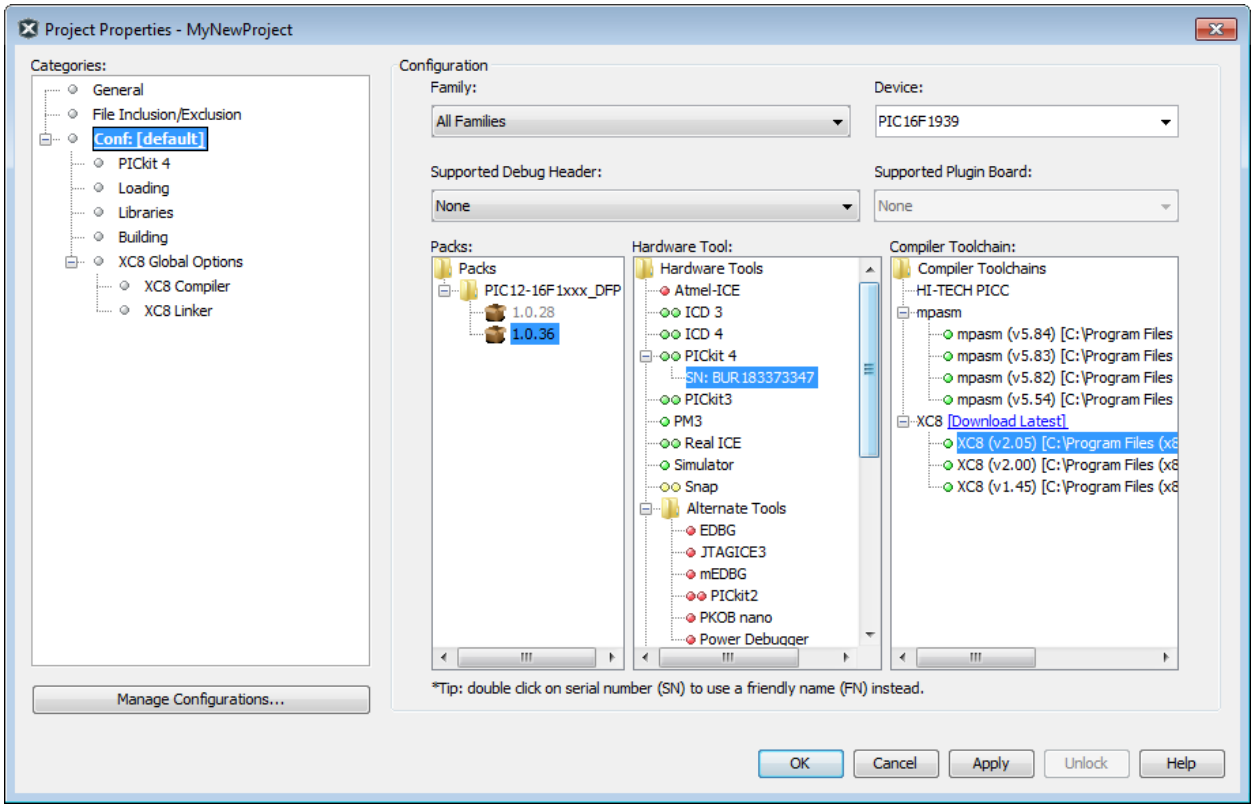
4.4 プロジェクト プロパティの表示と変更

開いたダイアログで、[Conf: [default]]カテゴリをクリックすると、プロジェクト コンフィグレーション (デバイス、デバッガ/プログラマツール、言語ツール等)が表示されます。

Categories	作成後のプロジェクト コンフィグレーションは[default]として設定されています。**[Manage Configurations]**ボタンをクリックすると、各種コンフィグレーションを作成できます。6.4「複数コンフィグレーションの使い方」を参照してください。

Device	シミュレートするデバイス、またはボード上のデバイスです。 デバイス変更時(特に別のファミリのデバイスに変更する場合)はパック、ハードウェア ツール、コンパイラを更新する必要があるため、注意が必要です。
Packs	選択したデバイスのデバイスファミリ パック(DFP)バージョンです。MPLAB X IDEの最新バージョンには、DFPの最新バージョンが付属します。しかし、別のバージョンに変更する事もできます。5.1「デバイスパックを使う方法」参照
Hardware Tool	サポートされているハードウェア デバッグツール(またはシミュレータ)を選択します。4.1.6.1「プロジェクト ツールサポート」参照
Compiler Toolchain	サポートされている言語ツール(通常はコンパイラ)を選択します。4.1.6.1「プロジェクト ツールサポート」参照 MPLAB XC Cコンパイラの最新バージョンは、[Download Latest]リンクをクリックすると入手できます。 4.1.8.2「最新MPLAB XCコンパイラのダウンロード」参照

図4-11　[Project Properties]ウィンドウ - 既定値コンフィグレーション

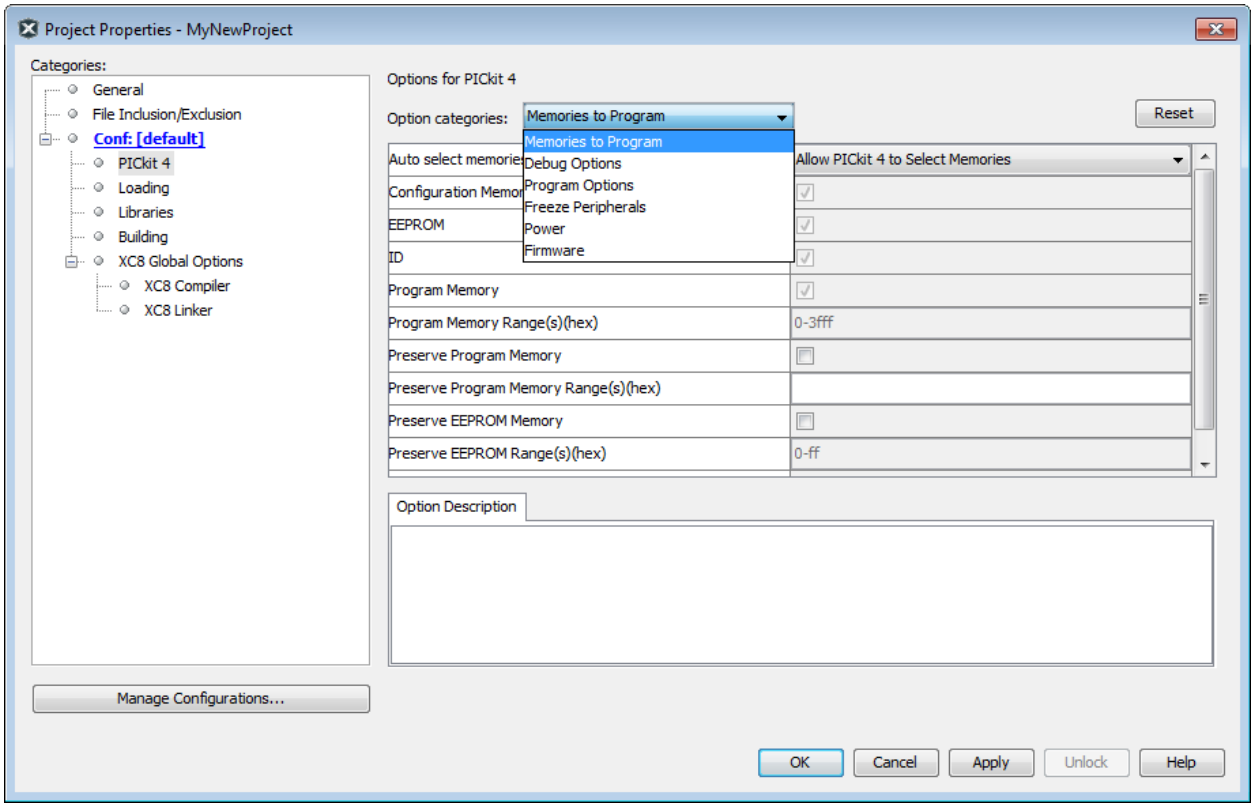


4.5　デバッガ/プログラマツール オプションを設定または変更する方法

[Project Properties]ウィンドウでは、ツールのオプションも設定します。

ハードウェア ツールまたはシミュレータ(Conf: [default]の下)をクリックして、関連するオプション メニューを表示させます。これらのオプションの意味は、各ツールの文書を参照してください。

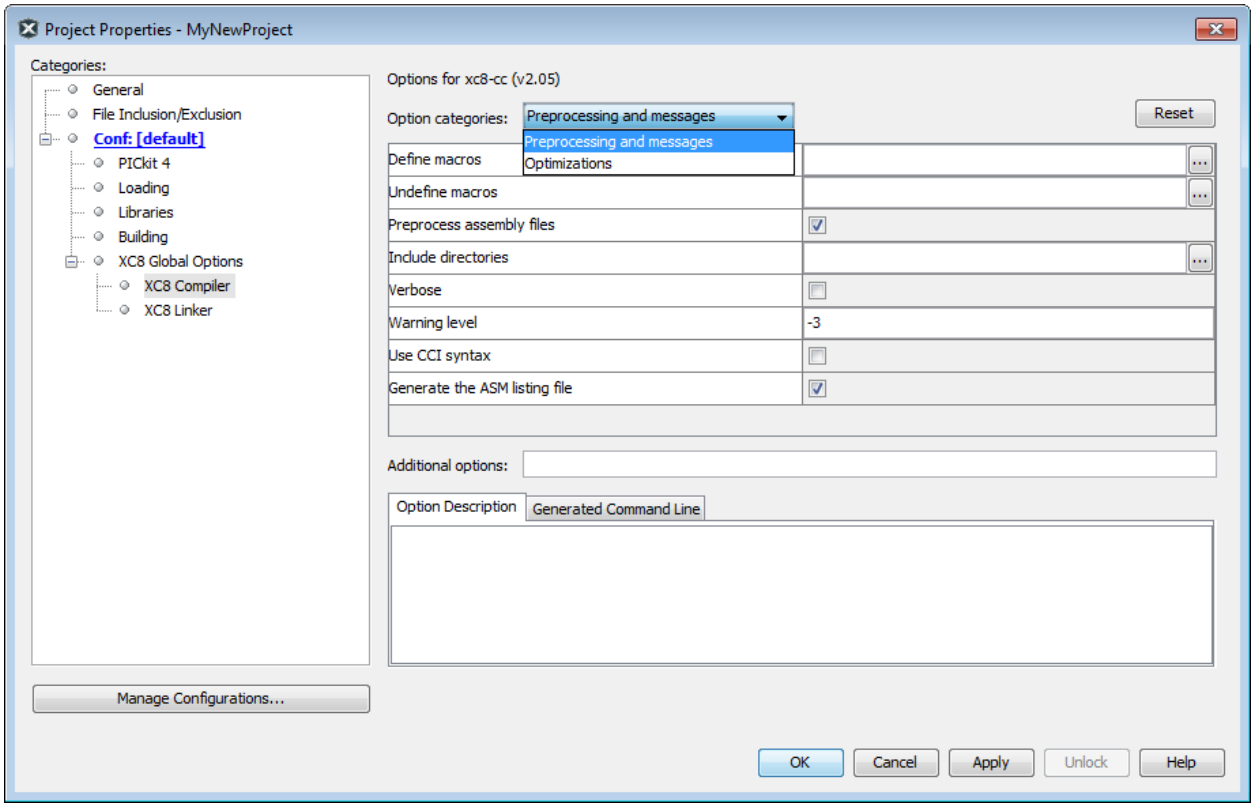
図4-12　サンプルツールの設定ページ



4.6　言語ツールオプションを設定または変更する方法

[Project Properties]ウィンドウの言語ツールをクリックして、設定オプションを表示させます。これらのオプションの意味は、言語ツールの文書を参照してください。

図4-13　言語ツールの設定ページ



IDEオプションとコマンドライン オプション

IDEに入力するオプションは、コマンドラインで入力するオプションとフォーマットが異なる場合があります。[Option Description]タブのオプション名をクリックするとオプションの詳細が表示されます。また、オプションデータを入力または選択してから、[Generated Command Line]タブをクリックして内容を表示させ、オプションで生成されたコマンドライン コードが正しい事を確認します。

グローバル オプション

グローバル オプションを設定すると、言語ツールスイート内の全ての言語ツールに適用されます。また、[User Comments]タブを使うと特定のオプションを選択した理由を入力できます。

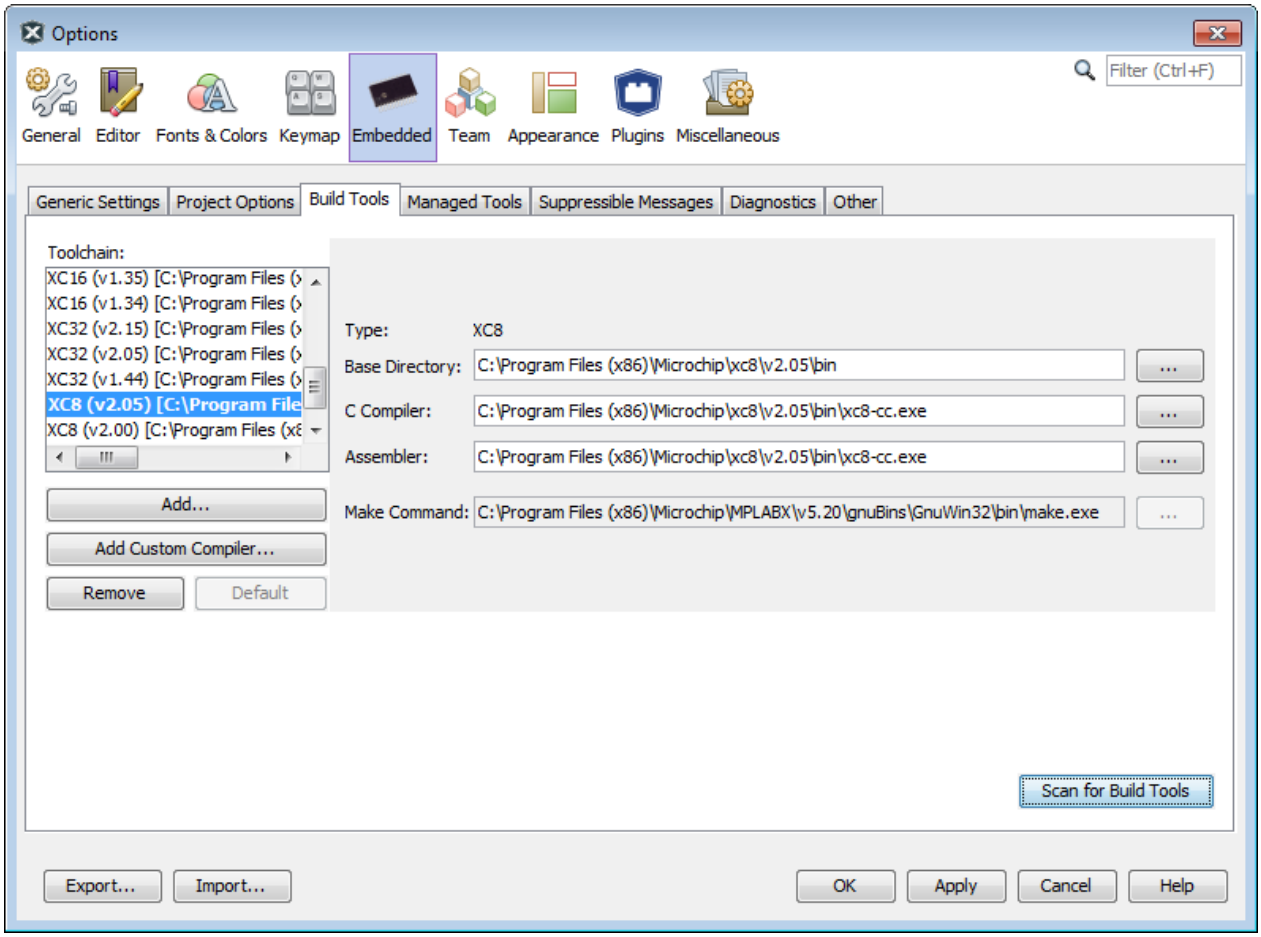
4.7　言語ツールのパスの指定

MPLAB X IDEで利用できる言語ツールとそれらのパスを表示/変更するには、以下の手順に従います。

- macOSの場合: メイン メニューバーから
  [MPLAB X IDE] > [Preferences] > [Embedded] > [Build Tools]を選択してビルドツールにアクセスする
- その他のOSの場合:
  [Tools] > [Options] > [Embedded] > [Build Tools]を選択してビルドツールにアクセスする

ウィンドウには、インストールされている全てのツールチェーンが自動的に表示されます。12.16.3「[Build Options] タブ」を参照してください。

図4-14　言語ツール (コンパイラ)のパス



4.7.1　ツールチェーンの追加または変更

[Toolchain]欄にインストール済みのツールが表示されない場合、以下を試します。

- [Scan for Build Tools]ボタン – 環境パスをスキャンし、PCにインストールされている言語ツールの一覧を表示します。
- [Add]ボタン – ツールの実行ファイルを保存したフォルダ (ベースフォルダ)へのパスを入力する事により、手動でリストにツールを追加します。一般的に実行ファイルは、ツールをインストールしたフォルダ内の「bin」サブフォルダに保存されています。

同じコンパイラの複数のバージョンをインストールしている場合、リストからいずれか1つのバージョンを選択します。

ツールのパスを変更するには、新しいパスを手入力するか、[...]ボタンをクリックして選択します。

4.7.2　ツールチェーンの削除

既存のツールチェーンを削除するには、以下の手順に従います。

1. リストからツールチェーンをクリックして選択します。
2. [Remove]ボタンをクリックします。

この操作では、コンパイラはPCからアンインストールされません。MPLAB X IDEがこのコンパイラを認識しなくなるだけです。ツールチェーンを再び追加する方法は、4.7.1「ツールチェーンの追加または変更」を参照してください。

コンパイラをPCからアンインストールする場合、上のようにMPLAB X IDE内のコンパイラの参照を最初に削除します。その後、コンパイラをアンインストールします。コンパイラを先にアンインストールし、MPLAB X IDEを起動した場合、ツールチェーン パスが赤で表示されます。このアンインストール済みコンパイラへの参照を削除するには、既述の手順で行います。

4.7.3 ツールチェーン パス

MPLAB X IDEは、既定値のインストールパスとPATH環境変数でツールチェーンを検索します。Windows 64ビットOSにおけるMPLAB XC16の既定値パスの例を以下に示します。

```
C:\Program Files (x86)\Microchip\xc16\vx.xx\bin
```

コンパイラをインストールする際、インストール先を選択できます。以下のどちらかを行う事ができます。

- 既定値の場所にインストールする

または

- 別の場所にインストールし、この場所をPATHに追加する

既定値以外の場所にインストールする場合、その場所をPATHに追加する必要があります。

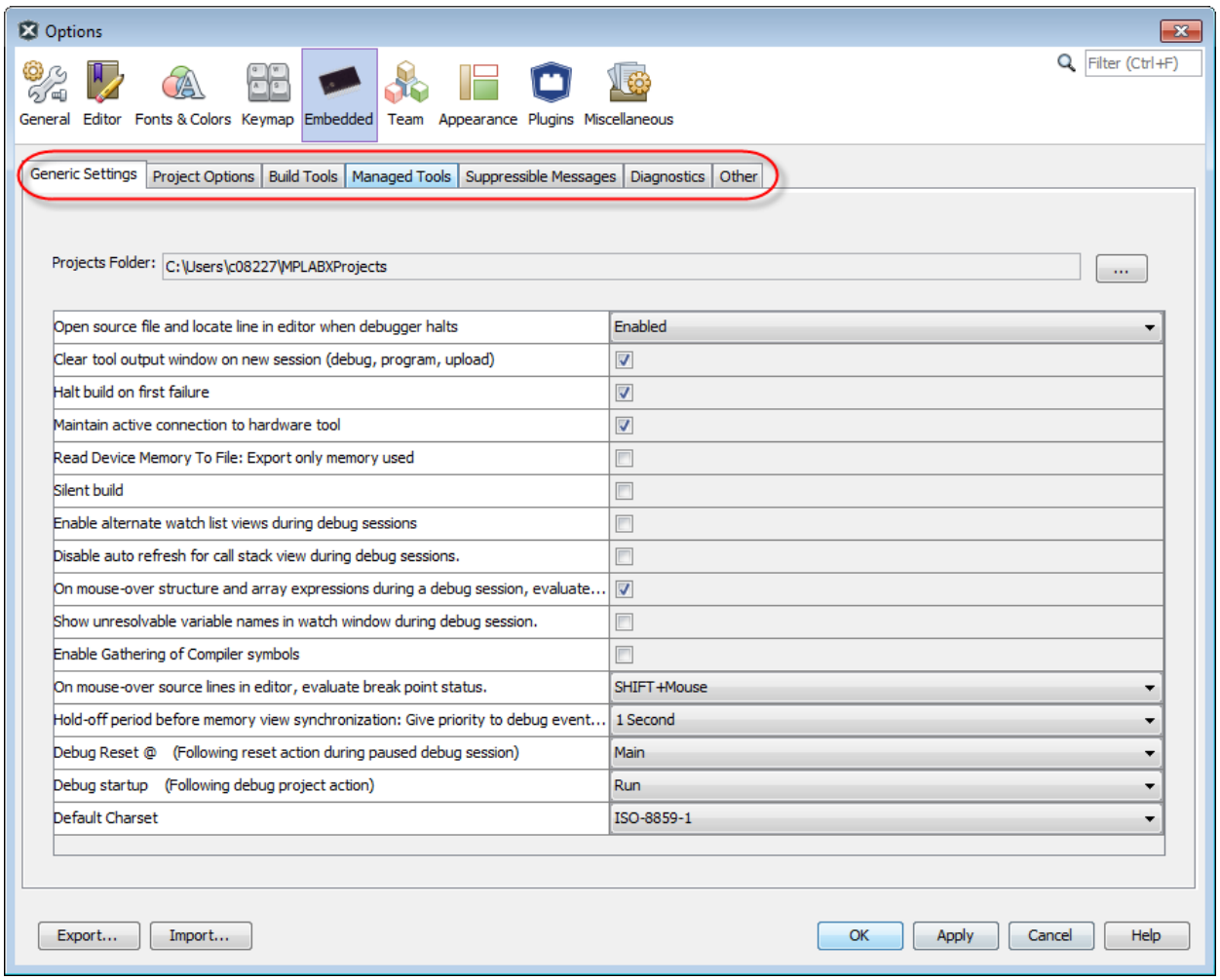
PATHに新規の場所を追加しない場合、MPLAB X IDEでコンパイラの場所を指定できます([Tools] > [Options]で[Build Tools]タブを開き、[Embedded]ボタンをクリック)。

最後に、インストール場所をPATH変数に手動で追加できます。

MPLAB X IDEがツールチェーンを認識しているにもかかわらずパスが見つからない場合、そのパスは赤の文字で表示されます。

4.8 その他のツールオプションの設定

ビルド用パス以外の組み込みオプションも設定できます。[Options]ウィンドウの[Embedded]カテゴリでタブを選択できます。12.16「[Tools] > [Options]ウィンドウ、[Embedded]」を参照してください。



4.9 プロジェクトへのファイルの追加

プロジェクトを新規作成したら、アプリケーションを作成するためにプロジェクトにファイルを追加します。コードを書くためにファイルを新規作成するか、既存のファイルをインポートまたは参照できます。エディタを使うと、言語ツール固有の色分けやその他の機能を使ってファイルの内容を作成または更新できます。

ファイルリンクの順序

リンク処理の順序は、プロジェクトにファイルを追加した順序で決まります。MPLAB IDE v8からプロジェクトをインポートする場合、その順序は.mcpファイルで決まります。プロジェクトに新規に追加したファイルが最後にリンクされます。ファイルを削除した後で再度追加した場合、そのファイルはリンクリストの末尾に移動します。

ライブラリとオブジェクト ファイルのリンク

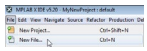
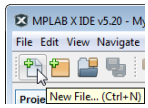
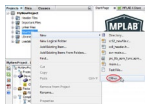
リンカが使うライブラリとオブジェクト ファイルは参照できます。

ビルドからのファイルの除外

ファイルまたはフォルダレベルのオプションを設定する事で、ビルドからファイルを除外できます。

4.9.1 [New File]ウィザードの起動

ファイルを新規作成してプロジェクトにコードを追加するには、[New File]ウィザードを使います。このウィザードはいくつかの方法で起動できます。

	[File] > [New File](または<Ctrl>+<N>)
	[New File]アイコン([File]ツールバー上)
	[Projects]/[Files]ウィンドウ内のプロジェクト フォルダ(例: [Source Files])を右クリックし、[New] > [Other]を選択する

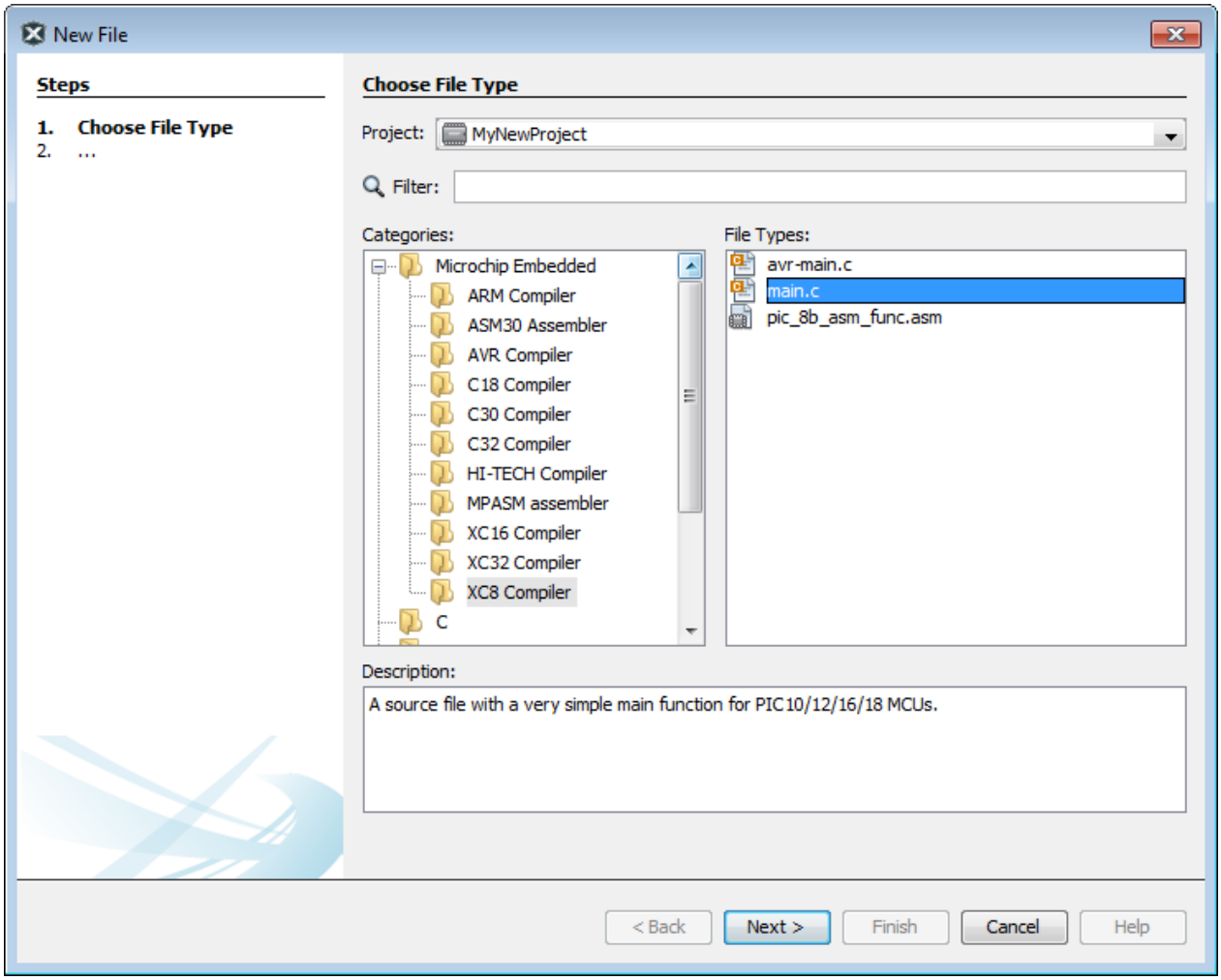
4.9.2 ファイルの新規作成

[New File]ウィザードの手順に従ってファイルを新規作成します。

Step 1. Choose File Type

[Microchip Embedded]を展開して、その中のファイルカテゴリを選択します。次にファイルタイプを選択します。**[Next>]**をクリックします。

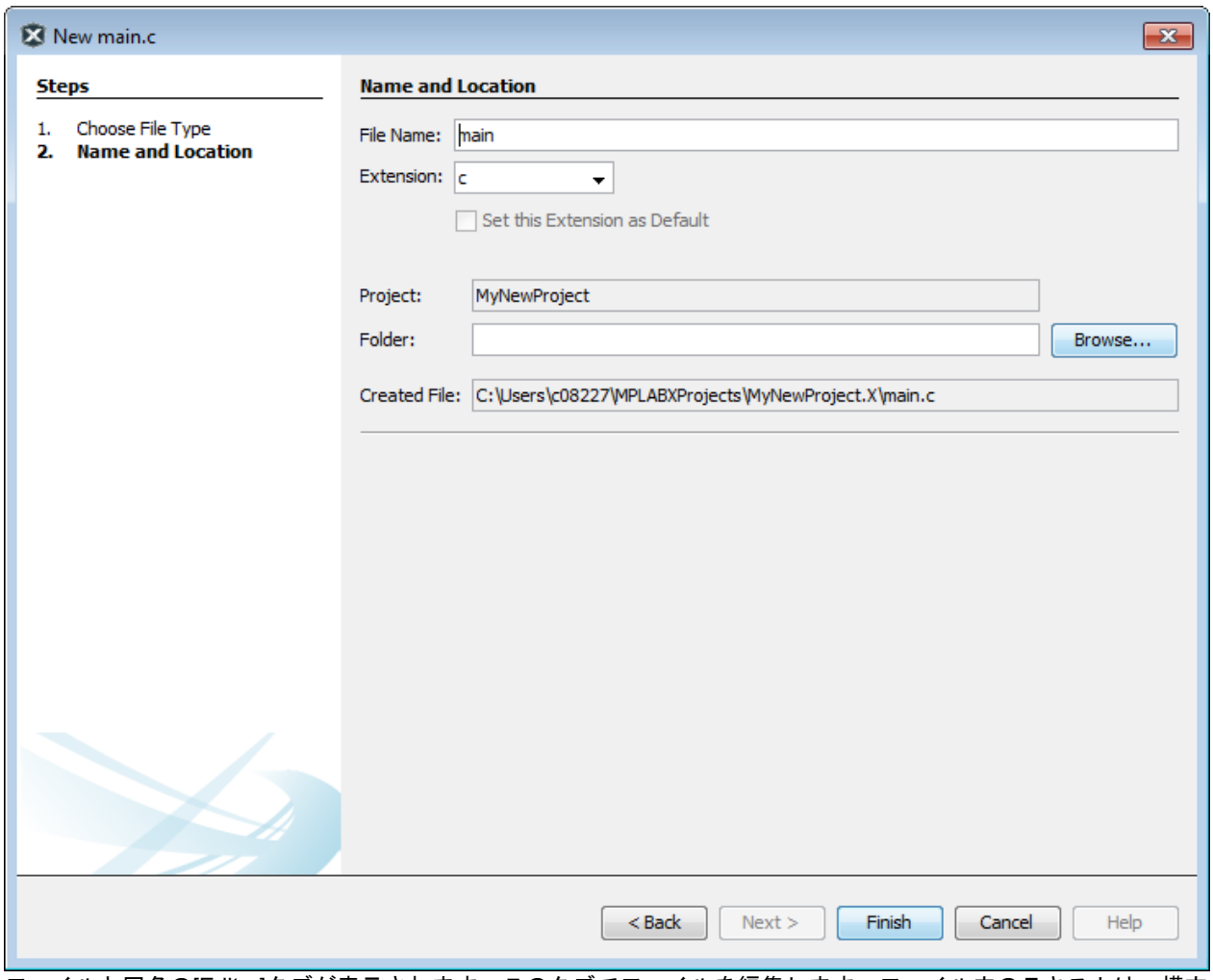
図4-15　ファイル ウィザード – カテゴリとタイプの選択



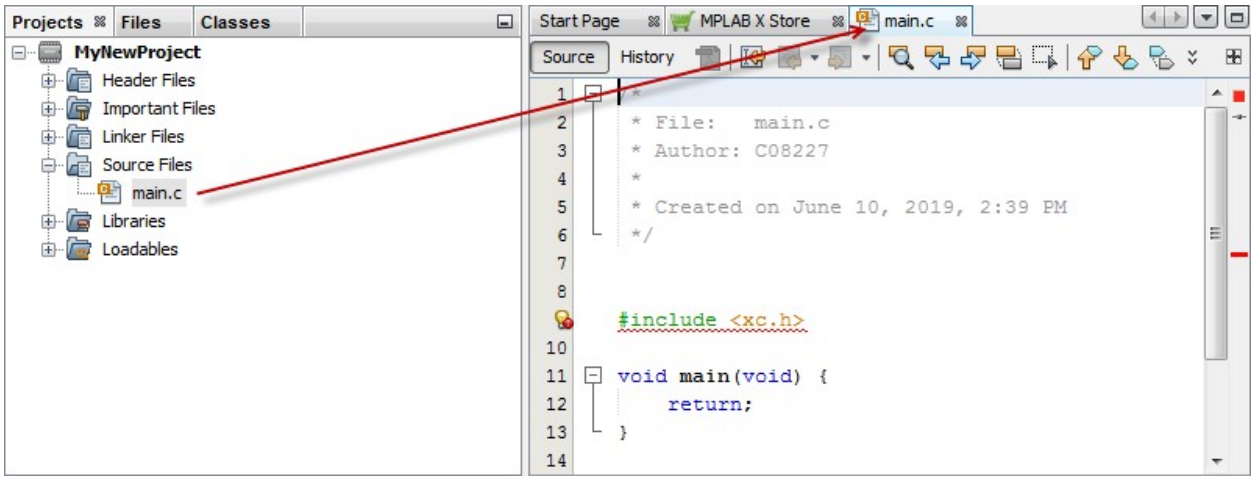
Step 2. Name and Location

ファイルに名前を付け、そのファイルをプロジェクト フォルダに保存します。**[Finish]**をクリックします。

図4-16　ファイル ウィザード – 関連付けるプロジェクトの選択



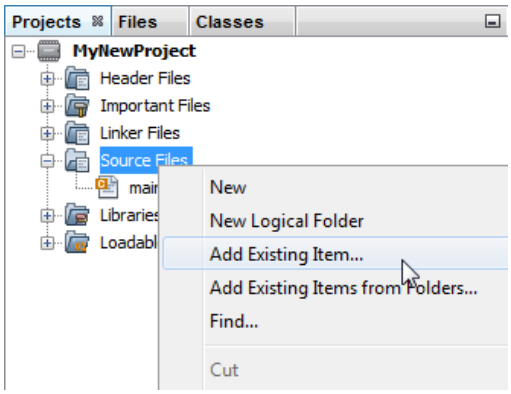
ファイルと同名の[Editor]タブが表示されます。このタブでファイルを編集します。ファイル内のテキストは、構文に基づいて色分け表示されます(色分けはファイルタイプによって異なります)。



4.9.3　既存ファイルのプロジェクトへの追加

以下のどちらかの方法により、既存ファイルをプロジェクトに追加できます。

- [Project/File]ウィンドウ内のプロジェクトを右クリックし、[Add Existing Item]を選択します。追加後はそのファイルをフォルダにドラッグ&ドロップできます。
- [Project/File]ウィンドウ内のフォルダ(例: [Source Files])を右クリックし、[Add Existing Item]を選択します。



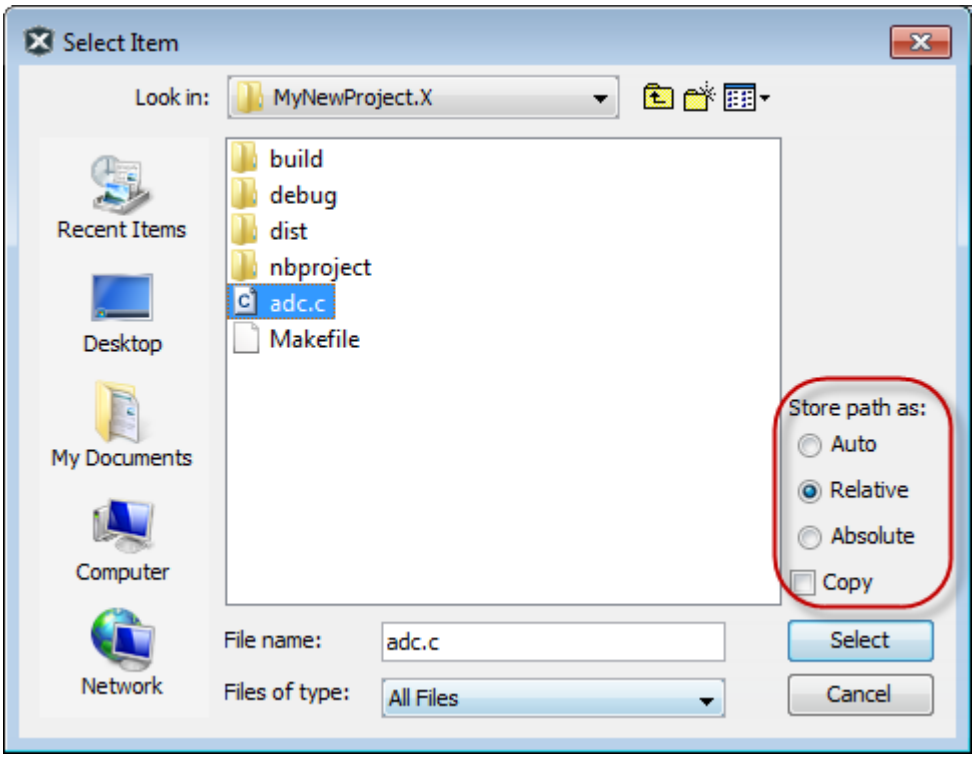
4.9.3.1 プロジェクト フォルダ内のファイルの追加

メイン プロジェクト フォルダ内のファイルを追加する際は、追加方法を以下から選択できます。

- Auto – MPLAB X IDEがファイルのパスの指定方法を自動的に選択します。
- Relative – ファイルのパスをプロジェクト フォルダに対して相対的に指定します(最も移植性に優れます)。
- Absolute – ファイルのパスを絶対パスで指定します。
- Copy – 指定したファイルをプロジェクト フォルダにコピーします。

パスモードを選択すると、MPLAB X IDEがこの設定を記憶します。変更する場合、
12.16.2「[Project Options]タブ」の[File Path Mode]を参照してください。

ファイルは[Projects]ウィンドウに表示されます。ファイルと同名の[Editor]タブも表示されます。



4.9.3.2 プロジェクト フォルダ外のファイルの追加

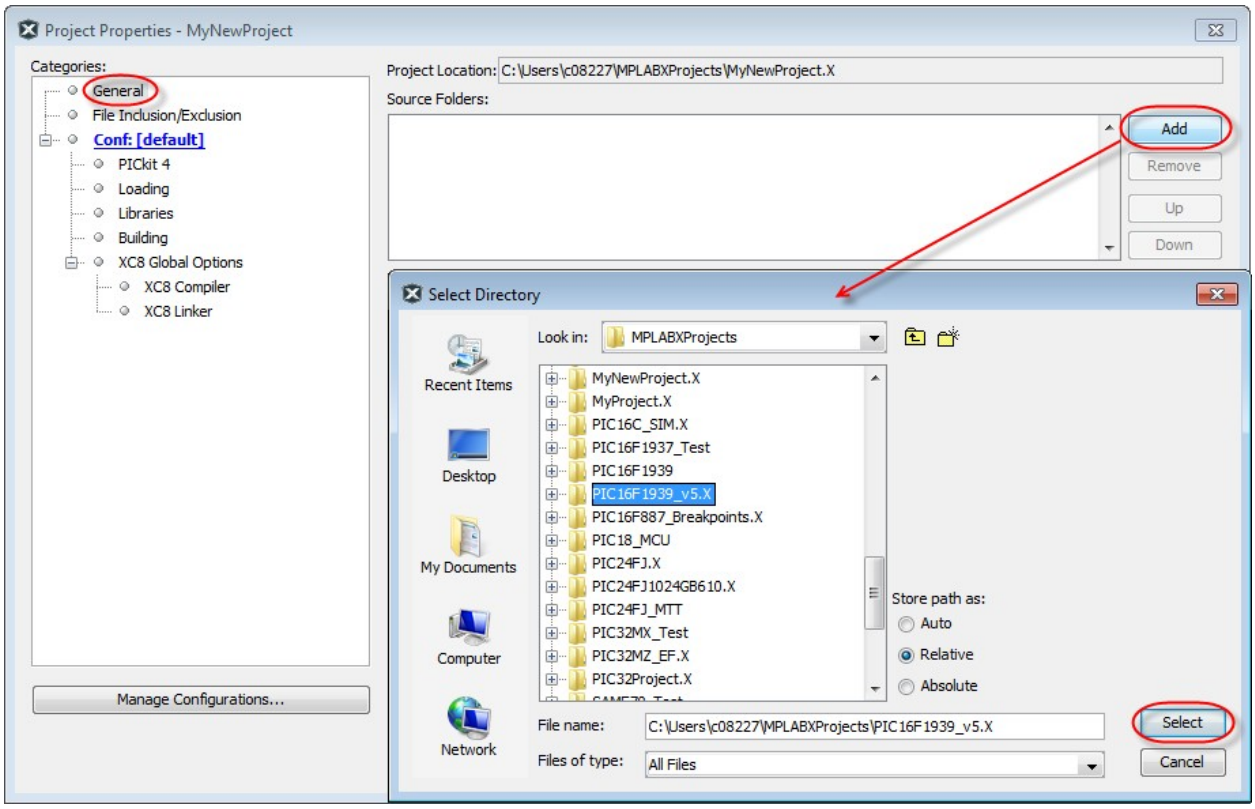
プロジェクト フォルダ以外の場所に保存されているソースファイルは、「Relative」として追加(パスを保存)します。これにより、外部ファイル用フォルダ(_ext)が作成され、プロジェクトはビルド時にそのファイルを見つける事ができるようになります。

//TODOやファイル コンテクスト メニュー等のナビゲーション機能を使えるようにするために、ファイルの保存場所をプロジェクト内で指定しておく必要があります。このために、以下を行います。

1. [Projects]ウィンドウ内のプロジェクト名の上で右クリックして、[Properties]を選択します。
2. [Categories]の下で[General]をクリックします。
3. [Source Folders]の横の[Add]をクリックします。
4. プロジェクトに追加した外部ファイルのパスを表示させ、**[Select]**ボタンをクリックします。
5. **[Apply]**または**[OK]**をクリックして、プロジェクトをリビルドします。

MPLAB IDE v8プロジェクトをインポートした場合、ソースファイルはプロジェクト フォルダ内に保存されません。自動的にファイルをインポートするには、[File] > [Import] > [MPLAB IDE v8 Project]を使います。

図4-17　外部ファイルパスの指定



4.9.4　ファイル内のコードの編集

コードの編集には、NetBeansエディタを使います。エディタの詳細はNetBeansヘルプトピックを参照してください。

https://netbeans.org/features/java/editor.html

このエディタに関するMPLAB X IDE関連の情報も参照してください。

7.「エディタ」

4.9.5　ライブラリやオブジェクト ファイルの追加と設定

リンカが使うライブラリ ファイルは以下の場所より参照できます。

- [Projects]ウィンドウの「Libraries」フォルダ
- [Project Properties]ウィンドウ

言語ツールのライブラリアンを使えば、その他のライブラリ ファイルも設定できます。

4.9.5.1　「Libraries」フォルダ

[Projects]タブで「Libraries」フォルダを右クリックすると、以下の項目が表示されます。

- [Add Library Project]
  プロジェクトに必要なライブラリを出力として生成するプロジェクトを作成します(依存性の確立)。詳細はNetBeansヘルプトピックを参照してください。

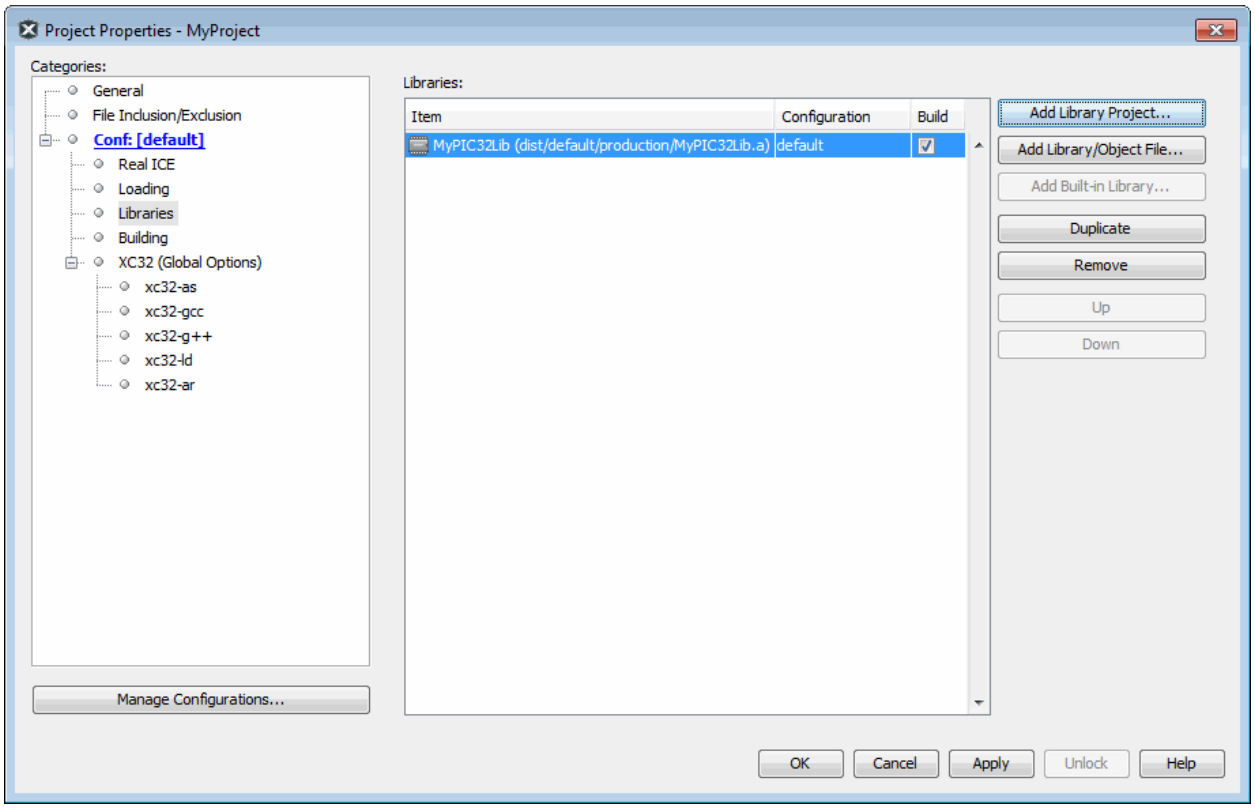
  https://netbeans.org/kb/73/java/project-setup.html#projects-dependencies.
- [Add Library/Object File]
  既存のライブラリ ファイルまたはプリビルド (オブジェクト) ファイルをプロジェクトに追加します。
- [Properties]
  その他の設定を行う[Project Properties]ウィンドウを開きます。

4.9.5.2 [Project Properties]ウィンドウ: [Libraries]カテゴリ

[Project Properties]ウィンドウを開き、[Libraries]カテゴリをクリックします。[Projects]ウィンドウの[Libraries]フォルダに追加した全てのファイルが表示されます。ウィンドウ右側のボタンを使って、さらにファイルを追加したり、表示されているファイルを管理したりする事ができます。

ライブラリリスト内のファイルの順序でリンクの順序が決まります。これは、プロジェクト内のライブラリが相互参照している場合、重要です。セカンダリ ライブラリのオブジェクトを参照するライブラリは、リンク順のリスト内でより上位に位置している必要があります。リンク順が正しくない場合、リンク処理中に「undefined reference to」エラーが発生する場合があります。リンク処理の順序の詳細は言語ツールの文書を参照してください。

図4-18 ライブラリ/オブジェクト ファイルの管理とリンク順の設定

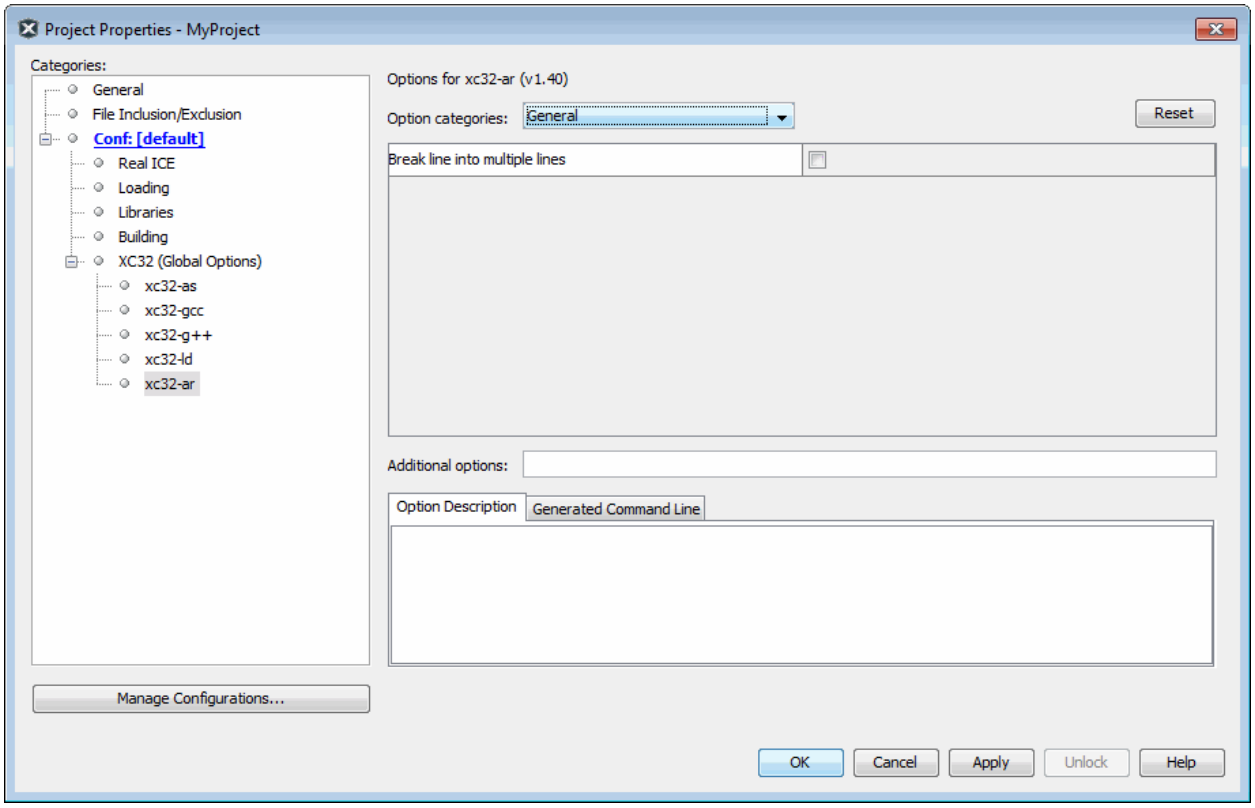


4.9.5.3 [Project Properties]ウィンドウ: ライブラリアン カテゴリ

[Project Properties]ウィンドウを開き、使う言語ツールのカテゴリの下のライブラリアン/アーカイバをクリックします。ライブラリアンの実行ファイル名は、言語ツールのマニュアルで調べます。

左側のペイン内でリンカ名をクリックします。次に右側のペイン内で、[Option categories]で[Libraries]を選択します。このリストからライブラリのオプションを選択します。

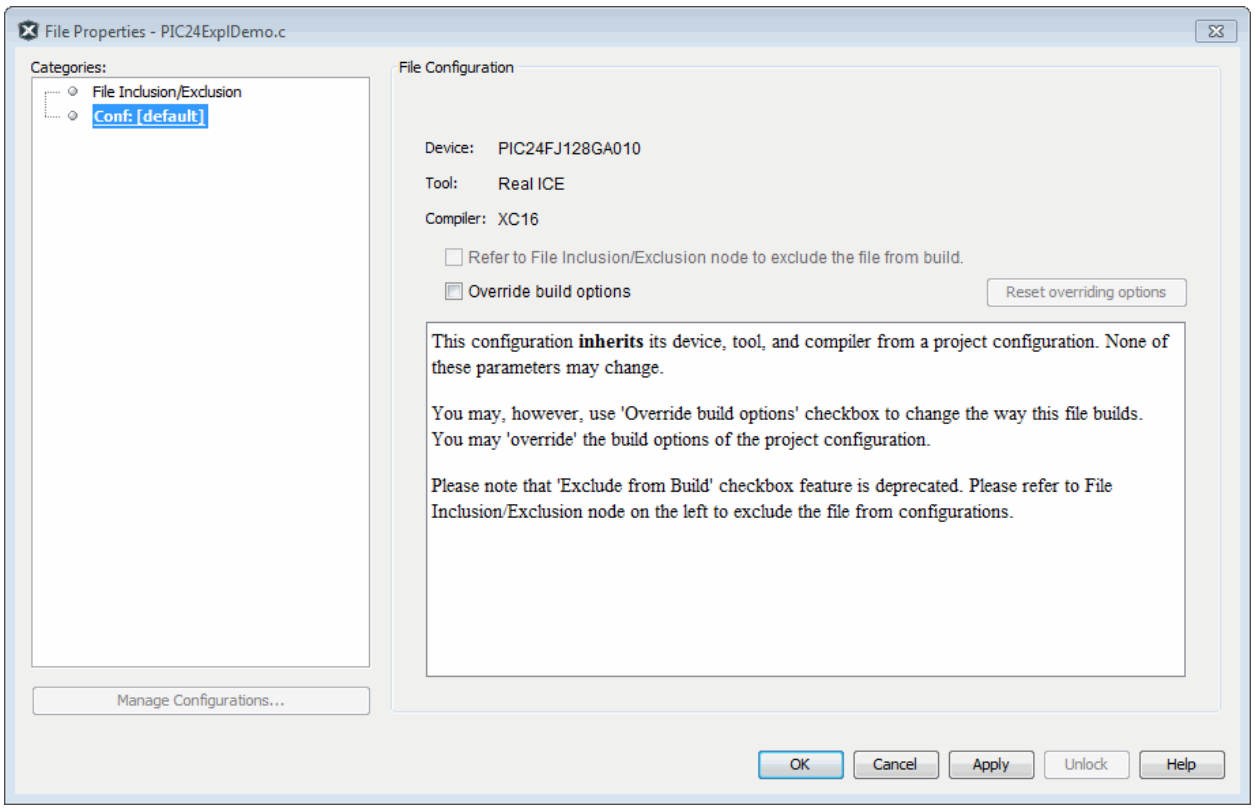
図4-19　ライブラリアンのライブラリ オプションの設定



4.9.6　ファイルおよびフォルダ プロパティの設定

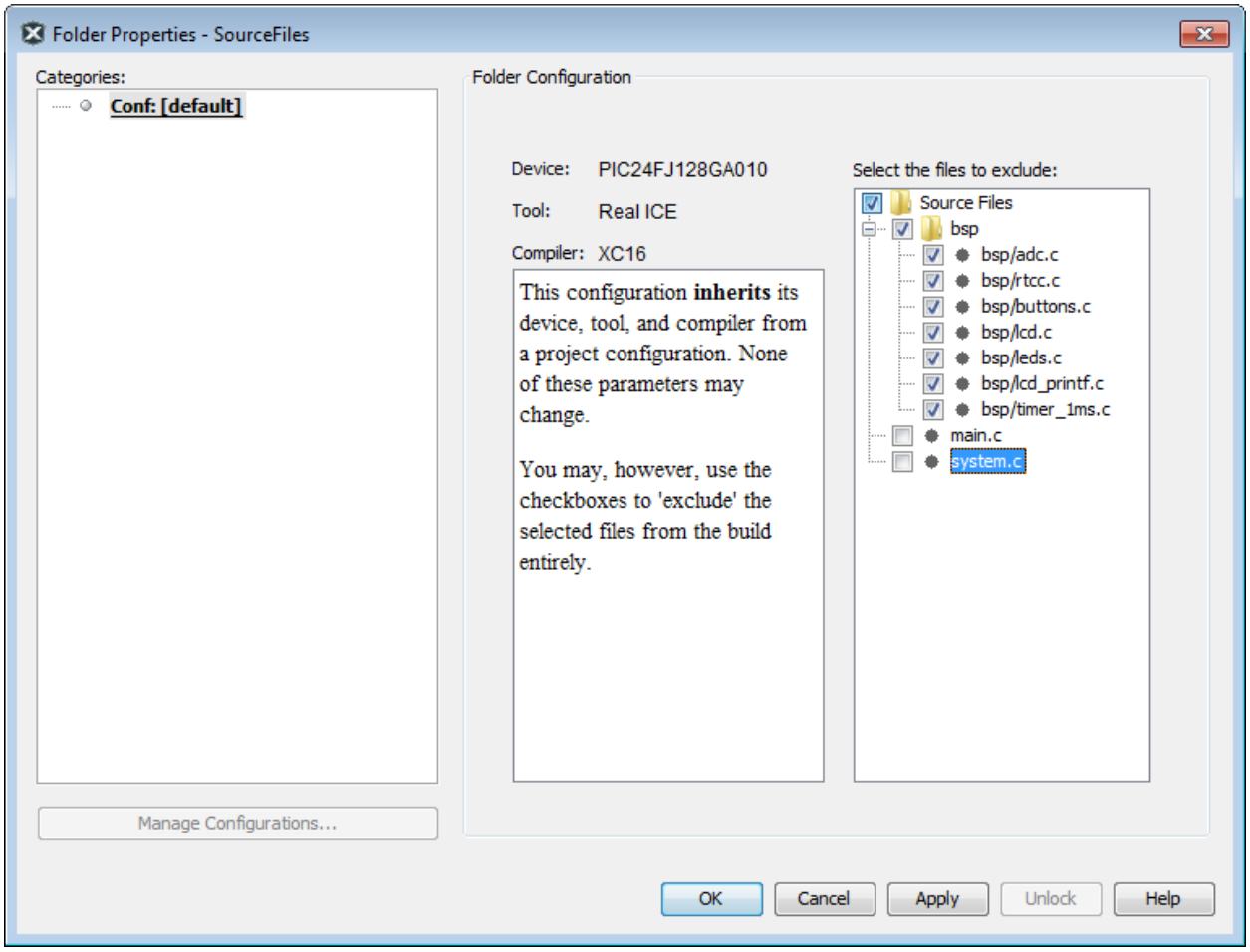
[File Properties]ダイアログを使うと、プロジェクト ファイルを、プロジェクト内の他のファイルとは異なる方法でビルドできます。このダイアログは、[File]ペインの[Projects]ウィンドウ内でファイル名を右クリックし、[Properties]を選択すると開きます。[Exclude from build]にチェックを入れると、このファイルをプロジェクトのビルドから除外できます。[Override build options]にチェックボックスを入れると、プロジェクト コンフィグレーション内のビルドオプションよりもここで選択したオプションを優先させる事ができます。[Override build options]にチェックを入れてから[Apply]をクリックすると、選択したオプションが[Categories]の下に表示されます。

図4-20　ファイル プロパティ



(仮想)フォルダ全体は、[File]ウィンドウの[Projects]タブ内でフォルダ名を右クリックし、[Properties]を選択する事でビルドプロセスから除外できます。フォルダまたは個々のファイルを選択してビルドから除外します。上位のフォルダを選択すると、そのフォルダの内容全体が選択されます。ファイルまたは他のフォルダは自由に選択を解除できます。

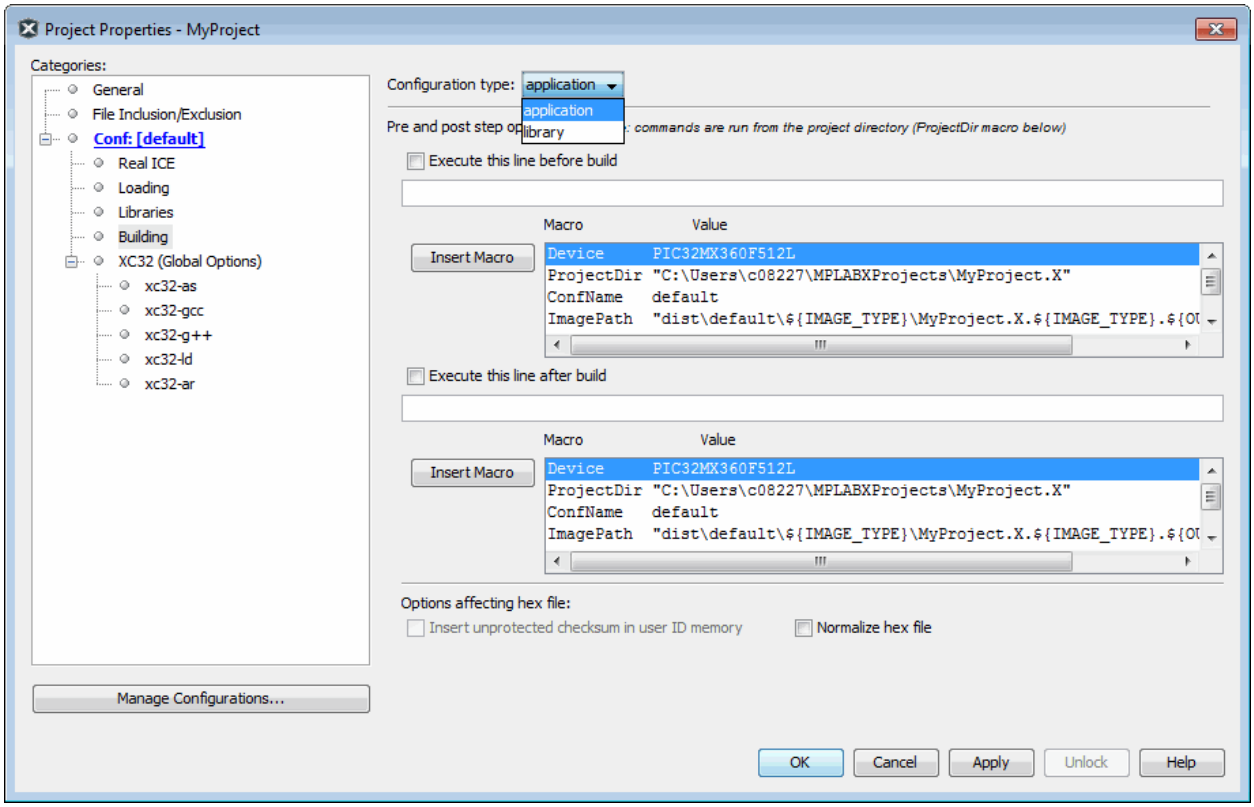
図4-21　フォルダ プロパティ



4.10　ビルドプロパティの設定

プロジェクトをビルドする前に、必要に応じて[Project Properties]ウィンドウの[Building]カテゴリでビルドプロパティを設定できます。

図4-22　ビルドプロパティ - makeオプション



4.10.1　プロジェクト コンフィグレーション タイプの変更

MPLAB X IDE v2.15では、[New Project]ウィザードでスタンドアロンまたはライブラリ プロジェクトを作成する場合、「application」と「library」のどちらかのコンフィグレーション タイプのプロジェクトが作成されます。これは、以下のセクションで示す事項に注意した上で、既存のプロジェクトのコンフィグレーション タイプを必要に応じて切り換える事ができる事を意味しています。

プリビルドおよびユーザMakefileプロジェクトは、この機能によって影響されません(これらは一度作成すると別のタイプのプロジェクトには変更できません)。

スタンドアロン プロジェクトからライブラリ プロジェクトへの切り換え

プロジェクト コンフィグレーション タイプを「application」から「library」に切り換える場合、ユーザが`main()`を削除する必要があります。コンフィグレーション タイプが「library」になると、「application」プロジェクト内の全てのロード可能ファイルはビルド/リンクされません。ロード可能ファイルの詳細は、5.5「ロード可能なプロジェクト、ファイル、シンボル」を参照してください。

ツールチェーン(XC8)がライブラリをサポートしていない場合、コンボボックスは無効になりコンフィグレーションを「library」に切り換える事はできません。

ライブラリ プロジェクトからスタンドアロン プロジェクトへの切り換え

プロジェクト コンフィグレーション タイプを「library」から「application」に切り換える場合、ユーザが`main()`を追加する必要があります。

4.10.2　ビルド前後でのスクリプト/マクロ実行

ビルドプロセスの先頭または末尾に実行するコマンドを入力します。これらのコマンドを`nbproject/Makefile-$CONF.mk`ファイルに挿入すると、ビルドプロセスをカスタマイズできます。呼び出すスクリプトまたはプログラム内でプロジェクト関連のアイテム(画像名等)を参照する必要がある場合、マクロを使います。

マクロを直接入力するか、**[Insert Macro]**をクリックして編集ボックス内の現在の位置にマクロ名をコピーします。コマンドは、カレントフォルダをMPLAB X IDEプロジェクト フォルダに設定して、makeプロセスで実行されます。プロジェクト フォルダは、`nbproject`フォルダを含むプロジェクトとして定義されます。

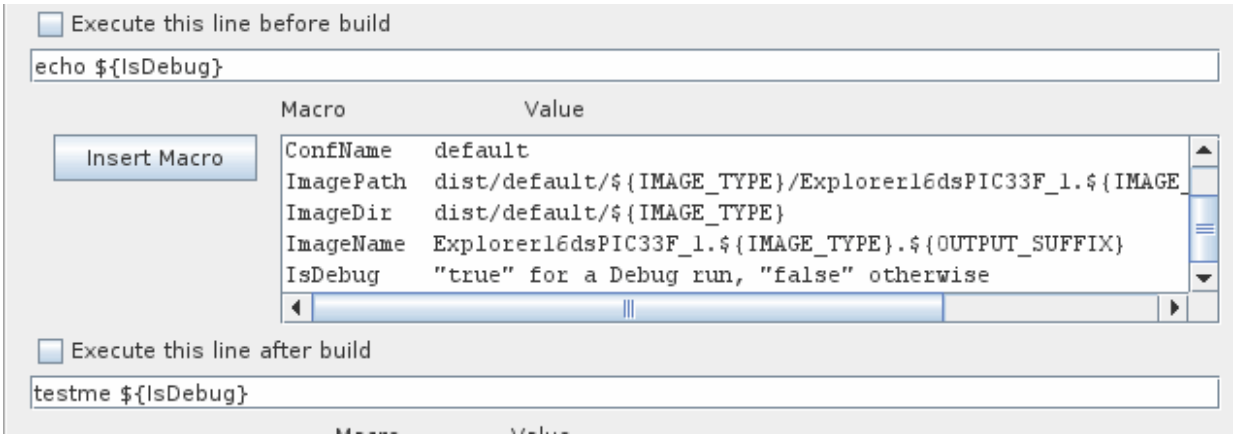
表4-6　マクロ

名前*	機能
Device	現在選択中のプロジェクト コンフィグレーションのデバイス
ProjectDir	PC上のプロジェクト ファイルの位置
ConfName	現在選択中のプロジェクト コンフィグレーションの名前
ImagePath	ビルドイメージへのパス
ImageDir	ビルドイメージを含むフォルダ
ImageName	ビルドイメージの名前
IsDebug	デバッグ実行は「true」、それ以外は「false」
* コンパイラによっては、その他のマクロも選択できる場合があります。	

例: デバッグ中にのみプロセスを実行する

以下に示すウィンドウ上で、IsDebugマクロをクリックして選択し、次に**[Insert Macro]**をクリックしてスクリプトに挿入します。

図4-23　ビルドプロパティ - プリ/ポストビルド



実行するスクリプトで、`$1` (LinuxまたはMac OS)または`%1` (Windows)の値にチェックを入れます。

Bashのサンプルコード

```
#!/bin/bash
echo is $1
if [ "$1" == "true" ]; then
echo We are in debug
else
echo We are in production
fi
```

Batch Code Example

```
@echo off
if "%1" == "true" goto debug
:production
@echo We are in production
goto end
:debug
@echo We are in debug
:end
```

4.10.3 ユーザIDメモリへの非保護チェックサムの挿入

チェックボックスをクリックすると、ビルドで生成されるチェックサムをユーザIDとして使います(デバイスがサポートしている(灰色でない)場合)。

PIC32ファミリのデバイスでは、チェックサムはユーザIDによって異なります。従って、PIC32デバイスはこの機能をサポートしていません。

4.10.4 HEXファイルの正規化

行アドレスが昇順であり、データが1サイズ(16バイト)にバッファリングされている場合、可能であればHEXファイルは正規化されます。

例えば、`myfile.hex`ファイルの行(レコード)とMPLAB XC16 Cコンパイラ/リンカの出力を以下に示します。

このファイルを正規化した場合、ファイルのデータは以下のようになります。

```
:10022800361E0000361E0000361E0000361E000076
:10023800361E0000361E0000361E0000361E000066
:10024800361E0000361E0000361E0000361E000056
```

フォーマットは以下の通りです。

```
:bbaaaarrdd...ddcc
```

これは以下を意味します。

:	開始レコード
bb	バイトカウント
aaaa	アドレス
rr	レコードタイプ
dd	データ
CC	チェックサム

MPLAB X IDEでは、Hexmateと呼ぶアプリケーションを使ってIntel HEXファイルを正規化します。[Normalize hex file]オプションにチェックを入れると、ビルド後に以下が呼び出されます。

```
hexmate myfile.hex -omyfile.hex
```

基本的に、HexmateはHEXファイル全体を解凍しHEXファイルで指定したアドレスにデータを配列します。次に、結果のメモリイメージを新規HEXファイルに圧縮し直します。結果ファイルでは全データは昇順であり、かつ連続しています。アドレスに隙間がある場合、出力ファイルにも隙間ができます(未使用アドレスは埋め込まれません)。

Hexmateの使い方の詳細はインストールしたMPLAB XC8 Cコンパイラの`docs`フォルダにある『MPLAB XC8 C Compiler User's Guide』(DS50002053)を参照してください。HexmateはMPLAB XC8の文書に記載されていますが、どのコンパイラと組み合わせても使えます。

正規化されたHEXファイルは、レコードバイトのばらつきが最小化されているため、シリアル接続でのブートローダ等のプログラミング コードとして便利です。

4.11 プロジェクトのビルド

MPLAB X IDEでは、実行またはデバッグの前にプロジェクトをビルドしておく必要はありません。ビルドは、実行とデバッグの一部として処理されます。しかし、開発の初期段階または既存プロジェクトの大幅変更時には、プロジェクトを実行またはデバッグする前に、プロジェクトのビルドを確認する場合もあります。

プロジェクトのビルド方法

- [Projects]ウィンドウのプロジェクト名を右クリックして[Build]を選択するか、[Clean and Build]を選択して中間ファイルを削除してからビルドする
- ツールバーの[Build Project]または[Clean and Build Project]アイコンをクリックする

ビルドの進捗状況は[Output]ウィンドウに表示されます。

ビルドには以下の機能があります。

表4-7　ツールバーボタンのビルドオプション

ボタンアイコン	機能	概要
	Build Project	プロジェクト内の全ファイルに対してmakeを実行します。
	Build for Debugging	プロジェクト内の全ファイルに対してmakeを実行し、ビルドされたイメージにデバッグ実行ファイルを追加します。
	Build with PRO Comparison	MPLAB XC Cコンパイラをフリーモードで使っている場合、PROモードでもビルドして出力を比較する事ができます。5.25「MPLAB XCコンパイラのFreeライセンスとPROライセンスの比較」を参照してください。
	Clean and Build	前回のビルドファイルを削除してから、プロジェクト内の全ファイルに対してmakeを実行します。
	Clean and Build for Debugging	前回のビルドファイルを削除してから、プロジェクト内の全ファイルに対してmakeを実行します。ビルドされたイメージにデバッグ実行ファイルを追加します。
	Clean and Build with PRO Comparison	前回のビルドファイルを削除してから、PRO Comparisonでビルドします。

[Output]ウィンドウへのエラーの表示方法

1. [Output]ウィンドウを右クリックして[Filter]を選択します。
2. [Filter]ダイアログの[Match Case]にチェックを入れて「: error」と入力します。ビルドを停止させたエラーだけが[Output]ウィンドウに表示されます。
3. フィルタのON/OFFは<Ctrl>+<G>でトグルします。

エラーに関する詳細は言語ツールのマニュアルを参照してください。

チェックサム情報の表示方法

[Dashboard]ウィンドウ(5.19「ダッシュボードの表示」参照)を開き、ビルド後のチェックサムを確認します。

4.12 コードの実行

コードのビルドに成功すれば、アプリケーションを実行できます。

コード実行時の動作

[Run Project]ツールバーアイコン  をクリックすると以下の動作が実行されます。

- make処理がビルドを必要と判断した場合、ビルドを実行します。
- インサーキット デバッガ/エミュレータまたはプログラマを使う場合、ターゲット デバイスには自動的にイメージ (デバッグ実行ファイルは含まず)がプログラミングされ、次いでデバイスが実行に向けてリリースされます。この場合、ブレークポイント等のデバッグ機能は有効になりません。

Note: プログラミング後にデバイスをリセット状態に保持するには、「Run Project」ではなく[Hold in Reset]  アイコンをクリックします。

- シミュレータの場合、アプリケーションは一切のデバッグ機能を使わずコードを実行します。

コード実行の進捗状況が[Output]ウィンドウに表示されます。

アプリケーション コードの実行

1. [Projects]ウィンドウでプロジェクトを選択するか、そのプロジェクトをメイン プロジェクトに指定します(右クリックして[Set as Main Project]を選択)。
2. 以下のどちらかの方法でプログラムを実行します。
   - [Run Program]アイコン  をクリックする
   - [Make and Program Device]アイコン  をクリックする

コード実行の進捗状況が[Output]ウィンドウに表示されます。

4.12.1 実行に関する注意事項

プログラムを実行する場合、MPLAB X IDEは実行時(RunまたはDebug)のみハードウェア ツールに接続する事に注意してください。従って、MPLAB X IDEの設定は、実行時にのみツールに反映します。停止中に変更された設定は、実行が再開した時にのみ更新されます。

MPLAB IDE v8のように常時ハードウェア ツールに接続するには、[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択し、**[Embedded]**ボタンをクリックし、**[Generic Settings]**タブで[Maintain active connection to hardware tool]にチェックを入れます。4.20「デバイスのプログラミング」も参照してください。

4.13 コードのデバッグ

コードが正常に動作しない場合、デバッグが必要です。

デバッグ時の動作

[Debug Project]ツールバー アイコン  をクリックすると以下の動作が実行されます。

- make処理がビルドを必要と判断した場合、ビルドを実行します。
- インサーキット デバッガ/エミュレータを使う場合、ターゲット デバイスまたはヘッダは自動的にイメージ (デバッグ実行ファイルを含む)がプログラミングされ、デバッグ セッションが始まります。
- シミュレータの場合、即座にデバッグ セッションが始まります。

デバッグの進捗状況は[Output]ウィンドウに表示されます。**アプリケーション コードのデバッグ**

アプリケーション コードをデバッグする方法

1. [Projects]ウィンドウでプロジェクトを選択するか、そのプロジェクトをメイン プロジェクトに指定します(右クリックして[Set as Main Project]を選択)。
2. [Debug Project]アイコン  または[Step Into]アイコンをクリックして、デバッグを始めます。

アプリケーション コードの実行を一時停止する方法

[Pause]アイコン  をクリックすると、プログラムの実行が一時停止します。

コードの実行を再開する方法

[Continue]アイコン  をクリックすると、プログラムの実行が再開します。

コードの実行を終了する方法

[Finish Debugger Session]アイコン  をクリックすると、プログラムの実行が終了します。

デバッガを起動する方法

コードがデバッグ用にビルドされたもので、単にデバッグツールを起動するだけならば、[Debug Project]アイコン横の下矢印をクリックして、[Launch Debugger]を選択します。

表4-8　[Debug]アイコン

アイコン	機能	関連メニュー項目
	Debug Project	Debug>Debug Project
	Step Into	Debug>Step Into
	Pause	Debug>Pause
	Continue	Debug>Continue
	Finish Debug Session	Debug>Finish Debugger Session

4.13.1　デバッグ用マクロの生成

MPLAB X IDEは、Microchip社製言語ツール向けのデバッグ用マクロを生成します。Microchip社製コンパイラおよびアセンブラに渡されるマクロを下表に示します。

表4-9　Microchip社製ツールのデバッグ用マクロ

マクロ名	関連ツール	機能
__DEBUG	全て	デバッグ用のビルドである事を示します。
__MPLAB_REAL_ICE_ __MPLAB_ICD3_ __MPLAB_PK3_ __MPLAB_PICKIT2_	XC8	使用中のハードウェア デバッグツールを示します。 フォーマットは_MPLAB_xxx_ (xxxはハードウェア ツールの指定子)です。
__MPLAB_DEBUGGER_REAL_ICE __MPLAB_DEBUGGER_ICD3 __MPLAB_DEBUGGER_PK3 __MPLAB_DEBUGGER_PICKIT2	XC16, XC32, MPASM	使用中のハードウェア デバッグツールを示します。 フォーマットは_MPLAB_DEBUGGER_xxx (xxxはハードウェア ツールの指定子)です。
__MPLAB_DEBUGGER_PIC32MXSK	XC32	使用中のスタータキットを示します。

これらのマクロはユーザコード内で使えます。例:

```
#ifdef_DEBUG
fprintf(stderr,"This is a debugging message\n");
#endif
```

4.13.2 デバッグに関する注意事項

コードをデバッグする場合、以下の事項に注意します。

- デバッグ機能(ウォッチ ウィンドウやメモリ ウィンドウでの変数値の表示等)の多くは、デバッグ セッション(デバッグモード)内でのみ使えます。
- MPLAB X IDEは常時ハードウェア ツールに接続しています。実行時のみ接続するには、[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択し、**[Embedded]**ボタンをクリックし、**[Generic Settings]** タブで[Maintain active connection to hardware tool]のチェックを外します。
  チェックを外すと、MPLAB X IDEで行った設定は実行時にのみツールに反映されます。停止中に変更された設定は、実行が再開した時にのみ更新されます。
- アプリケーションによっては、個別に実行するためにデバッグステップの分割が必要になる場合があります。それには、[Debug] > [Discrete Debugger Operation]による手順を使います。
- デバッグ中はコンパイラの最適化を無効にするか、コンパイラで利用可能な場合は「debug」最適化を使います。[Project Properties]ウィンドウの[Categories]で実行可能な言語ツールを選択します。次に、[Options categories]から[Optimizations]を選択し、最低レベルを選択します。「0」があれば、これを選択します。

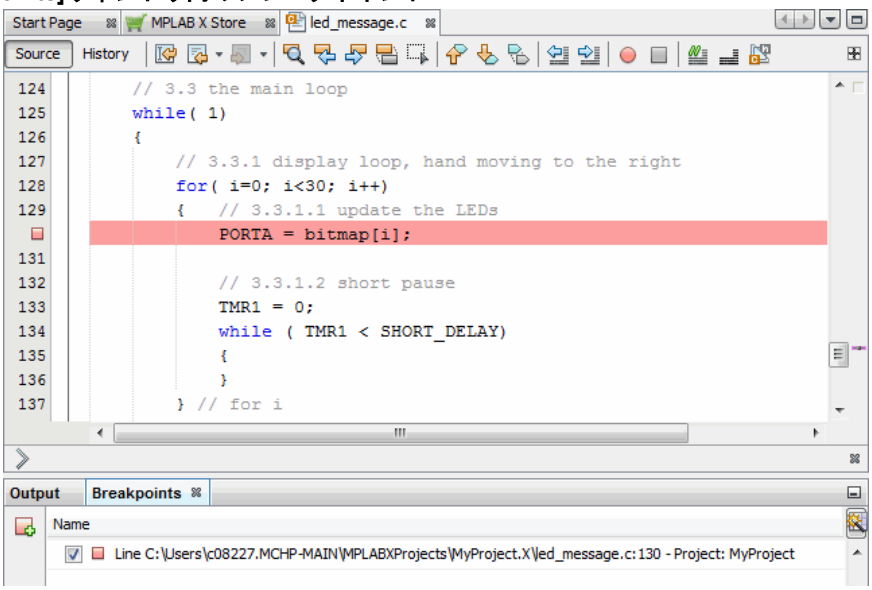
4.14 ブレークポイントによるプログラム実行の制御

コードをデバッグする際に、コードの特定位置で実行を一時停止して、変数の値を調べる事ができると便利です。そのためにブレークポイントを使います。

ブレークポイントの詳細はNetBeansヘルプトピックを参照してください。

https://netbeans.org/kb/docs/cnd/debugging.html#breakpoints

図4-24 [Breakpoints]ウィンドウ内のブレークポイント



関連リンク

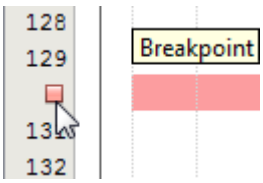
12.2 「[Breakpoints]ウィンドウ」
12.3 「[New Breakpoint]ダイアログ」

4.14.1 単純な行ブレークポイントのセット/クリア

以下のいずれかの方法により、コード行にブレークポイントをセット/クリアできます。

- ソースエディタで左側の余白をクリックする(下図参照)
- 行をクリックしてから<Ctrl>+<F8>を押す
- [Debug] > [Toggle Line Breakpoint]を選択する

図4-25　クリックしてブレークポイントを有効/無効にする



4.14.2　[New Breakpoint]ダイアログによるブレークポイントの設定

[New Breakpoint]ダイアログを開くには以下の手順を実行します。

1. [Breakpoints]ウィンドウの左上にある[Create New Breakpoint]アイコン  をクリックします。
2. [Debug] > [New Breakpoint]を選択します。ブレークポイントのタイプを選択し、オプションを設定します。

表4-10　ブレークポイントのタイプ

選択肢	説明
Line	[Editor]ウィンドウのコード行でブレーク
Data	データ(RAM)メモリ内のアドレスの読み書きでブレーク(SFRおよびシンボル位置も含む)
Address	プログラム (フラッシュ) メモリ内のアドレスを実行時にブレーク
Event	リセット、スリープ/ウェイクアップ、ウォッチドッグ タイマ タイムアウト等、特定のイベントでブレーク
Function	特定の関数へのエントリ時にブレーク

各ブレークポイント タイプに利用可能なオプションは12.3「[New Breakpoint]ダイアログ」を参照してください。

4.14.3　[Breakpoints]ウィンドウによるブレークポイントのセット/クリア

[Breakpoints]ウィンドウを使ってブレークポイントを表示し、トグルするには以下の手順を実行します。

1. チェックボックスを使ってブレークポイントをトグルします。
2. [Window] > [Debugging] > [Breakpoints]を選択します。

4.14.4　ブレークポイント シーケンスの設定(一部デバイスのみ)

ブレークポイント シーケンスは複数のブレークポイントのリストです。プログラムの実行は各ブレークポイントで停止せず、最後のブレークポイントの実行後に停止します。ブレークポイント シーケンスは、特定の命令に至るまでに複数の実行パスが存在する場合、その中の1つのパスの動作だけを調べる時に便利です。

ブレークポイント シーケンスの作成方法

1. 既存のブレークポイントを右クリックするか、複数の既存ブレークポイントを<Shift>+クリックで選択して右クリックします。
2. コンテクスト メニューから[Complex Breakpoint]へ進み、[Add a New Sequence]を選択します。
3. ダイアログ ボックスにシーケンス名を入力し、[OK]をクリッします。
4. 新しく作成したシーケンスの下に、最初に選択したブレークポイントが表示されます。
5. 別の既存ブレークポイントをシーケンスに追加するには、そのブレークポイントを右クリックし、[Complex Breakpoint] > [Add to Name]を選択します(Nameはシーケンス名)。
6. シーケンスに既存ではない新しいブレークポイントを追加するには、シーケンスを右クリックし、コンテクスト メニューから[New Breakpoint]を選択します。

シーケンスの並び順を選択する方法

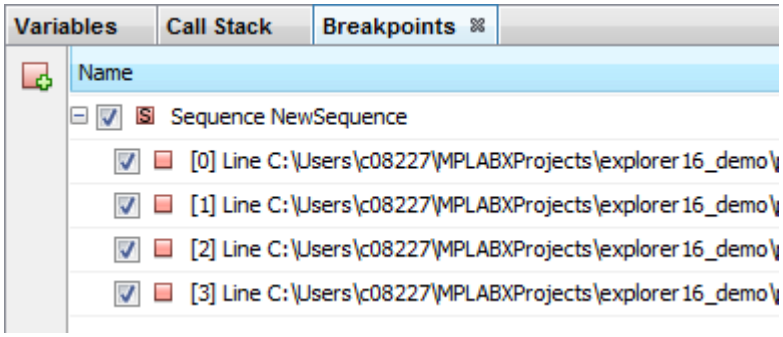
1. 全ての項目が表示されるように、シーケンスを展開します。
2. 項目を右クリックして、[Complex Breakpoints] > [Move Up]または[Complex Breakpoints] > [Move Down]を選択します。ブレークポイントのシーケンス実行順はボトムアップです。つまり、シーケンスの最後のブレークポイントが最初に発生します。

シーケンスまたはブレークポイントの無効化/削除方法

- 項目を右クリックして「Disable」を選択すると、その項目は一時的に無効になります。

- 項目を右クリックして「Delete」を選択すると、その項目は完全に削除されます。

図4-26　ブレークポイント シーケンスの例



4.14.5　ブレークポイント タプルの設定(一部デバイスのみ)

MPLAB X IDEでは、「タプル」はAND演算されるブレークポイントのリストを意味します。ブレークポイントのAND演算は、複数箇所で変更される変数が特定の箇所で変更された時にだけブレークさせたい場合に便利です。

AND演算は2点のみ可能です。1つはプログラムメモリ ブレークポイント、もう1つはデータメモリ ブレークポイントである事が必要です。両方のブレークポイントが同時に発生した時にのみ、プログラムが一時停止します。

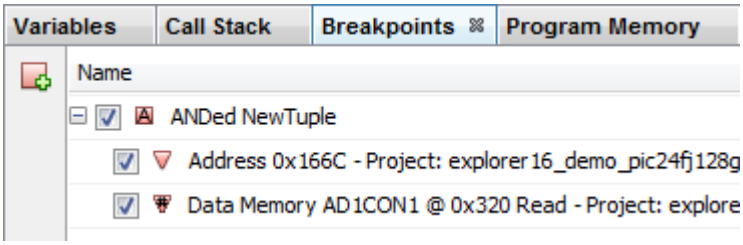
ブレークポイント タプルの作成方法

1. [Breakpoints]ウィンドウの左上の[Create New Breakpoint]アイコン  をクリックします。[New Breakpoint]ダイアログが開きます。
2. アドレス ブレークポイントを作成します。**[OK]**をクリックして、作成したブレークポイントを[Breakpoints]ウィンドウに追加します。
3. 上の1と2を繰り返して、データ ブレークポイントを作成します。
4. いずれかのブレークポイントを右クリックし、[Complex Breakpoint] > [Add to New Tuple]を選択します。
5. ダイアログ ボックスにタプル名を入力し、**[OK]**をクリックします。
6. 新しく作成したタプルの下に、そのブレークポイントが表示されます。
7. もう1つのブレークポイントを右クリックし、[Complex Breakpoint] > [Move to Name]を選択します(Nameはタプル名)。

タプルまたはブレークポイントを削除するには、以下のどちらかの操作を行います。

- 項目を右クリックし、[Disable]を選択すると、その項目は一時的に無効になります。
- 項目を右クリックし、[Delete]を選択すると、その項目は完全に削除されます。

図4-27　ブレークポイント タプルの例



4.14.6　ブレークポイント アドレスの表示

デバッグ セッションの前と実行中にブレークポイントのアドレスを表示するには、以下の手順を実行します。

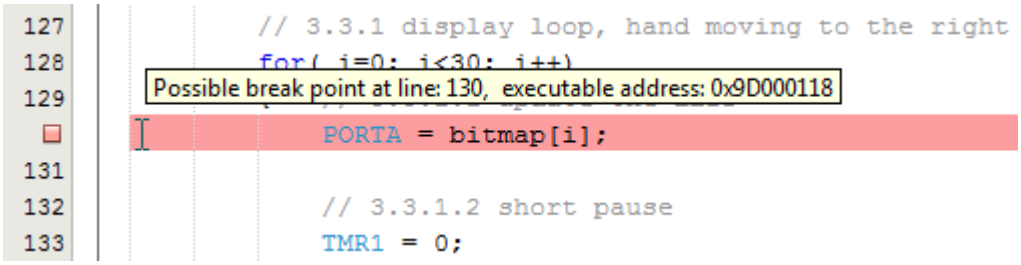
1. [Tools] > [Options]を選択し、**[Embedded]**ボタンをクリックし、**[Generic Settings]**タブで[On mouseover source lines in editor, evaluate break point status]を選択します。
2. 行ブレークポイントを設定します。
3. [Debug Project]アイコンをクリックしてコードをデバッグします。

4. ブレークポイント アイコンとソースコード先頭の間にカーソルを置き、ステップ1の操作を実行してマウスオーバーで確認します。

単一のブレークポイント アドレス

多くの場合、単一のアドレスが表示されます。プログラム/実行メモリウィンドウ([Window] > [Target Memory Views])を開くと、その位置でのメモリアドレスと命令が表示されます。

図4-28　ブレークポイント アドレスの表示

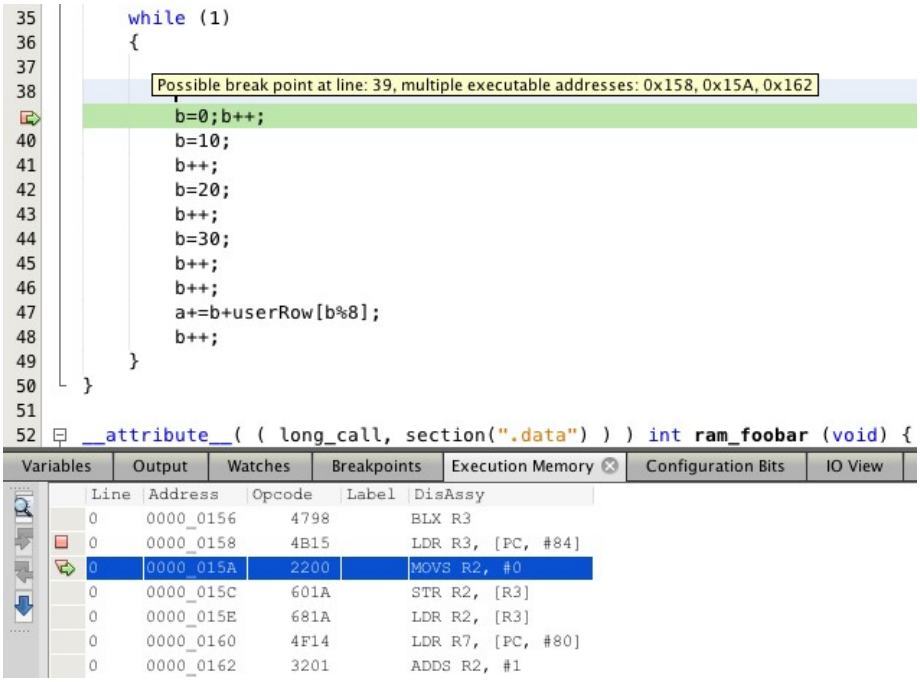


複数のブレークポイント アドレス

場合によっては、複数のブレークポイント位置が表示される事があります。理由としては、コンパイラの最適化レベル指定(-Os)が高く設定されている場合が考えられます。MPLAB X IDE v5.20以前は最初のアドレスのみ表示されていました。現在は全てのアドレスが表示されます。

プログラム/実行メモリウィンドウ([Window] > [Target Memory Views])を開くと、その位置での各メモリアドレスと命令が表示されます。設定したブレークポイントがBrokenブレークポイントあるいは設定できない場合、メモリウィンドウ内で次または他の位置を選択できます。その場合でも対応関係にあればコード内では同じ行番号に表示されます。

図4-29　複数ブレークポイント アドレスの表示



4.14.7　ハードウェア ブレークポイントの使い方

コードの行で一時停止する場合、以下のどちらかを実行します。

- ソフトウェア ブレークポイントからハードウェア ブレークポイントへ切り換える
- ハードウェア ブレークポイントを設定する

デバッグを実行すると、同じコード行で停止する事に注意が必要です。[Debug]を再実行してプログラム実行を継続するか、シングルステップ実行でブレークポイント行のコードを実行します。

4.14.8 デュアル パーティション デバイスとブレークポイント

デバッグの性質上、デュアル パーティション デバイスの場合、デバッガが停止したのがアクティブ パーティションであれば、そのアクティブ パーティション内にブレークポイントを設定する必要があります。従って、パーティション1で開始したら、ブレークポイントをパーティション2に設定する事はできません。切り換え点まで実行し、パーティション2に切り換えてから、パーティション2でのブレークポイントを設定する必要があります。

4.14.9 ブレークポイント用の機能

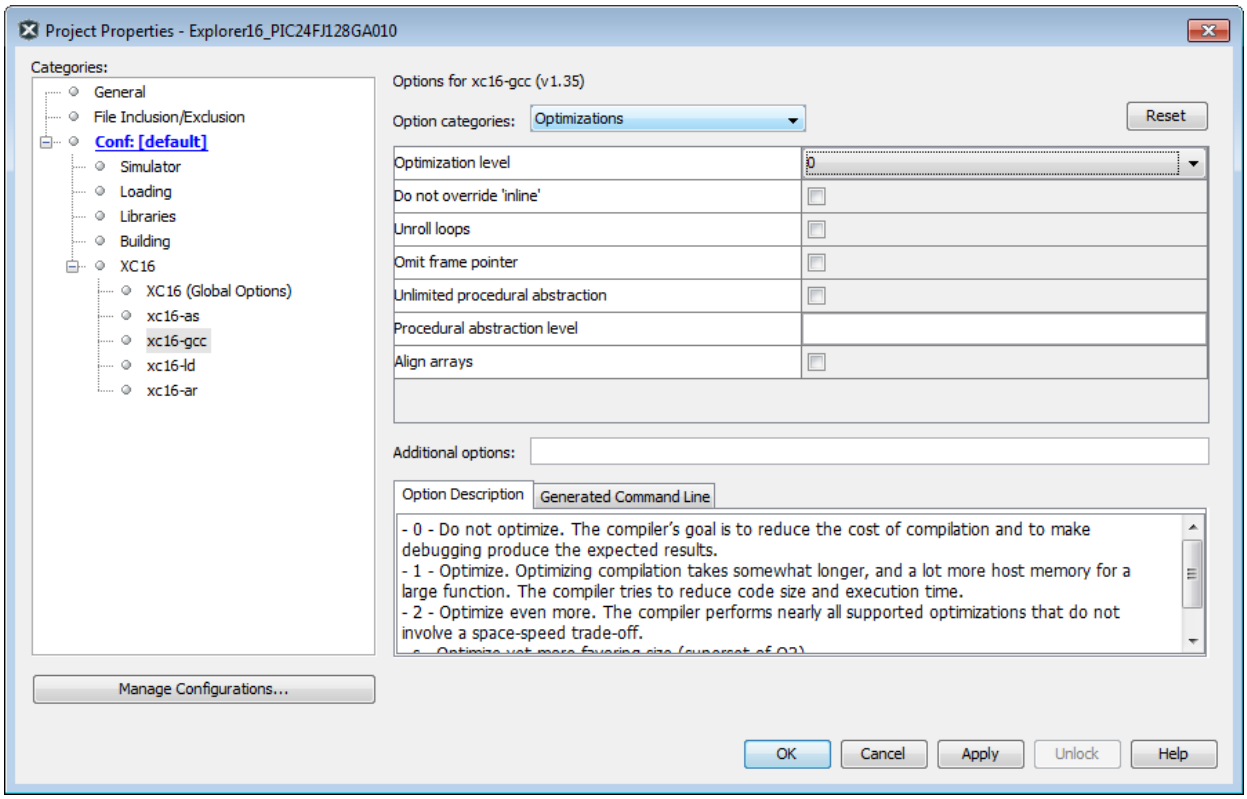
ブレークポイントから次のブレークポイントまでの実行時間を計測するにはストップウォッチを使います(5.15「ストップウォッチの使用」参照)。

ブレークポイント リソースを確認するには[Dashboard]ウィンドウ(5.19「ダッシュボードの表示」参照)を開きます。ブレークポイントの利用可能数と使用数、ソフトウェア ブレークポイントがサポートされているかを確認できます。

4.14.10 コンパイラの最適化とブレークポイント

コンパイラの最適化によって、プログラムの実行方法が変わる場合があります。そのような場合、ブレークポイントのメモリアドレス マッピングに影響し、複数アドレスが生成される可能性があります。ブレークポイント関連の問題を避けるため、デバッグ時は最低レベルの最適化を使う事を推奨します。

図4-30 最低レベルの最適化例



4.14.11 ブレークポイントのリソース使用量

ハードウェア デバッガ(例: インサーキット デバッグツール)では、使えるブレークポイント数に制限があります。利用可能なブレークポイントの数はデバイスによって異なります。利用可能なブレークポイントの数と、既に使用中のブレークポイントの数は、[Dashboard]ウィンドウ(5.19「ダッシュボードの表示」)で調確認できます。

以下のMPLAB X IDE機能は、ブレークポイントを必要とします。

	Step Over
	Step Out

	Run to Cursor
	Reset to Main

利用可能なブレークポイントが残っていない時にこれらの機能を使おうとすると、ダイアログが開いて全てのリソースが使用中である事を知らせます。

4.14.12 ブレークポイント情報

デバッグ セッション中と終了後に、ツールヒントとアイコンによってブレークポイント情報が表示されます。

- ビルドなしにデバッグ セッションから出た場合、ブレークポイントを設定しようとすると、マウス ツールチップが青色アイコンで表示されデバッグ情報がない事を示します。
- ビルドありでデバッグ セッションから出た場合、行の最適化によりブレークポイント設定ができない場合もあります。
- デバッグ セッション中の場合 - ブレークポイント設定が可能か不可能かがツールチップに表示されます。

ツールチップは非表示にする事もできます([TOOLS] > [OPTIONS]、**[Embedded]**、[Generic Settings])。

4.14.13 Brokenブレークポイント アイコン

ブレークポイント アイコンが破れた状態()で表示される場合があります。

- ビルドなしにデバッグセッションから出た場合、ブレークポイントを設定しようとすると、マウス ツールチップに破れたアイコンが表示されデバッグ情報はない事を示します。
- デバッグ セッション中にパス、ファイル、フォルダの命名制約が原因でソースファイルが見つからない事があります(「パス/ファイル/フォルダ名の制約」参照)。

4.15 コードのステップ実行

[Debug]メニューと[Debug]ツールバーのステップ実行機能を使うと、コードの先頭またはブレークポイントでの停止状態からコードをステップ実行できます。これにより、変数値の変化を観察したり、プログラムフローが正しいかどうかを判断できます。

コードのステップ実行は、以下の方法で行えます。

	**Step Over** – プログラムのソースを1行実行します。その行が関数コールである場合、呼び出した関数全体を実行した後に停止します。
	**Step Into** – プログラムのソースを1行実行します。その行が関数コールである場合、呼び出した関数の先頭の命令文を実行した後に停止します。
	**Step Out** – プログラムのソースを1行実行します。その行が関数コールの場合、実行後に呼び出し元に戻ります。
	**Run to Cursor** – プロジェクトをカーソル位置まで実行して、停止します。

[Editor]ウィンドウ以外に、[Disassembly]ウィンドウ(5.16「 [Disassembly]ウィンドウの表示」)と、[Memory]ウィンドウ内のプログラムメモリでも、コードをシングルステップ実行できます。

4.16 シンボル値の変化の観察

選択したシンボルの値の変化を[Watches]ウィンドウで観察します。これらの値がプログラム実行中に予測通りに変化するか確認する事により、コードのデバッグに役立てる事ができます。

以下のシンボルを[Watches]ウィンドウに追加できます。

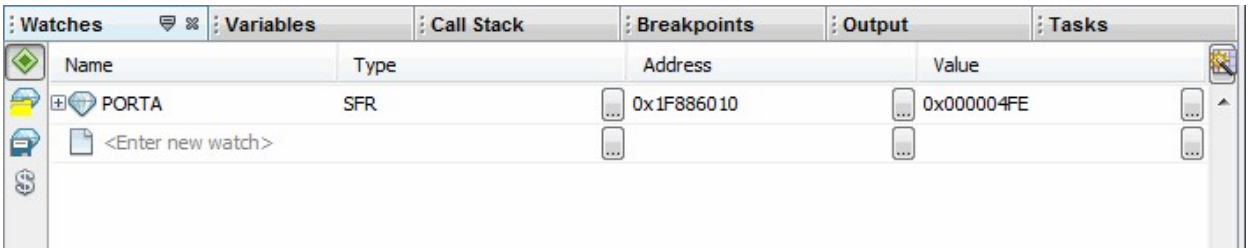
- グローバル シンボル – ビルド後に表示されます。
- SFR – 特殊機能レジスタ (デバイスによって異なります)
- 絶対アドレス

一般的に、値の変化を観察するにはデバッグを一時停止する必要があります。しかし、ツールによっては実行時の更新が可能です(プログラム実行中に値の変化を観察できます)。この機能が使えるかどうかは、ツールのマニュアルを参照してください。

PIC32 MCU以外のデバイスでは、ランタイム ウォッチに使うシンボルはデバイスメモリに適合したサイズである必要があります。つまり、8ビットデバイスを使用する場合、8ビットのシンボルを使う必要があります。

C言語エニュメレート タイプでは、enumラベル (テキスト)と整数値のどちらかをウィンドウで入力できます。ラベルは、大文字と小文字を区別する事に注意します。

図4-31　[Watches]ウィンドウ – プログラムの一時停止



[Watches]ウィンドウは以下の方法で開きます。

- [Window] > [Debugging] > [Watches]を選択して、[Watches]ウィンドウを開きます。
- ウィンドウが既に開いている場合、[Output]ウィンドウ内の[Watches]タブをクリックします。

新規ウォッチを直接作成する方法

以下のいずれかの方法により、シンボルを[Watches]ウィンドウに直接追加できます。

- [Name]列をダブルクリックして、グローバル シンボル、SFR、絶対アドレス(0x300)を入力します。
- [Editor]ウィンドウのグローバル シンボルまたはSFRを右クリックして、[New Watch]を選択します。
- [Editor]ウィンドウ内でグローバル シンボルまたはSFRを選択し、これを「Watches」ウィンドウにドラッグ&ドロップします。

[New Watches]ダイアログを使って新規ウォッチを作成する方法

以下のどちらかの方法により、シンボルまたはSFRを[Watches]ウィンドウに追加できます。

- [Watches]ウィンドウ内で右クリックして[New Watch]を選択するか、[Debug] > [New Watch]を選択します。ラジオボタンで、[Global Symbols]または[SFRs]のどちらをリストに表示するのかを選択します。リスト内でいずれかの名前をクリックし、**[OK]**をクリックします。
- [Editor]ウィンドウ内でシンボルまたはSFR名を選択し、右クリックで開いたメニューから[New Watch]を選択すると、その名前がウィンドウに表示されます。**[OK]**をクリックします。

ランタイム ウォッチの新規作成方法

ランタイム ウォッチを[Watches]ウィンドウに追加する前に、クロックを以下のように設定する必要があります。

1. プロジェクト名の上で右クリックして[Properties]を選択します。
2. デバッグツールの名前(REAL ICE等)をクリックし、オプション カテゴリ[Clock]を選択します。
3. 命令実行速度を設定します。

グローバル シンボルまたはSFRをランタイム ウォッチとして追加する方法は、上の「[New Watches]ダイアログを使って新規ウォッチを作成する方法」と基本的に同じですが、「New Watch」ではなく「New Runtime Watch」を選択する必要があります。

PIC32 MCU以外のデバイスでは、ランタイム ウォッチに使うシンボルはデバイスメモリに適合したサイズである必要があります。つまり、8ビットデバイスを使う場合、8ビットのシンボルを使う必要があります。

シンボル値の変化を表示する方法

1. プログラムをデバッグ実行し、一時停止します。
2. [Watches]タブをクリックしてアクティブにします。
3. シンボルをウォッチする場合、コードをデバッグ実行し、一時停止して値の変化を観察します。
   シンボルをランタイム ウォッチする場合、コードをデバッグ実行すると、プログラムを実行しながら値の変化を観察できます。

シンボル (グローバル シンボル、SFR、配列、レジスタのビットフィールド等)の値を表示するには、デバッグ セッションを実行している必要があります。

Note: この方法でマクロの値を表示する事はできません。デバッグ中はエディタ内でマクロを右クリックし、[Macro Expansion]ウィンドウを使います。

ウォッチシンボルの基数を変更する方法

シンボルの行で右クリックし、[Display Value As]を選択します。

その他の作業の方法

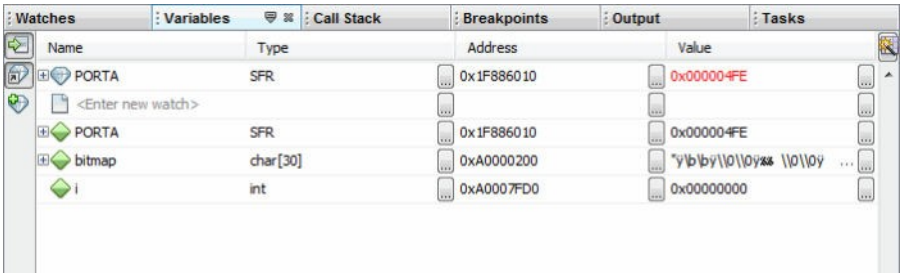
詳細は12.19「[Watches]ウィンドウ」を参照してください。

4.17 ローカル変数値の変化の観察

ローカル変数の値の変化を[Variables]ウィンドウで観察します。これらの値がプログラム実行中に予測通りに変化するか確認する事により、コードのデバッグに役立てる事ができます。

一般的に、値の変化を観察するにはデバッグを一時停止する必要があります。しかしツールによっては、実行中に値の変化を観察できます。この機能が使えるかどうかは、ツールのマニュアルを参照してください。

図4-32 [Variables]ウィンドウ – プログラムの一時停止



[Variables]ウィンドウを開く方法

- [Window] > [Debugging] > [Variables]を選択して[Variables]ウィンドウを開きます。
- [Variables]ウィンドウが既に開いている場合、[Output]ウィンドウ内の[Variables]タブをクリックします。

変数の変化を観察する方法

1. プログラムをデバッグ実行し、一時停止します。
2. [Variables]タブをクリックしてウィンドウを開くと、ローカル変数値が表示されます。

変数の基数を変更する方法

変数の行で右クリックし、[Display Value As]を選択します。

4.18 デバイスメモリの表示と変更

MPLAB X IDEは柔軟で簡潔な[Memory]ウィンドウを備えています。このウィンドウには、デバッグ中に各種タイプのデバイスメモリの内容を表示できます。このウィンドウ内の値を更新するには、デバッグを一時停止する必要があります。

デバイスメモリの表示

1. ペイン内のウィンドウをクリックしてペインをアクティブにします。[Memory]ウィンドウはこのペイン内に開きます。
2. [Window] > [Target Memory Views]を選択します。選択肢を下表に示します。一部のデバイスでは使えないメモリタイプもあります。

表4-11 メモリ画面 – 8/16ビットデバイス

タイプ	説明
Program Memory	デバイス上の全てのプログラムメモリ(ROM)
File Registers	デバイス上の全てのファイルレジスタ(RAM)

(続き)

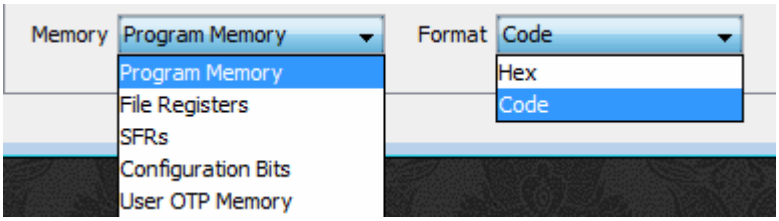
タイプ	説明
SFR	全ての特殊機能レジスタ(SFR)
Peripherals	周辺モジュールごとの全てのSFR
Configuration Bits	全てのコンフィグレーション レジスタ
EE Data Memory	デバイス上の全てのEEデータメモリ
User OTP Memory	ユーザOTPメモリ
User ID Memory	ユーザIDメモリ

表4-12　メモリ画面 – 32ビットデバイス

タイプ	説明
Execution Memory	デバイス上の全てのフラッシュメモリ
Data Memory	デバイス上の全てのRAM
Peripherals	全ての特殊機能レジスタ(SFR)
Configuration Bits	全てのコンフィグレーション レジスタ
CPU Memory	全てのCPUメモリ
User ID Memory	ユーザIDメモリ

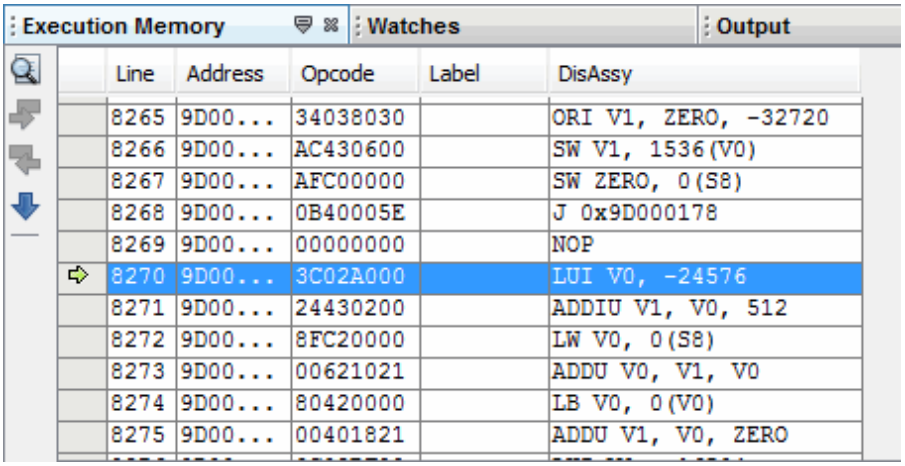
3. [Memory]ウィンドウでは、[Memory]ドロップダウン ボックスからメモリを選択して画面を変更できます。メモリ画面のフォーマットは[Format]ドロップダウン ボックスからも変更できます。フォーマットの選択肢はメモリタイプとデバイスによって異なります。

図4-33　メモリとメモリ フォーマットの選択



4. デバッグを一時停止すると、選択したメモリの内容が[Memory]ウィンドウに表示されます。
5. ウィンドウを閉じるには、そのウィンドウのタブの[x]をクリックします。

図4-34　[Memory]ウィンドウの内容



デバイスメモリの変更

メモリの値を変更するにはコードをデバッグする必要があります。コード実行中はメモリを変更できません。

Note: 変更したデータは、デバッグにのみ反映されます。アプリケーション コードは変更されません。

メモリの値を変更するには以下の情報を使います。

- 列をクリックしてデータを選択または入力すると、[Memory]ウィンドウ内の値を変更できます。ウィンドウによっては、変更されたデータを赤いテキストで表示します。
- [Fill Memory]機能は、ほとんどの[Memory]ウィンドウのコンテクスト(右クリック) メニューに表示されます。
- プログラムメモリの場合、変更を表示するにはリビルドする必要があります。変更したデータをデバイスに書き込んでデバッガを起動するには、[Debug] > [Discrete Debugger Operation]を選択します。

コンテクスト メニューを使った[Memory]ウィンドウのオプション設定

[Memory]ウィンドウ内で右クリックすると、各種オプション(表示オプション、メモリの書き込み、テーブルのインポート/エクスポート、ファイルへの出力等)を含むコンテクスト メニューが開きます。メニューの内容はウィンドウによって異なります。詳細は12.「MPLAB X IDEのウィンドウとダイアログ」を参照してください。

[Memory]ウィンドウの更新

プログラムメモリ、EEPROM、ユーザIDメモリ、コンフィグレーション ビットを表示する[Memory]ウィンドウは、以下の方法で更新できます。

1. デバッグ中の場合、プログラムを一時停止します。ただし、ツールとデバイスがデバッグ読み出し(以下参照)をサポートしている場合、一時停止の必要はありません。
2. [Read Device Memory]アイコン  をクリックします。

デバッグ読み出し

ほとんどのデバイスでは、デバイスメモリの読み出し前にプログラムを一時停止する(デバッガ セッションを終了する)必要があります。しかし、中にはデバッグモード中の読み出し(デバッグ読み出し)が可能なデバイスもあります。この機能を使えるかどうかは、デバッグ中に[Read Device Memory]アイコンが灰色表示になるか否かによって判別できます。

現在、MPLAB REAL ICEインサーキット エミュレータとMPLAB ICD 3がデバッグ読み出しに対応しています。

デバッグ読み出しは、ターゲットのオシレータ速度で実行されるため、ターゲットが非常に低速で動作している場合、読み出されるまでに長時間を要する可能性があります。デバッグ セッションを終了してから読み出しを実行する事で、強制的に高速のICSP読み出しを使う事もできます。デバッグ セッション中以外は必ずICSP読み出しが実行されるためです。

4.19 [Configuration Bits]ウィンドウでのコンフィグレーション値の設定

[Window] > [Target Memory Views]から[Configuration Bits]を選択すると、[Configuration Bits]ウィンドウが開きます。メモリウィンドウの使い方は4.18「デバイスメモリの表示と変更」を参照してください。

[Configuration Bits]ウィンドウは、コンフィグレーション ビットの設定作業に便利です。ただし、量産ではコンフィグレーション ビットをコードで設定する必要があります。

4.19.1 [Configuration Bits]ウィンドウの使い方

デバッグ セッションで[Configuration Bits]ウィンドウを使うと、コンフィグレーション ビットを一時的に変更できます(4.18 「デバイスメモリの表示と変更」参照)。設定が決まったら、以下のセクションで説明する方法で保存できます。

コンフィグレーション ビットを編集できるのはデバッグ セッション中のみです。リビルドすると[Configuration Bits]ウィンドウによる変更は全て失われます。

各種デバイスのコンフィグレーション ビットの設定に関するまとめは、14.「コンフィグレーション設定のまとめ」に記載しています。

Note: AVRまたはSAMデバイスの場合、このウィンドウに赤字が表示されると、何らかの対応が必要です。マウスカーソルをその上に置くと、指示がポップアップ表示されます。

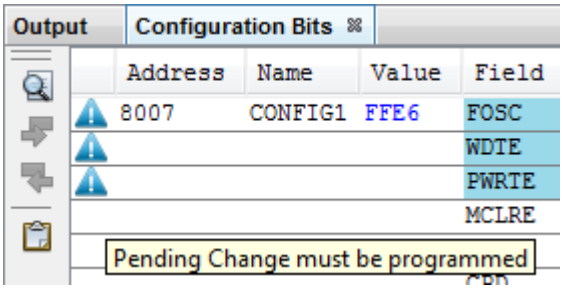
コンフィグレーション設定の編集

他のメモリウィンドウではデバッグモードにする必要がありますが、このウィンドウでは常時値を編集できます。ただし、リビルドすると[Configuration Bits]ウィンドウによる変更は全て失われます。

編集可能なフィールドの既定値は、[Configuration Bits]ウィンドウの最初の画面が表示される時と[Reset to Defaults]メニューを操作する時に表示されます。コンフィグレーション ビットがソースコードで表現される場合、全ての編集可能なフィールドの値は、ソースコードで割り当てられた値に初期化されます。これは、ビルド完了後とロード成功後に行われます。

コンフィグレーション ウィンドウで既定値を変更すると、その変更はターゲット デバイスに書き込まれるまで永続的ではなく一時的な変更である旨のメッセージが表示されます。

図4-35　保留中の変更



値は以下のように変更できます。

- **選択ボックスの値**: 選択ボックスの値は[Option]または[Setting]列に表示されます。フィールドの列の値をクリックし、リストから選択して変更します。
- **手動入力値:** 手動入力値は[Option]または[Setting]列に表示されます。[Option]列のテキストは、選択可能なレンジ(例: [User range: 0x0 - 0x3F]がある事を示します。[Setting]列のテキストは、値を入力する必要がある事(例: [Enter Hexadecimal value])を表示します。

  編集したいフィールドの[Option]または[Setting]列のテキストをクリックすると、対応する[Value]列が編集可能になります。入力値は絶対値に対して検証されるため、編集したフィールドの値が適用される実際の位置を考慮する必要はありません。入力値を「0」パディングして実際のマスク位置に近似しようとすると、フィルタがかけられ可能な範囲で絶対値に修正されます。

[Generate Source Code to Output]

[Configuration Bits]ウィンドウの下のボタン[Generate Source Code to Output]をクリックします(以下の最初の図)。[Output]ウィンドウにコンフィグレーション ビットの設定が生成され、コードにコピーできるようになります(以下の2番目の図)。

図4-36　[Configuration Bits]ウィンドウ - [Generate Source Code to Output]

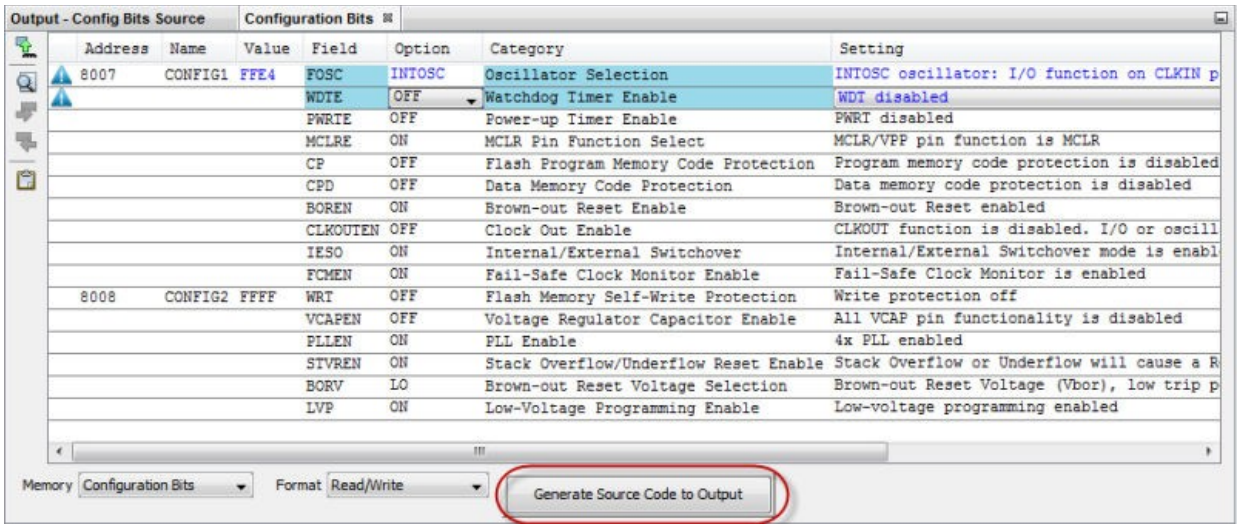
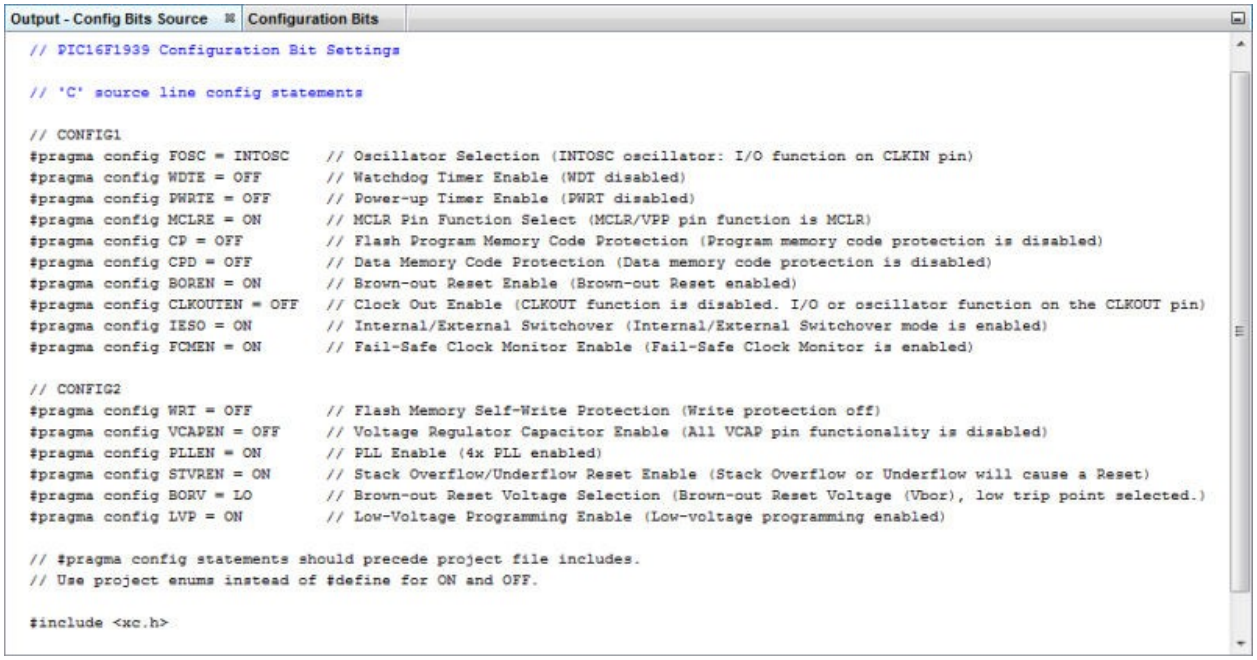


図4-37　[Output]ウィンドウ - 生成されたソースコード



[Insert Source Code in Editor]

[Insert Source Code In Editor]は、[Configuration Bits]ウィンドウのコンテクスト メニューまたはサイドバーの[Clipboard]アイコンから選択できます(下の最初の図)。エディタ内でメイン プロジェクトを表示または作成し、コンフィグレーション ビットの設定先にカーソルを置いて、アイコンをクリックします(下の2番目の図)。

図4-38　[Configuration Bits]ウィンドウ - [Insert Source Code in Editor]

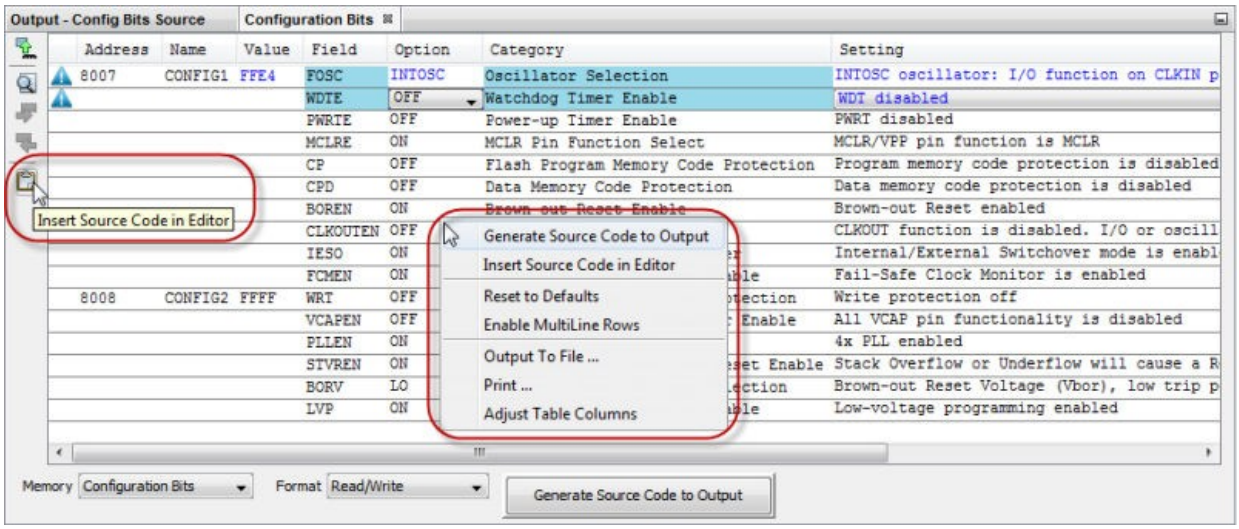
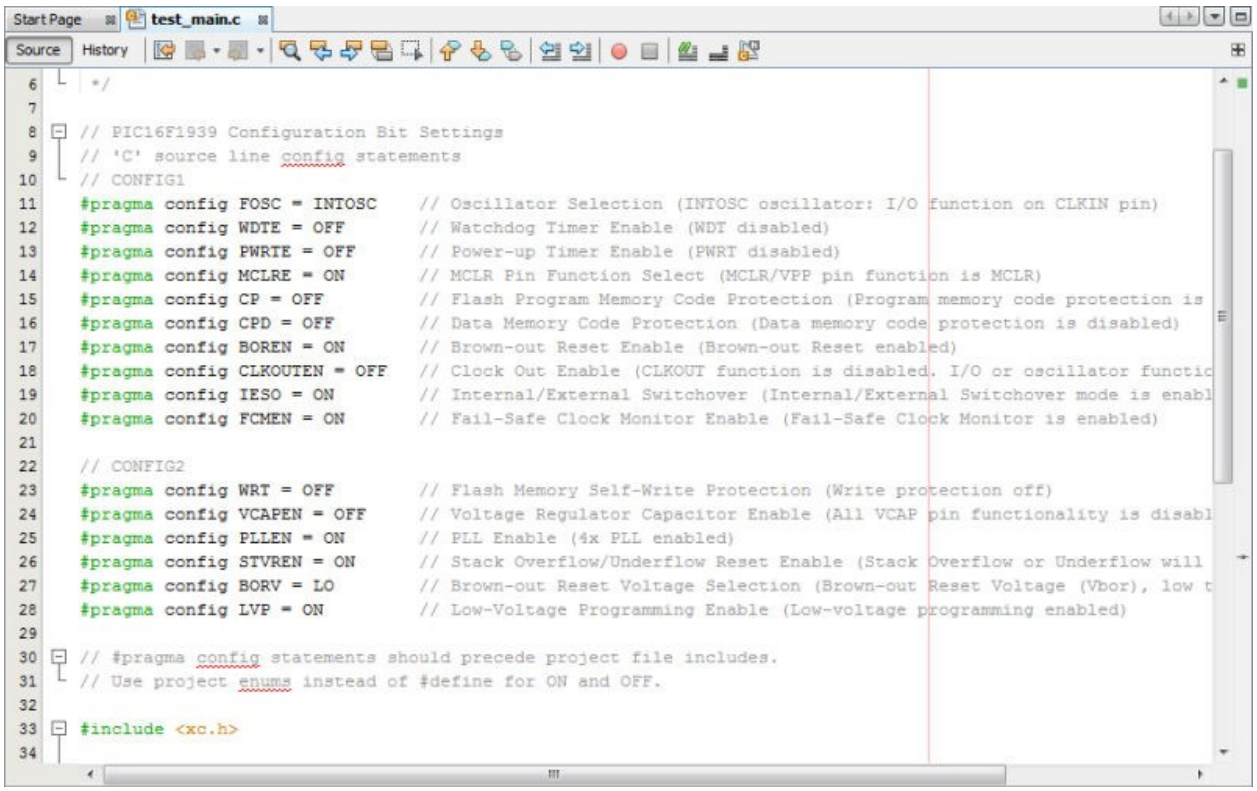


図4-39　[Editor]ウィンドウ - 生成されたソースコード



[Program Device for Debugging]

IDEツールバー アイコン[Program device for debugging]をクリックすると、[Configuration Bits]ウィンドウの現在値がデバイスにプログラミングされます。4.20「デバイスのプログラミング」を参照してください。

4.19.2　アセンブリ プロジェクトでのコンフィグレーション ビットの設定

MPLAB XCのアセンブリ プロジェクトでも[Configuration Bits]ウィンドウを使えます。(MPASMまたはMPLAB ASM30のプロジェクトでは使えません。) 簡単なC言語ファイルをプロジェクトに追加して、生成した`#pragma config`マクロを含めます。
そうすれば、アセンブリ`config`ワードを手入力する必要はありません。

アセンブリ`config`ワードと`#pragma config`の間に違いはありません。両方とも1プログラムワードを書き込み、結果としてプログラムサイズも同じです。ただし、`#pragma config`構文の方が柔軟で、検証、フォーマット、コメント機能が優れています。

4.20　デバイスのプログラミング

コードのデバッグ後は、そのコードをターゲット デバイスにプログラミングします。

プロジェクトのプログラミング プロパティの設定

[Project Properties]ウィンドウでプログラミング オプションを設定します。

1. [Projects]ウィンドウでプロジェクト名を右クリックして[Properties]を選択します。
2. [Categories]でプログラミングに使うハードウェア ツール(例: PM3)を選択します。
3. [Option categories]で[Memories to Program]の設定を確認します。
   [Preserve Memory]オプションを使う場合、コード保護が有効でない事を確認します。プログラマが書き込みに必要なデータを読み出し、デバイスのバルク消去を実行し、デバイスを再プログラミングして、保護する領域にデータを再度書き込む事で、コードは保護されます。
4. [Option categories]で[Program Options]の設定を確認します。
5. ハードウェア ツールによって、プログラミングのオプション カテゴリが異なります。各オプション カテゴリを確認しプロジェクトに対して設定が正しい事を確かめます。

6. DMCI、モータ制御アプリケーション等と一緒にRTDMを使う場合、[Loading]カテゴリの[Load symbols when programming or building for production (slows process)]チェックボックスにチェックを入れる必要があります。

プログラミング オプションの詳細は、ハードウェア ツールの文書を参照してください。

必要なプログラミング オプションを設定したら、デバイスをプログラミングできます。

プログラミングの実行

デバッグ済みコードでターゲット デバイスをプログラミングするには、ツールバーの[Make and Program Device Project]ボタンをクリックします。

表4-9に、その他のプログラミング関連の機能を示しています。各アイコンの先頭の機能は、ボタンをクリックするだけで起動します。その他の機能を選択するにはボタンアイコン横の下矢印をクリックします。

表4-13　プログラミング関連機能のツールバーボタン

ボタン アイコン	機能	説明
	Make and Program Device	(必要に応じて)プロジェクトがビルドされ、デバイスがプログラミングされます。プログラミングが完了すると、即座にプログラムの実行が始まります。
	Program Device for Debugging	デバイスがデバッグイメージに基づいてプログラミングされます。プログラミングが完了すると、即座にプログラムの実行が始まります。
	Program Device for Production	デバイスが量産イメージに基づいてプログラミングされます。プログラミングが完了すると、即座にプログラムの実行が始まります。
	Programmer to Go PICkit 3	PICkit 3のProgrammer to Go機能を使います。
	Read Device Memory	ターゲット デバイスのメモリの内容をMPLAB X IDEに転送します。
	Read Device Memory to File	ターゲット デバイスのメモリの内容を指定したファイルに転送します。
	Read EE/Flash Data Memory to a File	ターゲット データメモリの内容を指定したファイルに転送します。
	Hold In Reset	デバイスのリセットと実行をトグルします。

MPLAB統合量産環境を使ったプログラミング

MPLAB X IDEは、全てのプログラミング機能を備えている訳ではありません。その他のプログラミング機能のサポートについては、MPLAB X IDEに付属してインストールされるMPLAB IPEを参照してください。

	MPLAB IPEのデスクトップ アイコン

5. 追加作業

以下は、MPLAB X IDEにおける追加作業の要約です。

表5-1　追加作業の実行

1 プロジェクト関連	1. プロジェクトでデバイスパックを使う方法を学ぶ(デバイスパックを使う方法) 2. MPLAB X IDEプロジェクトの開き方を選択する(MPLAB X IDEプロジェクトを開く) 3. MPLAB X IDEにMPLAB IDE v8プロジェクトをインポートする(MPLAB IDE v8プロジェクトのインポート) 4. [Prebuilt Projects]インポート ウィザードを使ってビルド済みのイメージ(HEX、ELF、COF)を開く(ビルド済みのプロジェクト) 5. ロード可能なプロジェクト、ファイル、シンボルを使ってプロジェクトのHEXファイルを結合または置換する(ロード可能なプロジェクト、ファイル、シンボル)(一般的なアプリケーションでは、ロード可能なプロジェクトとファイル: ブートローダのように、ロード可能なプロジェクト/ファイルを使ってブートローダとアプリケーション コードを結合しています。) 6. ライブラリ プロジェクトを作成し、その出力をライブラリとしてビルドする(ライブラリ プロジェクト) 7. [Import Atmel Studio 7]または[Atmel START Project]ウィザードを使って、StudioまたはStartプロジェクトを開く(Atmel Studio 7またはAtmel STARTプロジェクトのインポート) 8. その他の組み込みプロジェクトまたはサンプル プロジェクトからプロジェクトを作成する(その他の組み込みプロジェクト、サンプル プロジェクト) 9. Microchip社コンパイラがサポートしているファイルタイプだけでなく他のタイプのファイルを使う(他のタイプのファイルの使い方)プロジェクトで使う既定値のファイル テンプレートを変更するためにコード テンプレートを変更または作成する(コード テンプレートの変更または作成) 10. プロジェクトで使うハードウェアまたは言語ツールを切り換える(ハードウェア ツールまたは言語ツールの切り換え) 11. プロジェクト フォルダとエンコードを変更する(プロジェクト フォルダとエンコードの変更)
2 コードのデバッグ	1. ストップウォッチを使ってブレークポイント間のタイミングを調べる(ストップウォッチの使用) 2. ウィンドウに逆アセンブルコードを表示する([Disassembly]ウィンドウの表示) 3. 関数コールをたどるため、コールスタックまたはコールグラフを表示する(コールスタックの表示、コールグラフの表示) 4. ダッシュボード画面にブレークポイント リソース、チェックサム、メモリ使用率等のプロジェクト情報を表示する(ダッシュボードの表示) 5. 設計およびデバッグ中にプロジェクトのレジスタを表示する(プロジェクト レジスタの表示([I/O View]))
3 コードの管理	1. リファクタリングおよびプロファイリング ツールを使ってコードを改善する(コードの改善*) 2. 内蔵のローカルファイル履歴機能を使ってソースコードを管理する(ローカルファイル履歴によるソースコードの管理) またはリビジョン管理システムを使ってソースコードを管理する(リビジョン管理システムによるソースコードの管理) 3. チームサーバと問題追跡システムを使って、コード開発とエラー追跡を連携する(コード開発とエラー追跡の連携*) 4. アプリケーションに適した最適化作業を決定するためにMPLAB XCコンパイラのFreeライセンスとPROライセンスを比較するMPLAB XCコンパイラのFreeライセンスとPROライセンスの比較)
4 機能の追加	1. コード開発を支援するためにプラグインツールを追加する(プラグインツールの追加)
* この機能については、**[Start Page]**の**[My MPLAB X IDE]**タブを開き、[Extend MPLAB]セクションの[Selecting Simple or Full-Featured Menus]トピックを参照してください。	

5.1 デバイスパックを使う方法

MPLAB X IDEがデバイスをシミュレートまたはエミュレートするには、デバイス固有の情報が必要です。この情報はデバイスファイル(.PIC)に保存されておりデバイスの電力要件、プログラミング方式、アーキテクチャ等のデータを含んでいます。

MPLAB X IDE v5.00以降のデバイスファイルは、バージョン管理されたデバイスファミリ パック(DFP)にまとめられています。MPLAB X IDEはデバイスパックと一緒にインストールされますが、デバイスパックをアップグレードして新しいデバイス、新しいデバイス機能、デバイスバグ修正のサポートを追加する事ができます。

例えば、Windowsコンピュータ上でPIC16F19197のデバイスファイルを収めたDFPのパスは以下の通りです。

```
C:\Program Files (x86)\Microchip\MPLABX\v5.00\packs\Microchip\PIC16F1xxxx_DFP
\1.0.38\edc\PIC16F19197.PIC
```

- インストール場所: `C:\Program Files (x86)\Microchip\MPLABX`
- バージョン: `\v5.00`
- パック保存フォルダ: `\packs`
- デバイスファミリ パック: `\Microchip\PIC16F1xxxx_DFP`
- パックバージョン: `\1.0.38`
- デバイスファイル保存フォルダ: `\edc`
- デバイスファイル: `\PIC16F19197.PIC`

詳細は以下のヘルプを参照してください。

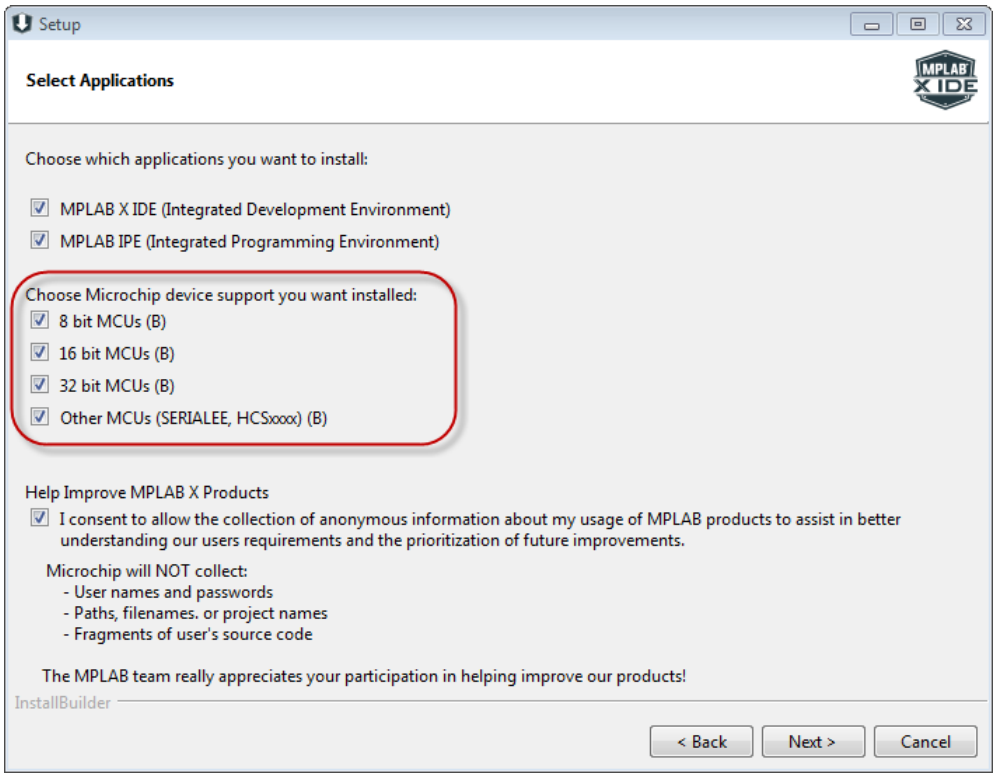
Device Packs in Projects

Device Pack Frequently Asked Questions

5.1.1 MPLAB X IDEインストール中のパックのインストール

MPLAB X IDEをインストールする際は、インストールするMicrochip社デバイスサポートを選択できます([Choose Microchip device support you want installed])。これにより、選択したデバイス アーキテクチャのパックがインストールされます。必要なものだけをインストールする事でディスク容量が節約でき、インストール時間も短くて済みます。

これは、分散インストーラ(distributed installer)とも呼ばれます。



MPLAB X IDEのインストールに関する情報はMPLAB X IDE Readmeに記載しています。

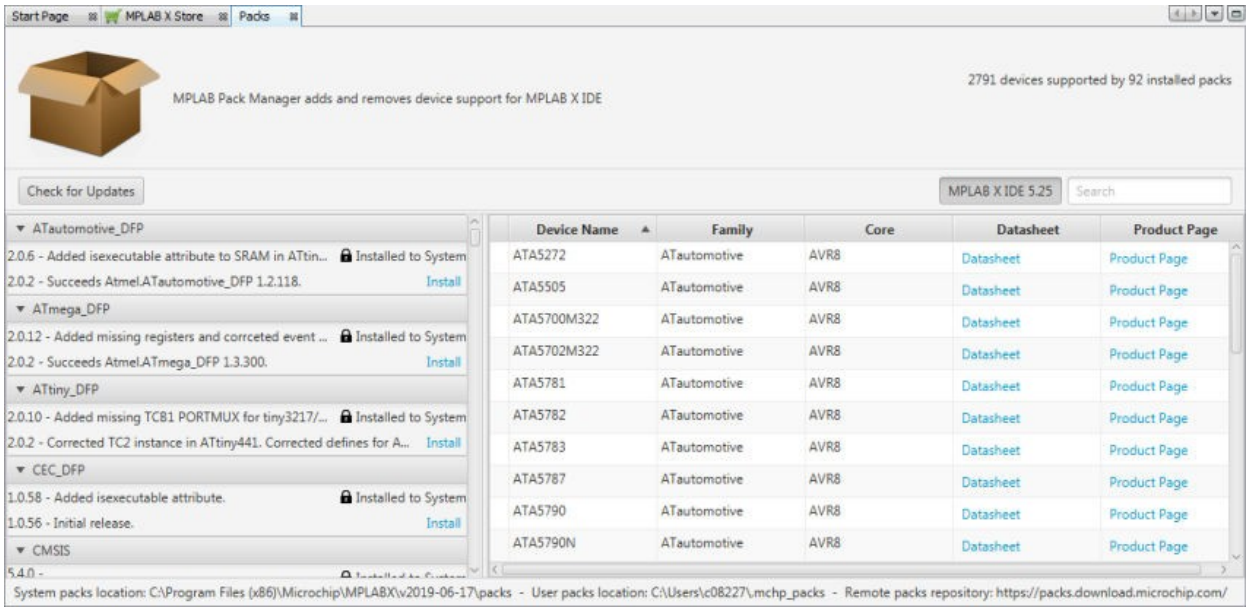
5.1.2 MPLAB X IDEインストール後のパックのインストール

[Tools] > [Packs]を選択すると、MPLAB Pack Manager経由でインストール可能なバージョン別パックリストが開きます。

初めて開くと、最新IDEリリースと互換のパックのみ表示されます。IDEリリースボタンをクリックするとフィルタ機能が無効になり、全バージョンのパックが表示されます。下図はリストの例です。

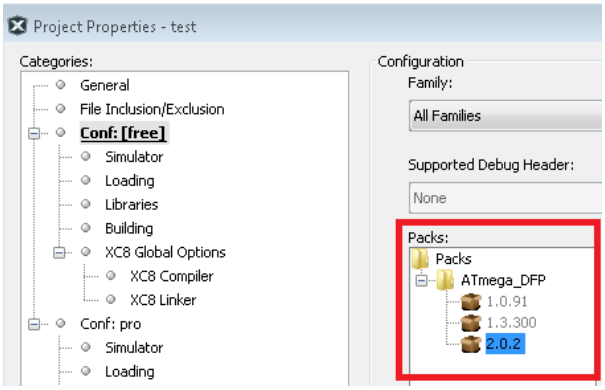
MPLAB X IDE 5.25	フィルタがかけられたパックリスト
MPLAB X IDE 5.25	全バージョンのパックを表示

図5-1　MPLAB Pack Manager



5.1.3　インストールしたパックの切り換え

MPLAB X IDEで開いたプロジェクトの場合、[Projects]ウィンドウでプロジェクト名を右クリックし、[Properties]を選択して[Project Properties]ウィンドウを開きます。[Packs]リストに、プロジェクト デバイスに使えるパックが表示されます。



灰色で表示されたパックは、使用中のMPLAB X IDEバージョンには含まれていません。パックのインストール方法は5.1.2「MPLAB X IDEインストール後のパックのインストール」を参照してください。フィルタをOFFにして全バージョンを表示し、目的のバージョンをインストールします。

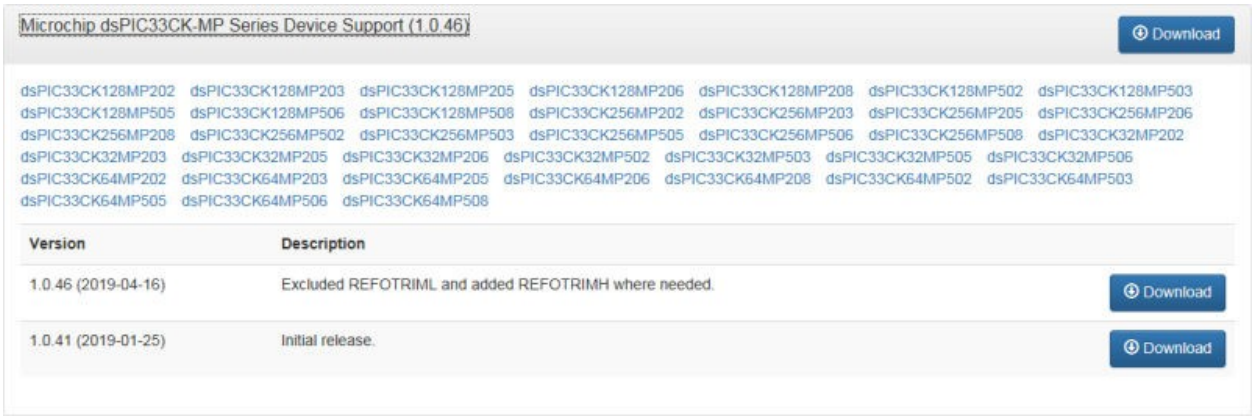
Note:　[Project Properties]でデバイスパックがインストールされていないデバイスに切り換えた場合、デバイスパックのインストールを促される事はありません。MPLAB Pack Managerを使って、自分でインストールする必要があります。

5.1.4　ウェブ上のパック

以下のウェブサイトでパックバージョンの検索とダウンロードが行えます。
https://packs.download.microchip.com/

パック名をクリックすると、サポートしているデバイス、パックバージョン履歴等の情報が表示されます。

図5-2　Microchip社のパックリポジトリ



5.2　MPLAB X IDEプロジェクトを開く

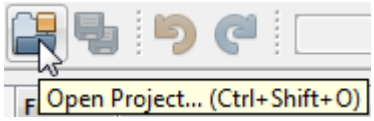
既存のプロジェクト(作成したプロジェクトまたはダウンロードしたサンプル)がある場合、いくつかの方法でこのプロジェクトを開く事ができます。

IDEメニューから開く

[File] > [Open Project]または[File] > [Open Recent Project]を選択します。

アイコンで開く

[**Open Project**]アイコンをクリックします。



プロジェクト フォルダをドラッグ&ドロップして開く

ファイル マネージャ ウィンドウでプロジェクト フォルダを選択します。そのフォルダを**[Editor]ペイン**にドラッグ&ドロップします(4.2「プロジェクト作成後のデスクトップ画面」参照)。プロジェクトが[Projects]ウィンドウ内に表示されます。

図5-3　フォルダのドラッグ&ドロップ

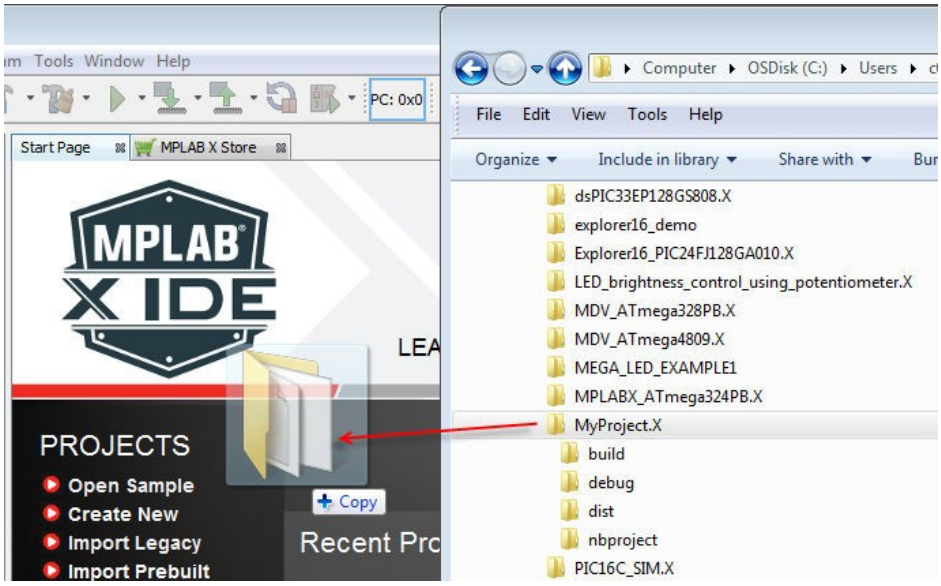
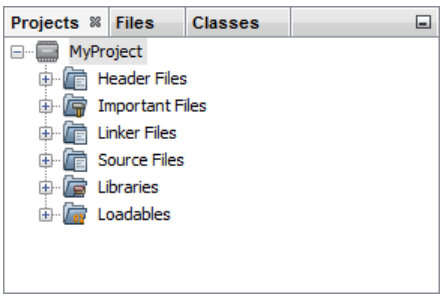


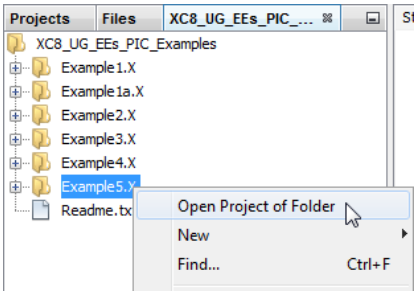
図5-4　プロジェクトが開いた状態



プロジェクトが保存されているフォルダをドラッグ&ドロップして開く

プロジェクトが保存されているフォルダを開くには、ファイル マネージャ ウィンドウでフォルダを選択します。このフォルダを**[Editor]ペイン**にドラッグ&ドロップします(前のステップ参照)。**[File]ペイン**のタブにフォルダ名の付いたウィンドウが開きます。このウィンドウには、フォルダと関連プロジェクトが表示されます。このフォルダでプロジェクトを開くには、プロジェクト名を右クリックし、[Open Project of Folder]を選択します。プロジェクトが[Projects]ウィンドウ内に表示されます。

図5-5　プロジェクトが保存されているフォルダが開いた状態



5.3　MPLAB IDE v8プロジェクトのインポート

[New Project]ウィザードを使うと、MPLAB IDE v8プロジェクトをMPLAB X IDEプロジェクトにインポートできます。以下にその際の注意事項を示します。

MPLAB IDE v8のワークスペースの設定は転送されません。

MPLAB IDE v8のワークスペースに保存されている設定(ツール設定等)は、新しいMPLAB X IDEプロジェクトに転送されません。ワークスペースに保存されている情報については、MPLAB IDE v8ヘルプ([MPLAB IDE Reference] > [Operational Reference] > [Saved Information])を参照してください。

プロジェクトの設定(コンパイラ、リンカ、アセンブラの設定等)は、新しいMPLAB X IDEプロジェクトに転送されます。

MPLAB IDE v8プロジェクトのファイルが変更される可能性

インポートするMPLAB IDE v8プロジェクトが、「MPLAB C Compiler for PIC24 MCUs and dsPIC DSCs」(MPLAB C30とも呼ぶ)とCOFFデバッグファイル フォーマットを使っている場合、以下のように変換されます。

- MPLAB X IDEプロジェクトがCOFFライブラリを使わない場合、デバッグファイル フォーマットはELF/DWARFに変換されます。MPLAB X IDEプロジェクトがCOFFライブラリを使う場合、フォーマットはCOFFのままとなります。
- MPLAB IDE v8はファイル拡張子の大文字/小文字(.cと.C等)を区別しませんが、MPLAB X IDEはこれらを区別し、.cをCのコードファイル、.Cを C++のコードファイルに関連付けます。従って、.C拡張子を持つCコードを含むMPLAB IDE v8プロジェクトをインポートすると、MPLAB X IDEはその.Cファイルを.cにリネームし、コンパイラの誤動作を防ぎます。

MPLAB IDE v8プロジェクトのビルドが変更される可能性

MPLAB X IDEにインポートしたMPLAB IDE v8プロジェクトは元のプロジェクトと同じようにはビルドされません。

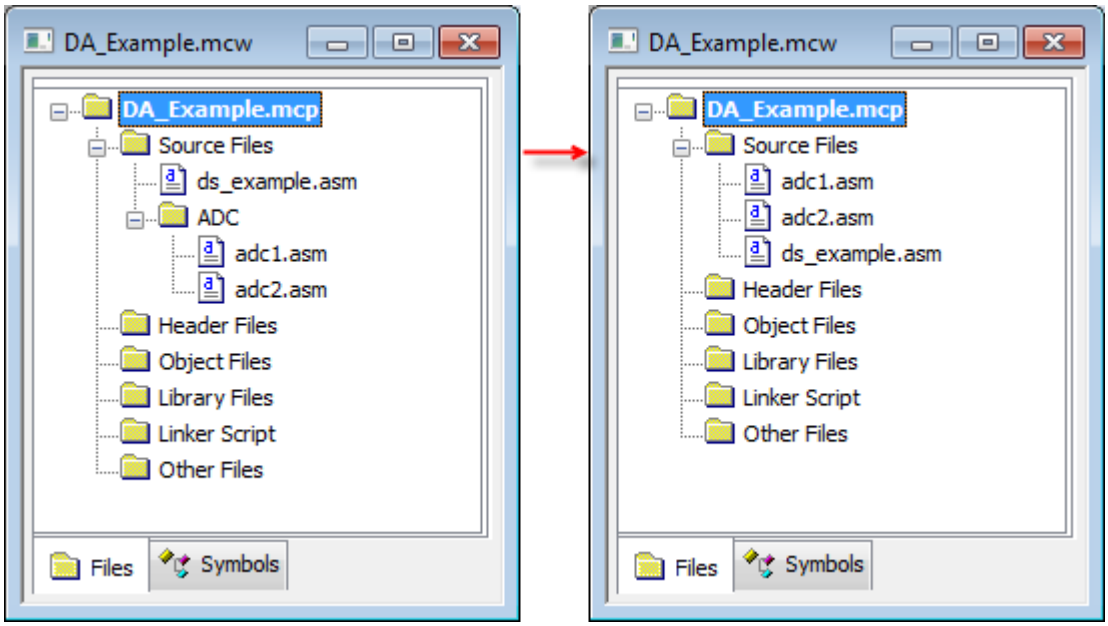
- プロジェクトコードに`__FILE__`, `assert()`, `__TIME__`, または`__DATE__`を使わないでください。ビルド(とチェックサム)が変わってしまいます。

- 論理サブフォルダを使うMPLAB IDE v8プロジェクトのチェックサムは、MPLAB X IDEにインポートされると変わってしまいます。同じチェックサムでビルドするには、論理サブフォルダ内のファイルをメイン論理フォルダ (ソースファイル、ヘッダファイル等)に移動してから、MPLAB X IDEにインポートします。

ソースファイル内の論理サブフォルダを削除する方法:

- ソースファイルの下のファイルにドラッグする
- フォルダを選択して<Delete>を押す
- または、フォルダを右クリックして[Remove from Project]を選択すると、フォルダがプロジェクトから削除されます。ただしPCからフォルダが削除される事はありません。

図5-6　ファイルの移動とサブフォルダの削除



5.3.1　[Import MPLAB IDE v8 Project]ウィザードを開く

このウィザードは以下2通りの方法で開きます。

[Import Legacy Project]ウィザードを使う方法

[Import Legacy Project]ウィザードは以下の方法で開きます。

- **[Start Page]**で**[Learn & Discover]**または**[My MPLAB X IDE]**タブをクリックし、[Projects]セクションで[Import Legacy]を選択します。
- [File] > [Import] > [MPLAB IDE v8 Project]を選択します。

[New Project]ウィザードを使う方法

[New Project]ウィザードは以下の方法で開きます。

- **[Start Page]**で、**[Learn & Discover]**または**[My MPLAB X IDE]**タブを開き、[Projects]セクションで[Create New]を選択します。
- [File] > [New Project]を選択します(または<Ctrl>+<Shift>+<N>)。
- ツールバーで[New Project]アイコン  をクリックします。

[New Project]ウィザードが起動し、プロジェクトの新規作成を支援します。

Step 1. Choose Project: [Microchip Embedded]カテゴリを選択し、プロジェクト タイプ[Existing MPLAB IDE v8 Project]からプロジェクトを選択します。**[Next>]**をクリックします。

5.3.2 [Import MPLAB IDE v8 Project]ウィザードの使用

以下の手順でMPLAB IDE v8プロジェクトをインポートします。ステップの番号は[Import Legacy Project]オプションに従っています(5.3.1「[Import MPLAB IDE v8 Project]ウィザードを開く」参照)

[Next>]をクリックすると次のステップに進みます。

Step 1. Locate MPLAB IDE v8 Project *.mcp File: レガシー プロジェクトを選択します。

Step 2. Select Device: アプリケーションで使う予定のデバイスを[Device]ドロップダウン リストから選択します。デバイスファミリを選択すると選択肢を絞り込めます。

Step 3. Select Header: このステップは、選択したデバイスでヘッダが利用可能である場合に表示されます。デバッグ用にヘッダが必要かどうか、デバイスがデバッグ回路を内蔵しているかどうかは、以下の文書で確認します。次にヘッダを使うかどうかを選択します。

- 『Processor Extension Pak and Debug Header Specification』
- 『エミュレーション拡張パック(EEP)およびエミュレーション ヘッダ ユーザガイド』

Step 4. Select Tool: アプリケーションの開発に使う開発ツールをリストから選択します。デバイスのサポートを判定する方法は4.1.6.1「プロジェクト ツールサポート」を参照してください。

Step 5. Select Plugin Board: プラグインボードを使う場合、ここで選択します。

Step 6. Select Compiler: アプリケーションの開発に使う言語ツール (コンパイラ)をリストから選択します。デバイスのサポートを判定する方法は4.1.6.1「プロジェクト ツールサポート」を参照してください。

Step 7. Select Project Name and Folder: 両方のプロジェクトの保守性を保つために、既定値の名前とフォルダは変更しない事を推奨します。下図の例を参照してください。

File Locations:

新規プロジェクトは、ソースファイルをプロジェクト フォルダにコピーしません。その代わりに、v8フォルダ内のファイル保存場所を参照します。

独立したMPLAB X IDEプロジェクトを作成するには、プロジェクトを新規作成して、MPLAB IDE v8のソースファイルを、そのプロジェクトにコピーする必要があります。

Main Project:

インポート時にこのプロジェクトをメイン プロジェクトとするために、チェックを入れます。

Project Location:

MPLAB X IDEプロジェクトの保存場所はMPLAB IDE v8プロジェクト フォルダ内ではないため、チェックを外します。

File Formatting:

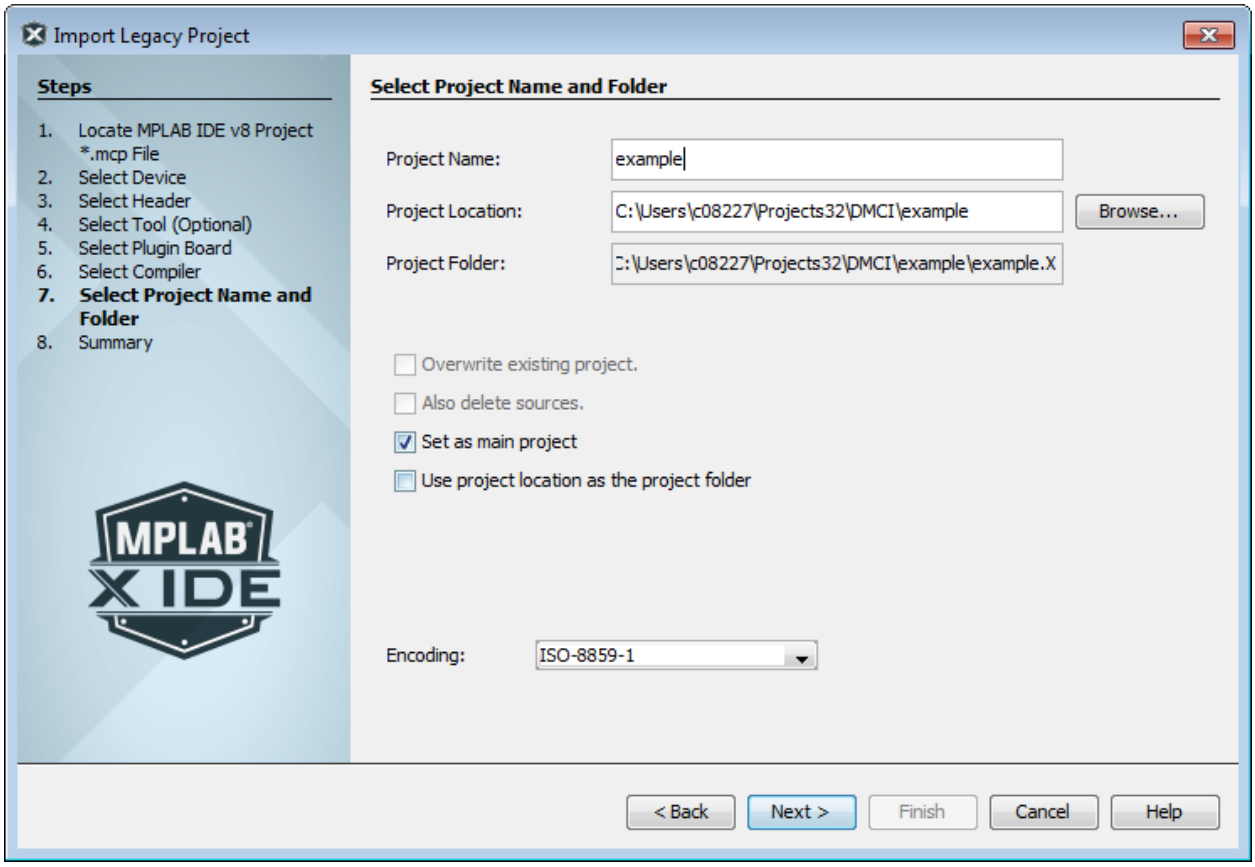
MPLAB IDE v8からプロジェクトをインポートする際に使う文字エンコードの既定値は「ISO-8859-1」です。

インポート元のプロジェクトで実際に使っているエンコードを選択する必要があります。例えば、MPLAB IDE v8のフォーマットが「950 (ANSI/OEM – Traditional Chinese Big5)」である場合、ドロップダウン リストから「Big5」を選択します。

Step 8. Summary: **[Finish]**をクリックする前に、設定した内容を確認します。不適切な設定が見つかった場合、**[Back]**ボタンを使って以前のステップに戻って修正します。

レガシー プロジェクトが[Project]ウィンドウ内に開きます。

図5-7　Step 7のダイアログ例



5.4　ビルド済みのプロジェクト

[Import Image File]ウィザードを使って、ビルド済みのロード可能イメージ(HEX、ELF、COFファイル)からプロジェクトを作成します。

Note:　デバイスにプリビルト イメージをプログラミングする場合、makeは実行しませんが**[Make and Program Device]**アイコンをクリックします。

このウィザードを開くには、[Start Page]を使う方法と[New Project]を使う方法があります。

[Import Prebuilt Project]を使う方法

[Import Image File]ウィザードは、以下の方法で開きます。

- **[Start Page]**で**[Learn & Discover]**または**[My MPLAB X IDE]**タブをクリックし、[Projects]セクションで[Import Prebuilt]へのリンクを選択します。
- [File] > [Import] > [Hex/ELF (Prebuilt) File]を選択します。

[New Project]ウィザードを使う方法

[New Project]ウィザードを開く方法は、4.1.2「[New Project]ウィザードの起動」を参照してください。[New Project]ウィザードが起動し、プロジェクトの新規作成を支援します。

- **Step 1. Choose Project:** [Microchip Embedded]カテゴリを選択し、プロジェクト タイプ[Prebuilt (Hex, Loadable Image) Project]からプロジェクトを選択します。**[Next>]**をクリックします。
- 「Import Image File」ウィザードが開きます。

[Import Image File]ウィザード

以下の手順により、イメージファイルをインポートします。ステップ番号は[Import Prebuilt Project]オプションに従っています。

[Next>]をクリックすると次のステップに進みます。

Step 1. Import Image File: イメージファイルの名前と保存場所を選択します。ダイアログで場所を選択できます。

Step 2. Select Device: アプリケーションで使う予定のデバイスを[Device]ドロップダウン リストから選択します。デバイスファミリを選択すると選択肢を絞り込めます。

Step 3. Select Header: このステップは、選択したデバイスでヘッダが利用可能である場合に表示されます。デバッグ用にヘッダが必要かどうか、デバイスがデバッグ回路を内蔵しているかどうかは、以下の文書で確認します。次にヘッダを使うかどうかを選択します。

- 『Processor Extension Pak and Debug Header Specification』
- 『エミュレーション拡張パック(EEP)およびエミュレーション ヘッダ ユーザガイド』

Step 4. Select Tool: アプリケーションの開発に使う開発ツールをリストから選択します。使うデバイスのサポートを判定する方法は4.1.6.1「プロジェクト ツールサポート」を参照してください。

Step 5. Select Project Name and Folder: 新規プロジェクトの名前と保存場所を選択します。ダイアログで保存場所を選択できます。完了したら、**[Finish]**をクリックします。

作成したプロジェクトが[Projects]ウィンドウ内に表示されます。

プロジェクトをHEXファイルとしてエクスポートする方法は12.15「[Projects]ウィンドウ - [Project]メニュー」を参照してください。

5.5 ロード可能なプロジェクト、ファイル、シンボル

複数のプロジェクトの結合、複数のHEXファイルの結合、プロジェクトとHEXファイルの結合、プロジェクトHEXファイルの置換には、ロード可能なプロジェクトおよびファイルを使います。Hexmateアプリケーションを使って、プロジェクトまたはロードしたHEXファイルを1つのファイルにマージします。このアプリケーションの使い方の詳細は、インストールしたMPLAB XC8 Cコンパイラのdocsフォルダにある『MPLAB XC8 C Compiler User's Guide』(DS50002053)を参照してください。9.5 「エラー」のセクション[HEXMATE Conflict Report Address Error]も参照してください。

RTDM (Real Time Data Monitoring)とDMCI (Data Monitoring and Control Interface)を使う場合(モータ制御アプリケーション等)のように、量産ビルドまたはプログラミング中にデバッグシンボルをロードするには、ロード可能なシンボルを使います。

ロード可能なプロジェクトまたはファイルは、ブートローダとアプリケーションを結合したコードを作成する場合に便利です。5.6「ロード可能なプロジェクトとファイル: ブートローダ」を参照してください。

以下に、現在のプロジェクトとロード可能なプロジェクトおよびファイルの組み合わせを示します。

表5-2　ロード可能プロジェクト/ファイルの組み合わせ

現在のプロジェクト	ロード可能プロジェクト/ファイル	注意
スタンドアロン、 既存のMPLAB IDE v8 ライブラリ	スタンドアロン	なし
	HEXファイル	なし
	ELF/COFファイル	デバッグできますがビルドはできません。 エラーが[Output]ウィンドウに表示されます。
ビルド済み(HEX)	HEXファイル	重複しているメモリ領域は自動ではチェックしません。デバッグできますがビルドはできません。 [Build]ボタンは無効になります。
	ELF/COFファイル	
ビルド済み(COF/ELF)	HEXファイル	
	ELF/COFファイル	

以下にオプションを示します。

表5-3　ロード可能プロジェクト/ファイルのオプション

ロード可能プロジェクトの追加	現在のプロジェクトに既存のプロジェクトをロードします。 現在のプロジェクトをビルドする際に、全てのプロジェクトがビルドされHEXファイルが1つに結合されます。デバッグファイルも全てが結合されます(ELFまたはCOF)。
ロード可能ファイルの追加	現在のプロジェクトに既存のHEXファイルをロードします。 現在のプロジェクトをビルドする際に、ロードしたHEXファイルとその他のHEXファイルが1つのファイルに結合されます。 Note: HEXファイルをロードしたプロジェクトはデバッグできません。デバッグが必要な場合、ロード可能プロジェクトを使います。
代替ファイルの追加	代替HEXファイルをロードします。 このオプションを使うとビルド後の処理を設定できます。この処理では、プロジェクトのHEXファイルを別の場所にコピーまたは移動し、HEXMATE等のツールを使ってファイルを別のHEXファイルとマージし、IDEにロードする事ができます。

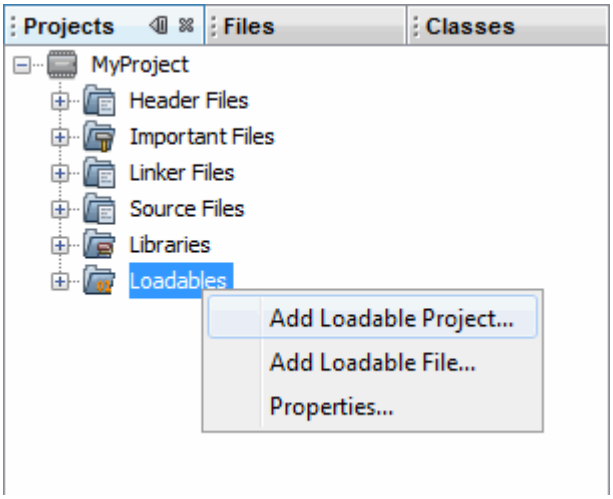
5.5.1　[Projects]ウィンドウ – [Loadables]フォルダによる設定

[Projects]ウィンドウの[Loadables]フォルダを右クリックして、以下のオプションを選択します(下図参照)。

- Add Loadable Project – 既存のプロジェクトを選択して、現在のプロジェクトに追加します。さらにプロジェクトを追加するには手順を繰り返します。
- Add Loadable Files – 既存のHEXファイルを選択して、現在のプロジェクトに追加します。さらにHEXファイルを追加するには手順を繰り返します。
- Properties – ロードのために[Project Properties]ウィンドウを開きます。5.5.2「[Project Properties]ウィンドウ – [Loading]カテゴリによる設定」を参照してください。

全てのプロジェクトをビルドしてHEXファイルを1つに結合するには、現在のプロジェクトをビルドします。デバッグファイルも全て結合されます。

図5-8　[Projects]ウィンドウ – [Loadables]フォルダ

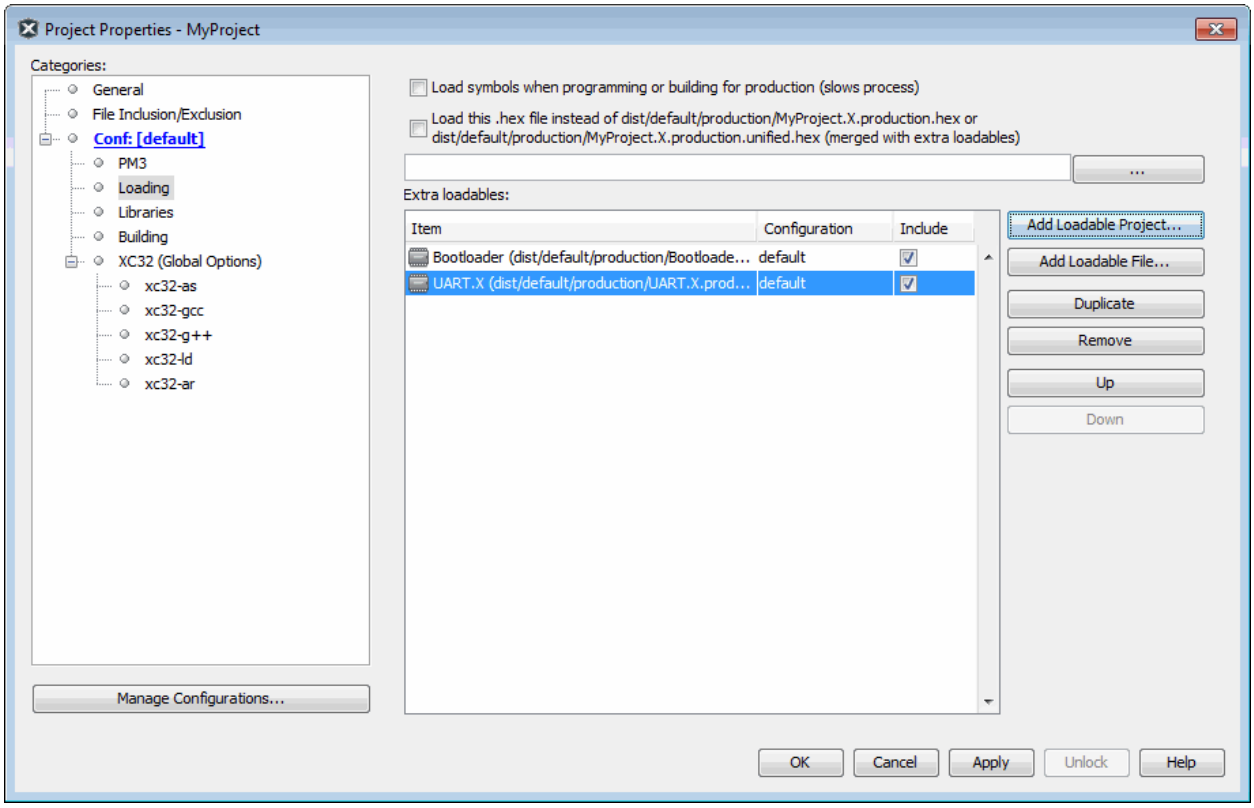


5.5.2　[Project Properties]ウィンドウ – [Loading]カテゴリによる設定

[Project Properties]ウィンドウを開き([File] > [Project Properties])、[Loading]カテゴリをクリックします。

- 現在のプロジェクトを他のプロジェクトと結合する方法
- 現在のプロジェクトHEXファイルを他のHEXファイルと結合する方法
- 代替HEXファイルをロードする方法
- プログラミング/ビルド中にデバッグシンボルをロードする方法

図5-9　[Project Properties]ウィンドウ – [Loading]カテゴリ

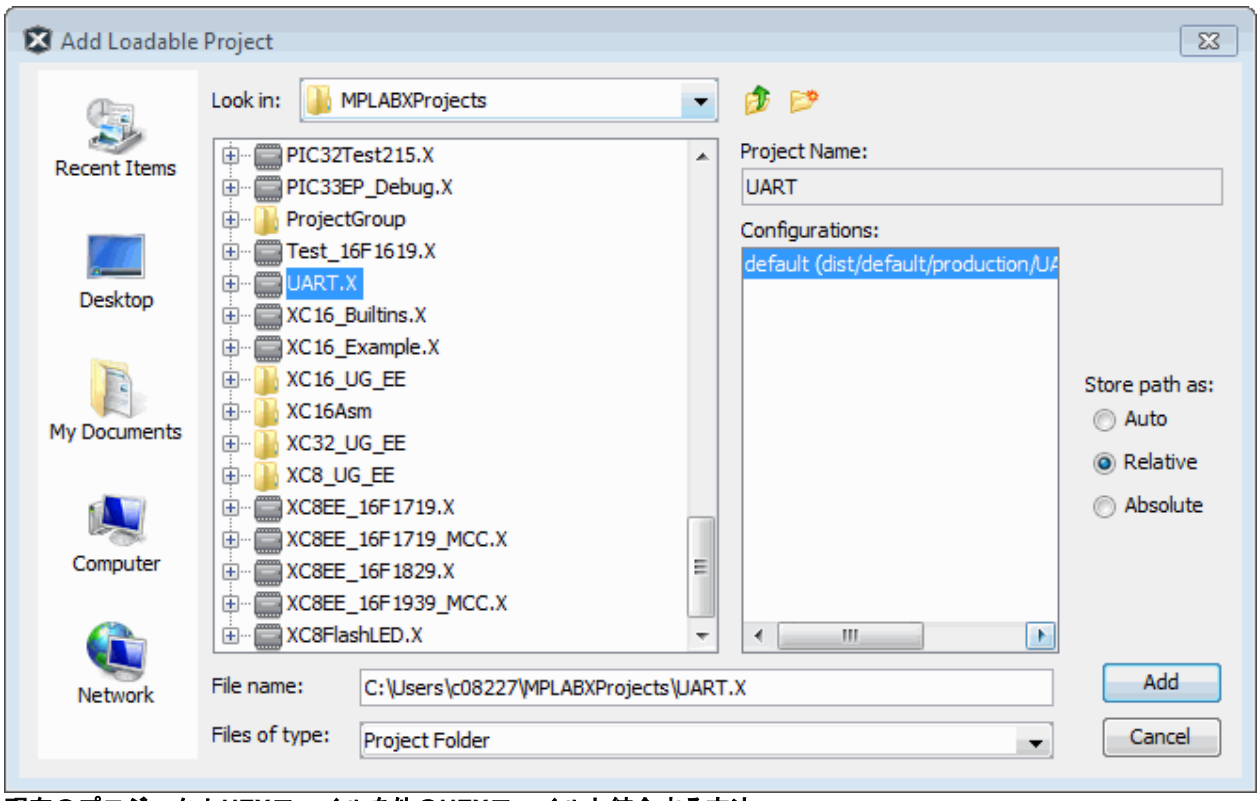


現在のプロジェクトを他のプロジェクトと結合する方法

1. **[Add Loadable Project]**をクリックします。[Add Loadable Project]ダイアログが開きます。
2. プロジェクトを選択します。
3. [Configurations]からこのプロジェクトで使うビルド コンフィグレーションを選択します。このプロジェクトに他のコンフィグレーションを追加していなければ、ここには[default]しか表示されません。
4. [Store path as]から、ロード可能プロジェクトへのパス情報の保存方法を選択します。以下から選択します。
   - 4.1. Auto – ロード可能プロジェクトへのパスを相対パスと絶対パスのどちらにするかを、その保存場所に基づいてMPLAB IDEが決定できるようにします。
     メイン プロジェクトのフォルダにロード可能プロジェクトが含まれている場合、参照は相対パスになります。それ以外の場合、参照は絶対パスになります。
   - 4.2. Relative – ロード可能プロジェクトを(メイン プロジェクトへの)相対パスで参照します。これは、パスを追加する際に推奨する方法です。
   - 4.3. Absolute – ロード可能プロジェクトを絶対パスで参照します。
     このモードが便利な場合もありますが、プロジェクトを別のシステムに移動する場合、しばしばパスの修正が必要です。
5. **[Add]**をクリックしてダイアログを閉じます。
6. 現在のプロジェクトのビルドにこのプロジェクトを含める場合、[Project Properties]ウィンドウで[Include]にチェックを入れます。
7. 複数のプロジェクトを追加した場合、ここに表示される順序が、現在のプロジェクトのHEXファイルにこれらのHEXファイルが追加される順序を決定します。追加の順序を変更するには、**[Up]**と**[Down]**ボタンを使います。
8. **[Apply]**または**[OK]**をクリックして変更を確定します。

次に現在のプロジェクトをビルドする際は、ここに表示されたプロジェクトも([Include]がチェックされていれば)ビルドされ、それらのHEXファイルが現在のプロジェクトのHEXファイルと結合されて、1つの出力HEXファイルが生成されます。デバッグファイルも全て結合されます。

図5-10　[Add Loadable Project]ダイアログ



現在のプロジェクトHEXファイルを他のHEXファイルと結合する方法

1. **[Add Loadable File]**をクリックします。[Add Loadable File]ダイアログが開きます。
2. HEXファイルを選択します。
3. [Store path as]から、ロード可能ファイルへのパス情報の保存方法を選択します。以下の方法から選択します。
    3.1. Auto – ロード可能ファイルへのパスを相対パスと絶対パスのどちらにするかを、その保存場所に基づいてMPLAB IDEが決定できるようにします。
    メイン プロジェクトのフォルダにロード可能ファイルが含まれている場合、参照は相対パスになります。それ以外の場合、参照は絶対パスになります。
    3.2. Relative – ロード可能ファイルを(メイン プロジェクトへの)相対パスで参照します。これは、パスを追加する際に推奨する方法です。
    3.3. Absolute – ロード可能ファイルを絶対パスで参照します。
    このモードが便利な場合もありますが、プロジェクトを別のシステムに移動する場合、しばしばパスの修正が必要です。
4. **[Add]**をクリックしてダイアログを閉じます。
5. [Project Properties]ウィンドウで、複数のファイルを追加した場合、ここに表示される順序が、現在のプロジェクトのHEXファイルにこれらのHEXファイルが追加される順序を決定します。追加の順序を変更するには、**[Up]**と**[Down]**ボタンを使います。
6. **[Apply]**または**[OK]**をクリックして変更を確定します。

次に現在のプロジェクトをビルドする際は、ここに表示されたHEXファイルが現在のプロジェクトのHEXファイル(`CurrentProject.X.Production.hex`)と結合されて、1つの出力HEXファイルが生成されます。このファイル(`CurrentProject.X.unified.hex`)は、同じフォルダ内に現在のプロジェクトHEXファイルとして生成されます。

代替HEXファイルをロードする方法

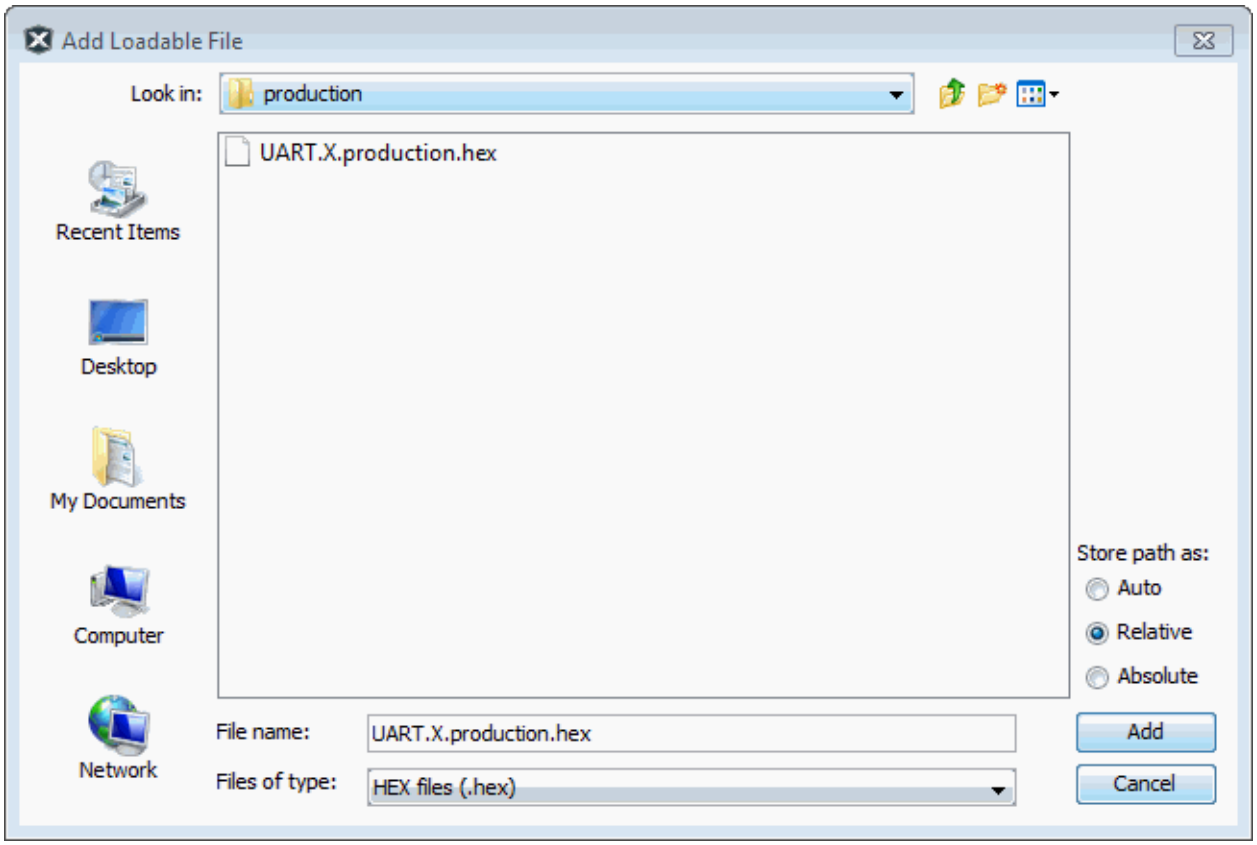
1. [Load this .hex file instead of ...]にチェックを入れます。
2. [...]ボタンをクリックして、[Add Loadable File]ダイアログを開きます。このダイアログを使ったHEXファイルの追加の詳細は、前のトピックを参照してください。

次に現在のプロジェクトをビルドする際、その完了時に代替HEXファイルがロードされます。

プログラミング/ビルド中にデバッグシンボルをロードする方法

RTDMとDMCIを使う場合(モータ制御アプリケーション等)、[Load symbols when programming or building for production (slows process)]にチェックを入れると、プログラミング中にデバッグシンボルをロードする事ができます。それ以外の場合、チェックは外したまま(既定値)にしてプログラミングとビルドの速度を向上させます。

図5-11　ロード可能ファイルの追加



5.5.3　ロード可能プロジェクト/ファイルの推奨使用法

ロード可能プロジェクト/ファイルは、以下の方法で使う事を推奨します。

1. デバイスのコンフィグレーション ビットの設定と初期化を行うプロジェクトをメイン プロジェクトとして選択します(右クリックして[Set as Main Project]を選択)。メイン プロジェクトと競合するため、ロード可能プロジェクト/ファイルにデバイス コンフィグレーション設定を持たせる事はできません。
2. 他のプロジェクトを、ロード可能プロジェクト/ファイルとしてメイン プロジェクトに追加します。ロードするプロジェクトに複数のプロジェクト コンフィグレーションが設定されている場合、ロードする際にそれらを全て指定します。

ロード可能プロジェクト/ファイルを含むプロジェクトをメイン プロジェクトに指定しておく事で、どのロード可能プロジェクト/ファイルに変更が生じても、ビルドを実行する際にその変更が確実に反映されるようにします。

5.6　ロード可能プロジェクトおよびファイル: ブートローダ

ブートローダとアプリケーションのコードを結合する方法

1. アプリケーション用とブートローダ用にそれぞれ1つずつプロジェクトを作成します。
2. ブートローダのプロジェクトまたはHEXファイルをアプリケーション プロジェクトにロードします。この方法は5.5.1「[Projects]ウィンドウ – [Loadables]フォルダによる設定」または5.5.2「[Project Properties]ウィンドウ – [Loading]カテゴリによる設定」を参照してください。

次にアプリケーション プロジェクトをビルドすると、ブートローダとアプリケーションのHEXファイルが結合されたものが生成されます。デバッグファイルも全て結合されます。

ビルドのエラーは5.5.3「ロード可能プロジェクト/ファイルの推奨使用法」または以下のセクションを参照してください。

PIC18 MCU (MPLAB C18)向けMPLAB Cコンパイラ

このコンパイラはアプリケーションの起動コード(`c018x.o`)を生成します。リセットベクタ (アドレス0)で始まるこの起動コードはソフトウェア スタックの初期化、必要に応じてidataセクションの初期化、`main()`へのジャンプを含みます。この起動コードがアプリケーション内に残っていると、ブートローダ コードリセットと競合が生じ、データ競合に関するリンカエラー メッセージが表示されます。

解決策の1つは、0以外のアドレスで始まるように起動コードを編集する事です。

MPLAB XC8 – PIC18 MCUの例

以下のMicrochip社ウェビナーでは、PIC18 MCU向けにブートローダとアプリケーション コードを組み合わせる方法を、MPLAB XC8とMPLAB X IDEを使って詳しく説明しています。
Linking PIC18 Bootloaders and Applications

MPLAB XC16ブートローダの例

サンプル ブートローダについては、『MPLAB XC16 Cコンパイラ ユーザガイド』(DS50002071)またはヘルプファイルを参照してください。

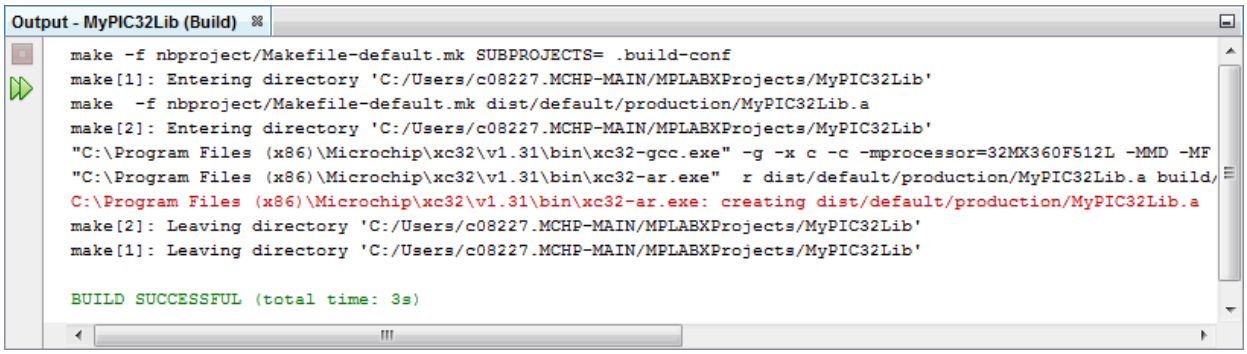
MPLAB XC32ブートローダの例

Microchip社ウェブサイトで『PIC32 Bootloader Application Note』(DS01388)を参照してください。

5.7 ライブラリ プロジェクト

HEXファイルではなくライブラリ ファイル(コンパイラによって*.libまたは*.a)を生成するプロジェクトを作成します(赤字部分参照)。このプロジェクトはmain()関数を持たないため、単独で実行する事はできません。作成したライブラリは、main()関数を持つ別のプロジェクトで使います。

図5-12 ライブラリ プロジェクトの例



[New Project]ウィザードの開き方は4.1.2「[New Project]ウィザードの起動」を参照してください。

[New Project]ウィザードが起動し、プロジェクトの新規作成を支援します。**[Next]**をクリックすると次のステップに進みます。

Step 1. Choose Project: 「Microchip Embedded」カテゴリを選択し、プロジェクト タイプ「Library Project」からプロジェクトを選択します。

Step 2. Select Device: アプリケーションで使う予定のデバイスを[Device]ドロップダウン リストから選択します。選択肢を絞り込むために、最初にデバイスファミリを選択します。

Step 3. Select Header: このステップは、選択したデバイスでヘッダが利用可能である場合に表示されます。デバッグ用にヘッダが必要かどうか、または、使うデバイスがデバッグ回路を内蔵しているかどうかは、以下のどちらかの文書で確認できます。

- プロセッサ拡張パックとデバッグヘッダ
- エミュレーション拡張パックとエミュレーション ヘッダ

Step 4. Select Tool: アプリケーションの開発に使う開発ツールをリストから選択します。使うデバイスのサポートを判定する方法は4.1.6.1「プロジェクト ツールサポート」を参照してください。

Step 5. Select Compiler: アプリケーションの開発に使う言語ツール (コンパイラ)をリストから選択します。使うデバイスのサポートを判定する方法は4.1.6.1「プロジェクト ツールサポート」を参照してください。

Step 6. Select Project Name and Folder: 新規プロジェクトの名前と保存場所を選択します。ダイアログを使って保存場所を選択できます。完了したら、**[Finish]**をクリックします。

作成したプロジェクトが[Projects]ウィンドウ内に表示されます。

必要に応じてライブラリ プロジェクトをスタンドアロン (アプリケーション) プロジェクトに変更できます。詳細は4.10.1「プロジェクト コンフィグレーション タイプの変更」を参照してください。

5.8 Atmel Studio 7またはAtmel STARTプロジェクトのインポート

[New Project]または[Import]ウィザードを使って、MPLAB X IDE用のAtmel Studio 7プロジェクトまたはAtmel STARTエクスポート ファイルをインポートします。

ウィザードは以下の2通りの方法で開きます。

[Import Project]ウィザードを使う方法

[Import Atmel Studio]または[START Project]ウィザードを開くには、以下のどちらかを選択します。

- `File>Import>Atmel Studio Project`
- `File>Import>START MPLAB Project`

[New Project]ウィザードを使う方法

[New Project]ウィザードは、以下の方法で開きます。

- [Start Page]で、**[Learn & Discover]**または**[My MPLAB X IDE]**タブをクリックし、[Projects]セクションで[Create New]へのリンクを選択します。
- [File] > [New Project](または<Ctrl>+<Shift>+<N>)を選択します。
- ツールバーで[New Project]アイコンをクリックします。

[New Project]ウィザードが起動し、プロジェクトの新規作成を支援します。

Step 1.Choose Project: [Microchip Embedded]カテゴリを選択し、プロジェクト タイプから[Import Atmel Studio Project]または[Import Atmel START MPLAB Project]を選択します。**[Next]**をクリックします。

5.8.1 [Import Atmel Project] - Locate Project File

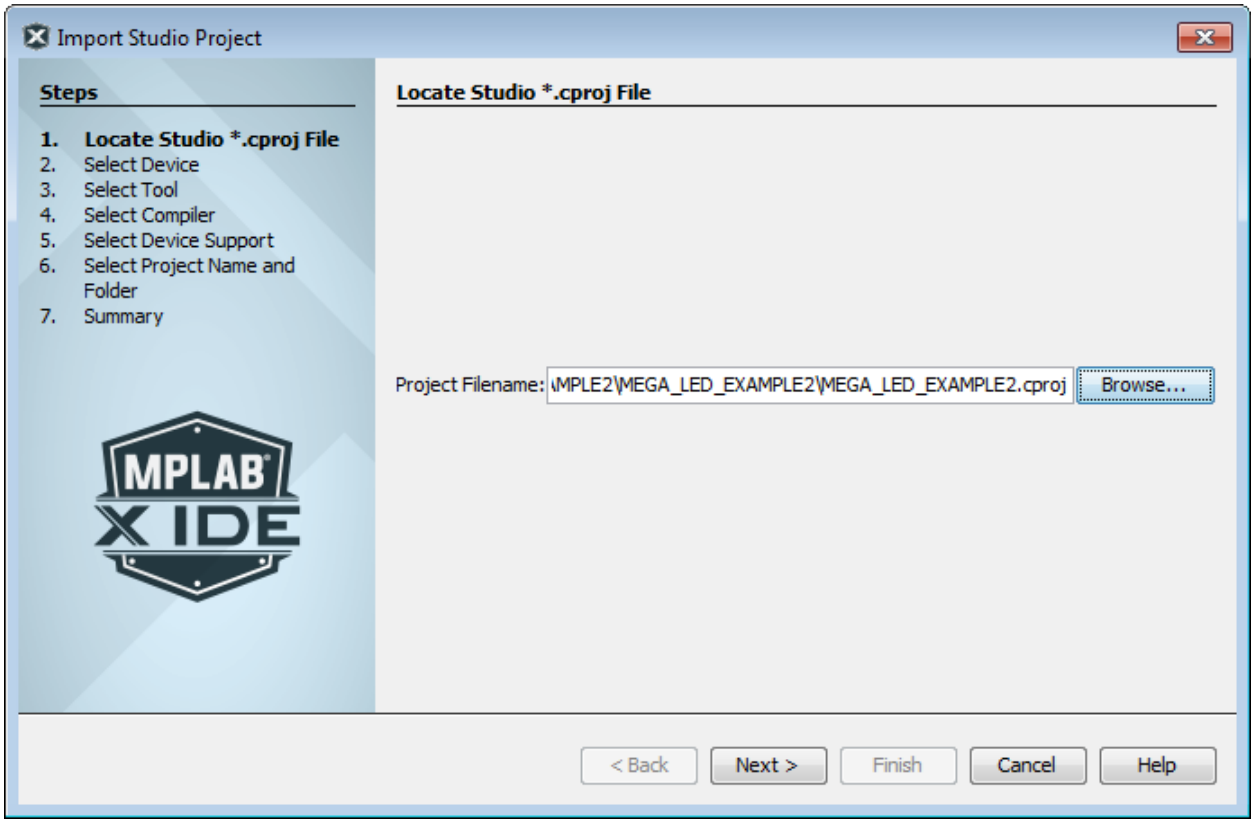
Step 1または2.Locate Atmel StudioまたはSTART Project File

要求したファイルを選択します。例えばWindows 7 OSの場合、プロジェクト ファイルは以下の場所に保存されています。

`C:\Users\Admin\Documents\Atmel Studio\7.0\`

Atmel Studioの場合、`*.cproj`または`*.cppproj`がプロジェクト ファイルです。Atmel STARTの場合、`*.atzip`がプロジェクト ファイルです。

図5-13　[Import Atmel Studio Project]

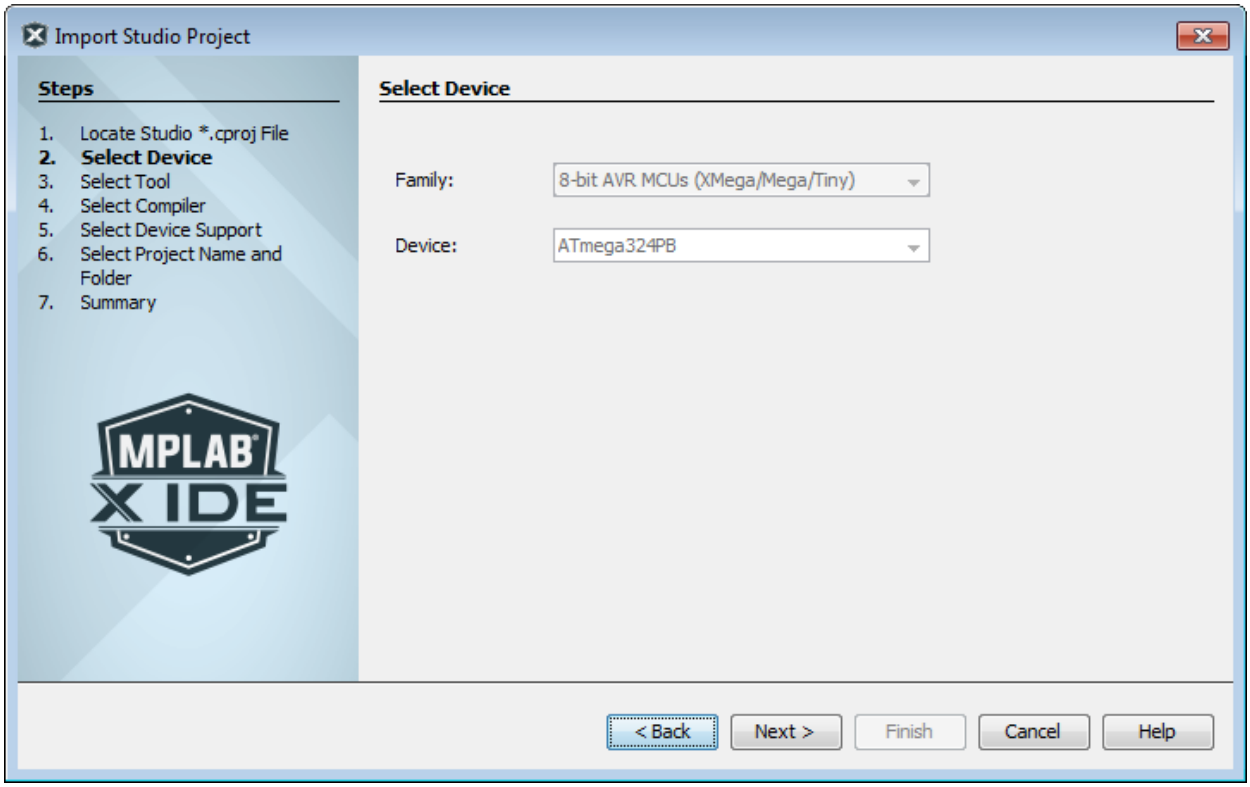


5.8.2　[Import Atmel Project] - Select Device

Step 2または3. Select Device

このウィンドウにはインポートしたプロジェクトのデバイス情報が表示されます。

図5-14　[Import Atmel Project] - Select Device



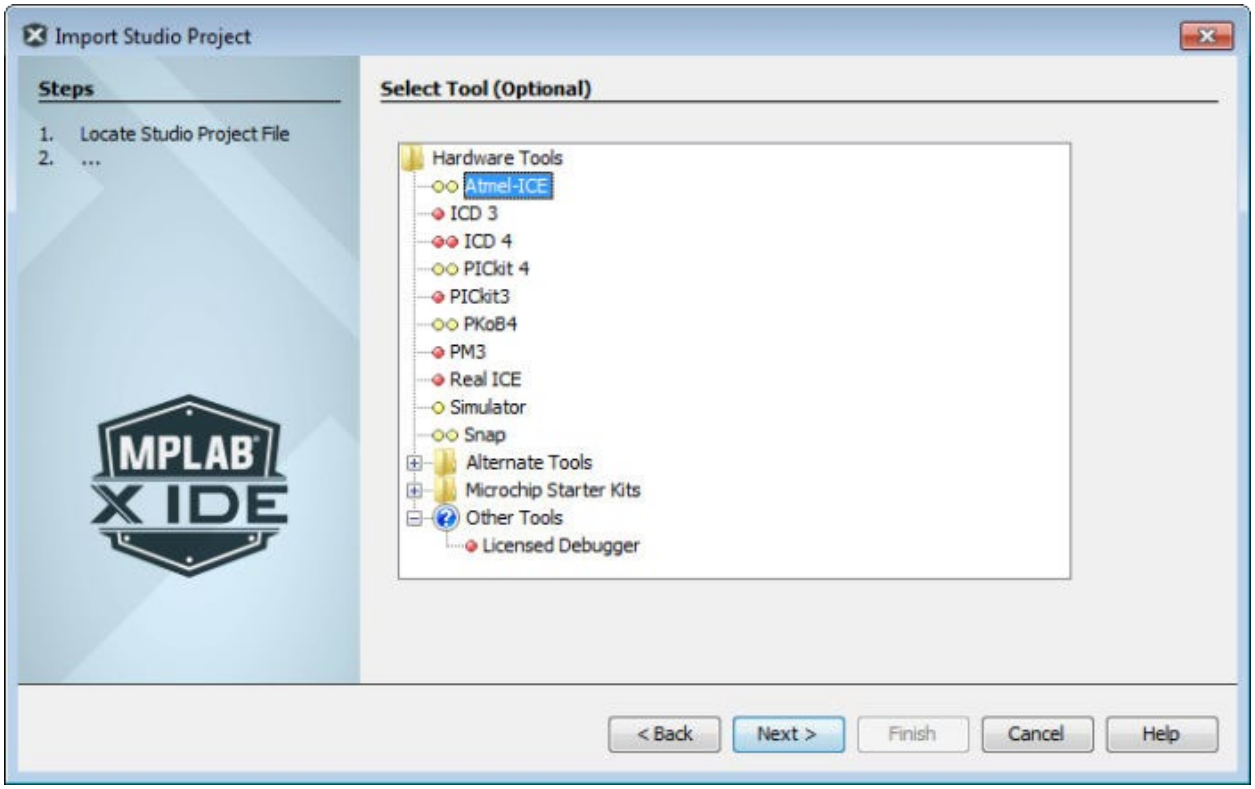
5.8.3　[Import Atmel Project] - Select Tool

Step 3または4. Select Tool

このウィンドウには、MPLAB X IDEで使えるAtmelプロジェクトのデバッグツールが表示されます。リストから別のツールを選択する事はできますが、プロジェクトが正しく動作する事を確認するため、インポートが完了するまで元のプロジェクト ツールから変えないでおきましょう。後から別のツールを選択できます。

デバイスに対するツールのサポートレベルが、ツール名の横のマークで色分け表示されます。この表示の詳細は「基本作業」の「新規プロジェクトの作成」を参照してください。

図5-15　[Import Atmel Project] - Select Tool



5.8.4　[Import Atmel Project] - Select Compiler

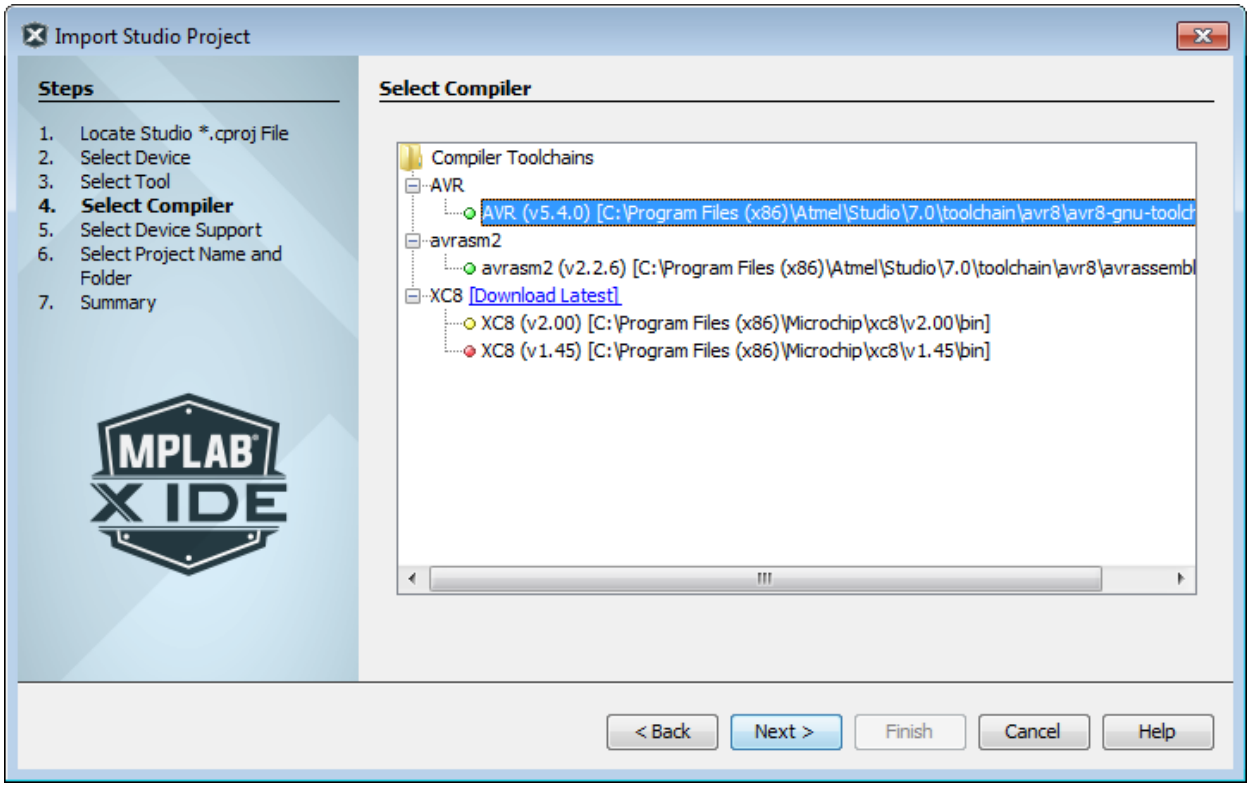
Step 4または5. Select Compiler

このウィンドウには、MPLAB X IDEで使えるAtmelプロジェクトのコンパイラが表示されます。リストから別のコンパイラを選択する事はできますが、プロジェクトが正しく動作する事を確認するため、インポートが完了するまで元のプロジェクト コンパイラから変えないでおきましょう。後から別のコンパイラを選択できます。

デバイスに対するツールのサポートレベルが、ツール名の横のマークで色分け表示されます。

互換性のあるコンパイラが表示されない場合、**[Cancel]**をクリックして「基本作業」の「新規プロジェクトの作成」に記載されている手順を実行します。

図5-16　[Import Atmel Project] - Select Compiler



5.8.5　[Import Atmel Project] - Select Device Support

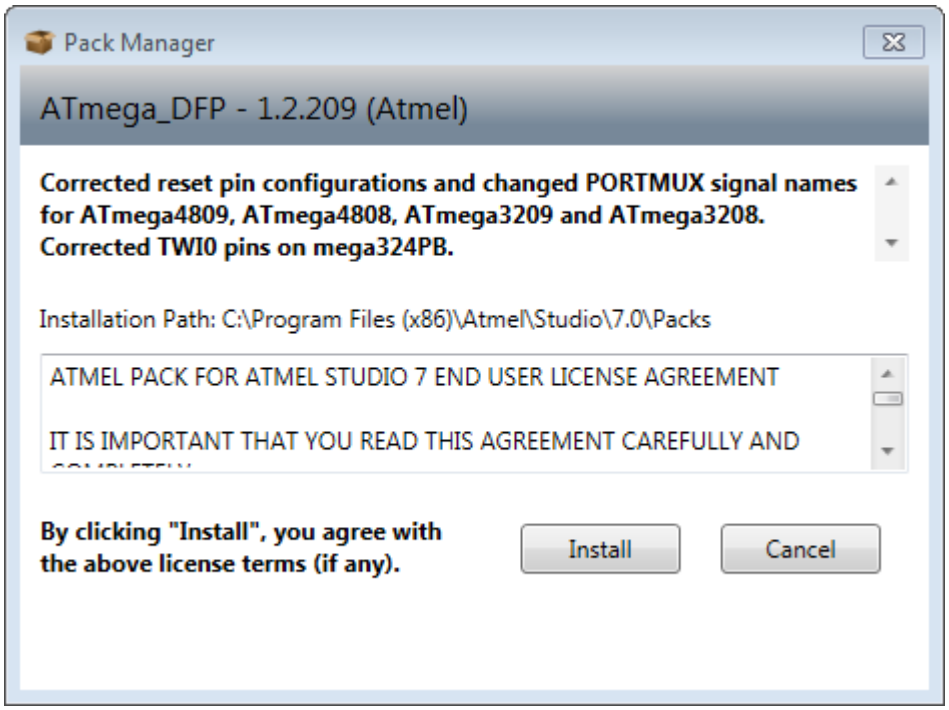
Step 5または6. Select Device Support

MPLAB X IDE v5.00以降は、バージョン管理されたデバイスパックでサポートされています。デバイスパック バージョンがMPLAB X IDEでサポートされていない場合:

1. [Override Default Device Support]にチェックを入れ、ヘルプアイコン(?)をクリックして表示される指示に従ってパックサポートをダウンロードします。パックをダウンロードすると、Atmel Studioにインストールされます(下図参照)。

図5-17 [Import Atmel Project] - Select Device Support

図5-18 デバイスパックのインストール



2. [Override Default Device Support]にチェックを入れると、Atmel Studioパックフォルダ内で正しいバージョンを検索できます。例:

   `C:\Program Files (x86)\Atmel\Studio\7.0\packs\atmel\ATmega_DFP\1.2.209`

3. [Override Default Device Support]のチェックを外すと、プロジェクト内で最新のMPLAB X IDEデバイスパック バージョンを使う事ができます。例えば、MPLAB X IDEプロジェクト パックは以下に保存されています。

```
C:\Program Files (x86)\Microchip\MPLABX\v5.xx\packs\Atmel\ATmega_DFP
```

5.8.6 [Import Atmel Project] - Select Project Name and Folder

Step 6または7. Select Project Name and Folder

両方のプロジェクトの保守性を保つため、既定値の名前とフォルダは変更しない事を推奨します。

Project Files Location:

Atmel Startの場合、ソースファイルはMPLAB X IDEプロジェクトにコピーされます。

Atmel Studioの場合、新規プロジェクトはソースファイルをプロジェクト フォルダにコピーしません。その代わりに、Atmelプロジェクト フォルダ内のファイル保存場所を参照します。

独立したMPLAB X IDEプロジェクトを作成するには、新規プロジェクトを作成してソースファイルをそのプロジェクトにコピーする必要があります。

Main Project:

[Set as main project]にチェックを入れて、このプロジェクトをインポート時のメイン プロジェクトに設定します。

Project Location:

Atmel Studioでは、[Use project location as the project folder]にチェックを入れずに、MPLAB X IDEプロジェクトとAtmelプロジェクトを一緒に保存する事を推奨します。WindowsでAtmelプロジェクトパスが長くなる場合(下図の警告参照)、MPLAB X IDEにインポートする前にプロジェクトのパスを変更します。

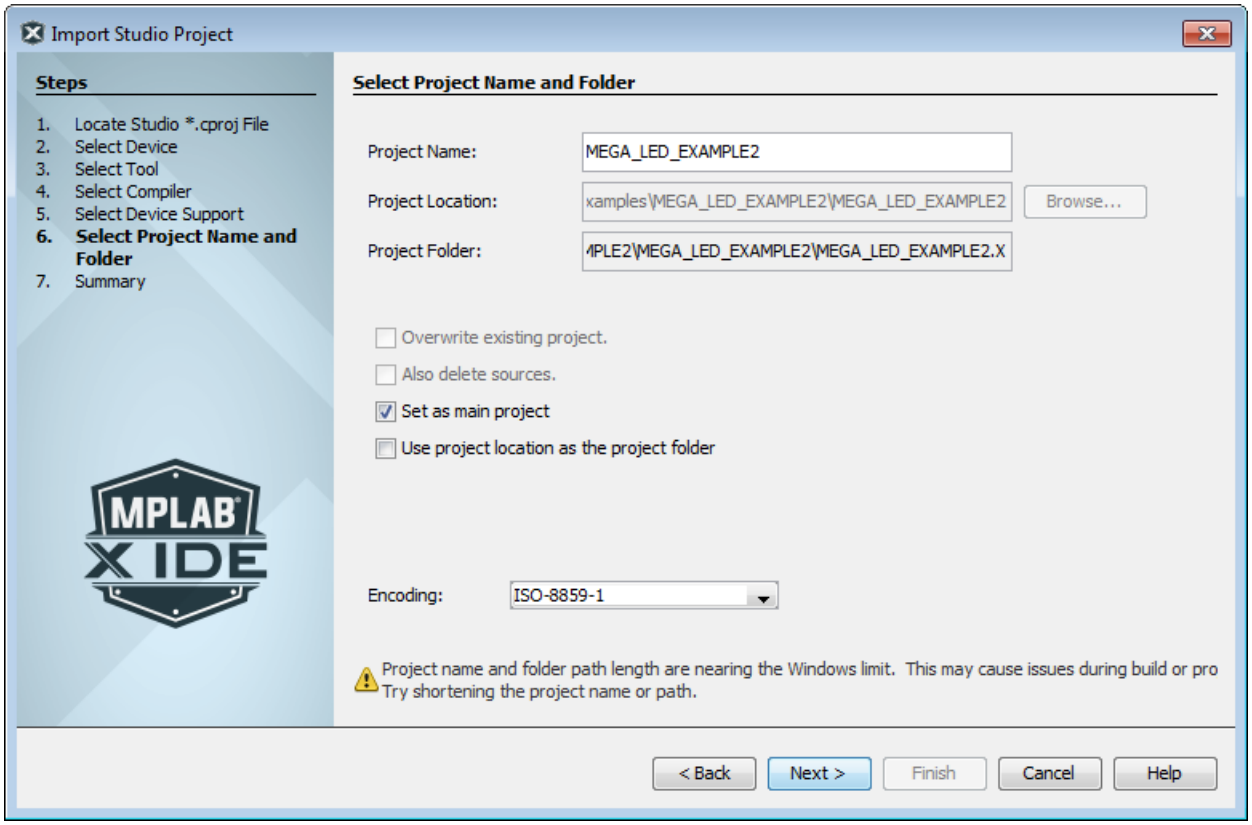
File Formatting/Encoding:

インポート元のプロジェクトで実際に使っているエンコードを選択する必要があります。例えばフォーマットが「950 (ANSI/OEM – Traditional Chinese Big5)」である場合、ドロップダウン リストから[Big5]を選択します。

Path Length (Window OS):

プロジェクトへのパスが長過ぎる場合、ビルド中の問題を避けるために短くする必要があります。

図5-19 [Import Atmel Project] – Select Project Name and Folder



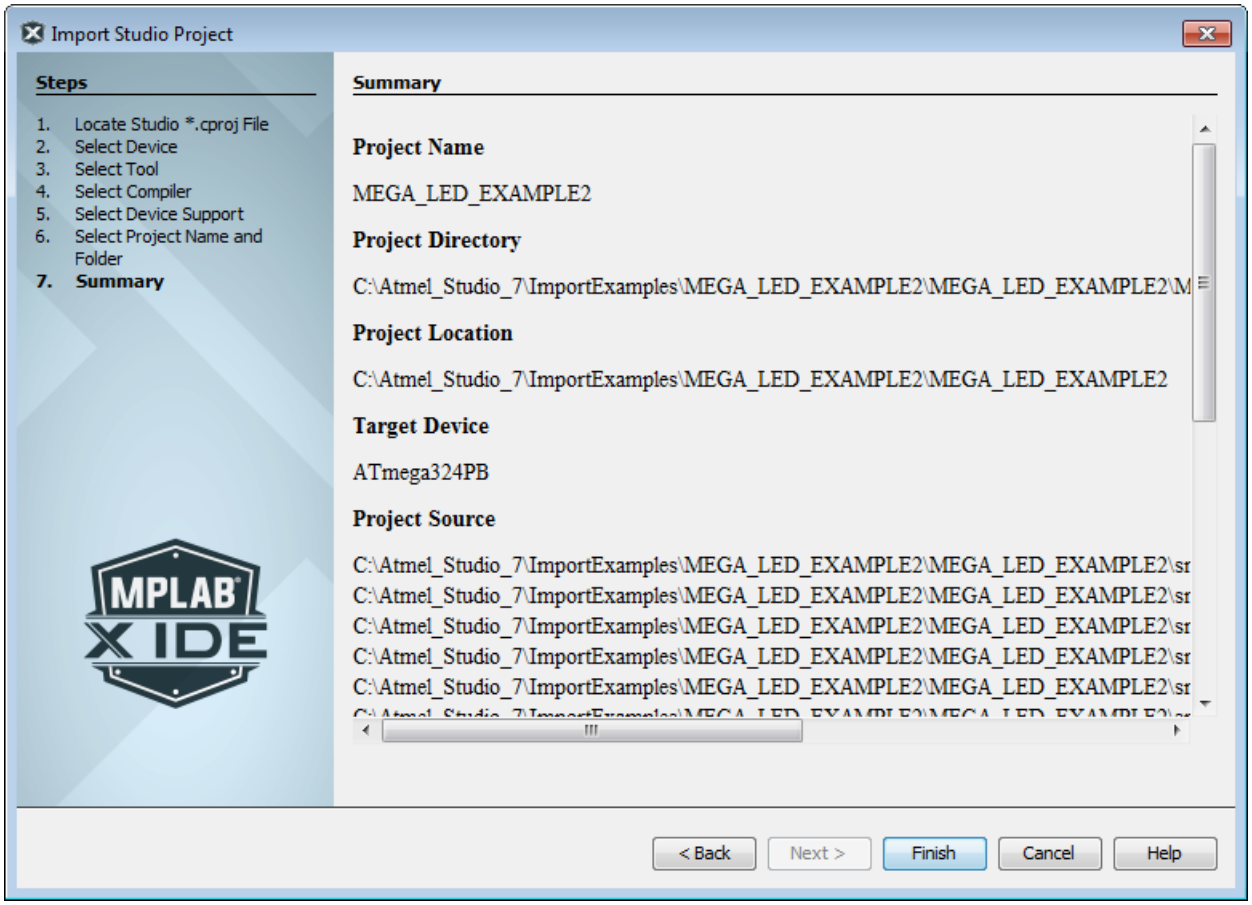
5.8.7 [Import Atmel Project] - Summary

Step 7または8. Summary

[Finish]をクリックする前に、設定した内容を確認します。不適切な設定が見つかった場合、**[Back]**ボタンを使って以前のステップに戻って修正します。

プロジェクト インポート中は、[Output]ウィンドウに警告等が表示されないか確認します。プロジェクトのビルドに失敗する場合、変更を加える必要があるでしょう。

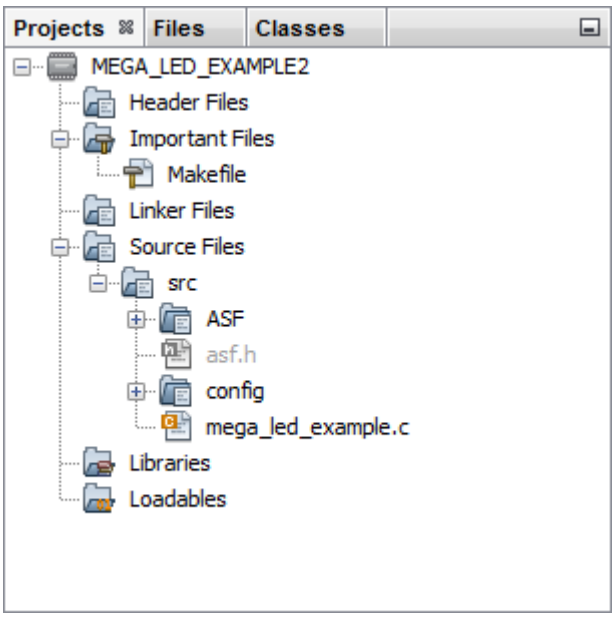
図5-20 [Import Atmel Project] – Summary



作成したプロジェクトが[Projects]ウィンドウ内に表示されます。

Note: 元のAtmelプロジェクトのソースファイルはリンクされます。コピーはされません。従って、Atmelまたは MPLAB X IDEプロジェクトでソースファイルに変更が加えられた場合、両方で通知されます。

図5-21　[Import Atmel Project] – [Projects]ウィンドウ



5.9　その他の組み込みプロジェクト

MPLAB X IDEでは、他の組み込みプロジェクトを選択して、そこからプロジェクトを作成できます。

1. [File] > [New Project]を選択します。
2. [Categories]の下の[Other Embedded]をクリックし、使用可能な組み込みプロジェクトのリストから選択します。
3. MPLAB X IDEプロジェクトの作成を継続します。

この機能は既存のファイルをMPLAB X IDEプロジェクトにインポートします。他の組み込みプロジェクト設定の変換やコード変換には未対応です。

MPLAB X IDEの使い方の詳細は以下を参照してください。

- チュートリアル
- 基本作業

使用可能なコンパイラの詳細は以下を参照してください。

https://www.microchip.com/mplab/compilers

5.10　サンプル プロジェクト

サンプル プロジェクトを作成すると、Microchip社のデバイス、ツール、MPLAB X IDEの習得に役立ちます。

1. [File] > [New Project]を選択します。
2. [Categories]の下の[Samples]>[Microchip Embedded]をクリックし、使用可能な組み込みプロジェクト (デモボードのライトを点滅させるもの)またはテンプレート プロジェクトのリストから選択します。詳細は説明を参照してください。

デモボードの製品番号は以下の通りです。

- Explorer16デモボード: DM240001
- PICDEM 2 Plus: DM163022-1

5.11 他のタイプのファイルの使い方

[File] > [New File]を選択すると、多くのタイプのファイルが提示されます。ここまでは、Microchip社のコンパイラファイルの使い方を説明してきました。しかし、使うプロジェクト言語ツールや特定のファイルを作成する必要性に応じて、他のタイプのファイルを選択する事もできます。

表5-4 ファイルタイプ

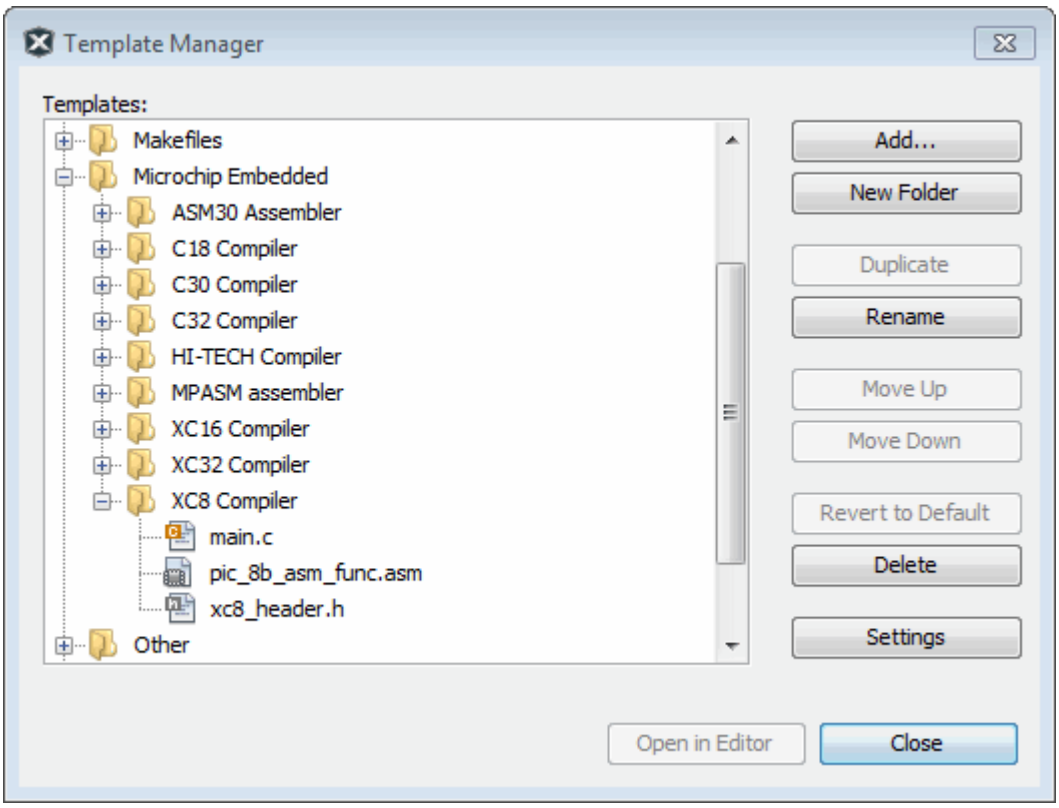
カテゴリ	ファイルタイプ
Microchip Embedded	MPLAB X IDEがサポートする言語ツール用のファイルです。利用可能なファイルタイプは、コンパイラ フォルダを選択すると表示されます。
C	汎用Cファイル
C++	汎用C++ファイル
Assembler	汎用アセンブリ ファイル
Shell Scripts	シェルスクリプト ファイル: Bash、C、Korn等
Makefile	Makefileファイル
XML	XMLファイル
Other	HTML、JavaScript等の他のタイプのファイル 一覧に必要なファイルタイプが表示されない場合、[Empty File]を選択し、次のウィンドウで、拡張子と共にファイルに名前を付けます。

5.12 コード テンプレートの変更または作成

プロジェクトに追加するファイルを作成(4.9.2「ファイルの新規作成」参照)する際に、新規ファイル用のテンプレートを使います。このテンプレートを編集するには、[Tools] > [Templates]から[Open in Editor]を選択します。このダイアログの[Add]または[Duplicate]ボタンを使って、テンプレートを新規作成する事もできます。

[New Folder]ボタンを使うと、テンプレートを保存する新しいフォルダを作成できます。MPLAB X IDEは「Microchip Embedded」、「Shell Scripts」、「Makefiles」、「Other」以外の全てのフォルダを無視するため、ファイルまたはフォルダは、これらのフォルダの下に作成する必要がある事に注意してください。

図5-22　テンプレート マネージャ



テンプレート オプションは、[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択し、**[Editor]**ボタンをクリックし、**[Code Templates]**タブで設定できます(以下参照)。

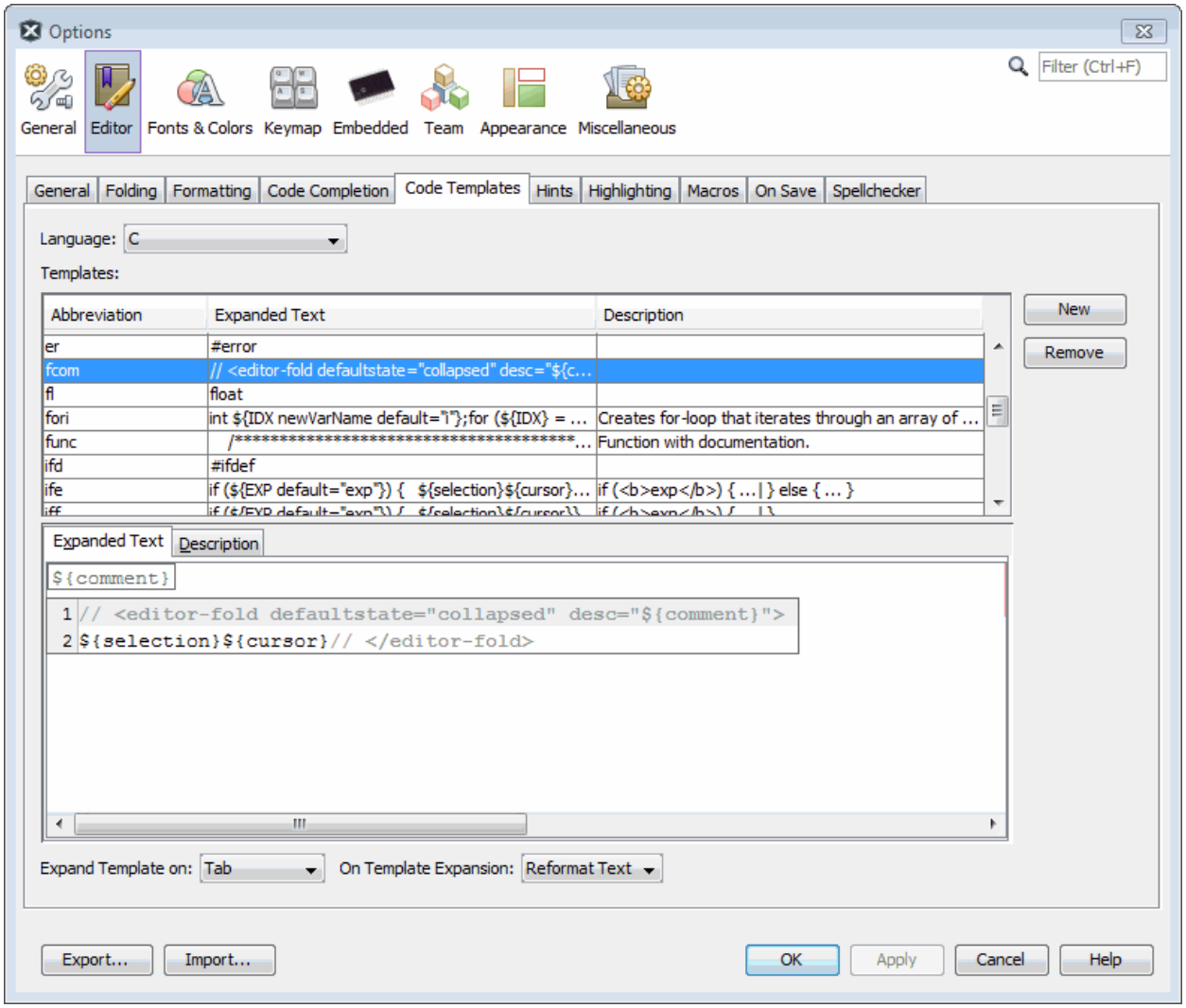
「fcom」オプションは、Cコード用のコメント オプションです。[Editor]ウィンドウに「fcom」と入力して<Tab>キーを押すと、以下のテキストが挿入されます。

```
// <editor-fold defaultstate="collapsed" desc="${comment}">
${selection}${cursor}// </editor-fold>
```

このオプションを使うと、上のテキストで囲まれたコードセクションを非表示(折り畳み)/表示(展開)できます。

[Code Template]タブで「fcom」を選択すると、[Expanded Text]セクションに「${comment}」が表示されます。このテキストの上にマウスカーソルを置くと、テキスト全体が表示されます。

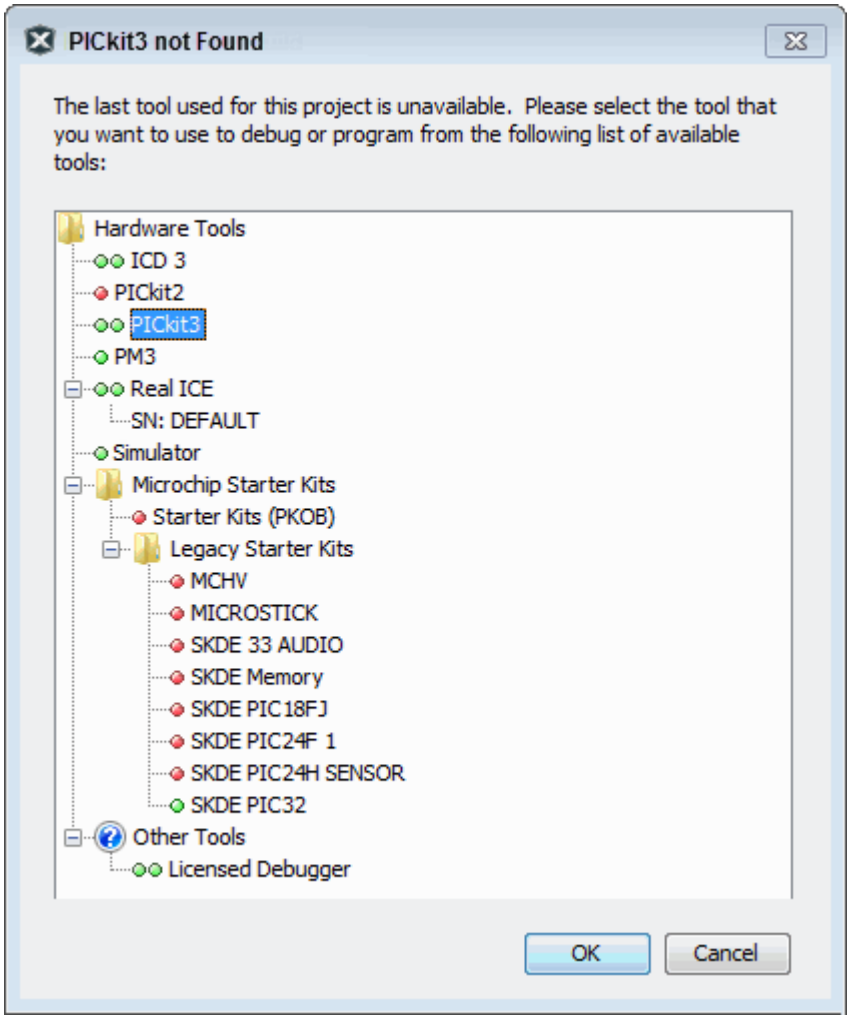
図5-23　テンプレート オプション



5.13　ハードウェア ツールまたは言語ツールの切り換え

現在接続しているものと異なるデバッグツールを選択しているプロジェクトを実行またはデバッグしようとした場合、そのプロジェクトに使う別のハードウェア ツールを選択できるダイアログが表示されます。

図5-24　[Switch Hardware Tool]ダイアログ



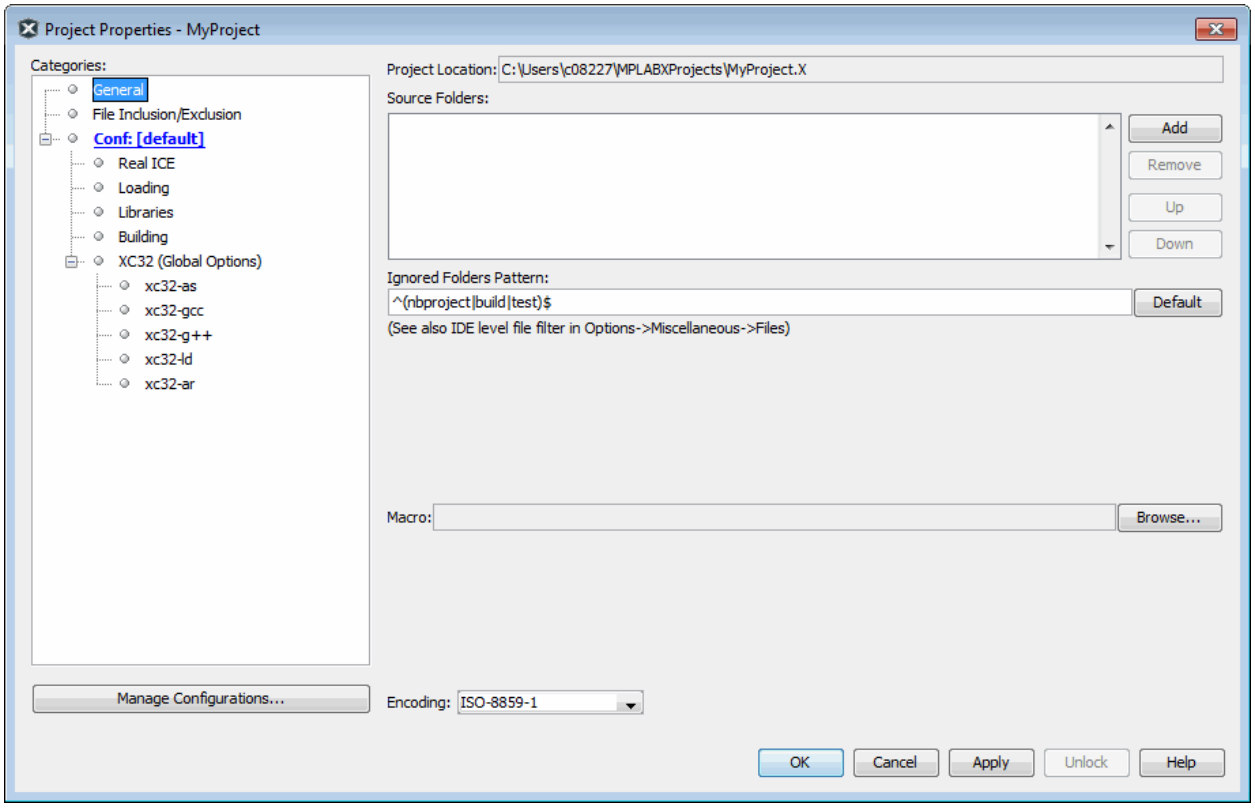
複数のハードウェア ツールを接続しておき、[Project Properties]ウィンドウでそれらのツールを切り換える事もできます(4.3「[Project Properties]ウィンドウを開く」参照)。

複数バージョンのコンパイラ ツールチェーン(言語ツール)を切り換える場合も、[Project Properties]ウィンドウを使います。

5.14　プロジェクト フォルダとエンコードの変更

プロジェクトを作成した際、プロジェクト フォルダとエンコードを指定しました。プロジェクト作成後は、[Project Properties]ウィンドウの[General]カテゴリを使ってプロジェクト フォルダを追加または無視し、プロジェクト エンコードを変更できます。

図5-25　[Project Properties]ウィンドウ – [General]カテゴリ



[Project Location]

現在のプロジェクトの保存場所を表示します。プロジェクトの保存場所の変更は8.9「プロジェクトの移動、コピー、リネーム」を参照してください。

[Source Folders]

プロジェクト ファイルを探す際にMPLAB X IDEが検索するフォルダを追加します。

Note:　プロジェクト フォルダの外にファイルを追加するとプロジェクトの移植性が損なわれる場合があります。

[Ignored Folders Pattern]

指定した正規表現パターンに従ってプロジェクト フォルダ内のフォルダを無視します。

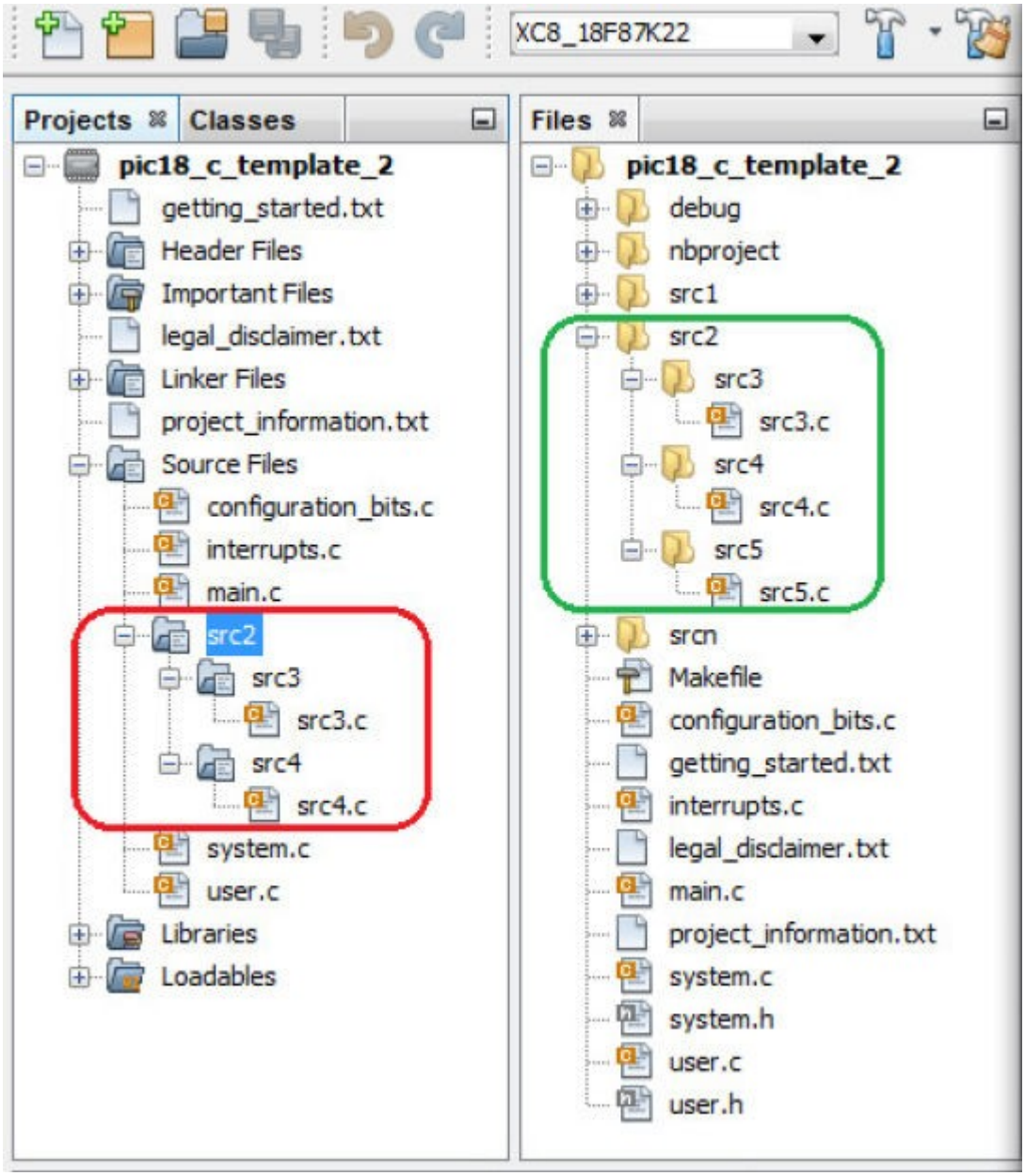
例: フォルダ`src5`を除外する

プロジェクトから`src5`フォルダとその中身を除外するには、[Ignored Folders Pattern]テキストボックスに以下の正規表現を入力します。

```
^(src5|nbproject|build|test)$
```

[Projects]ウィンドウを表示して`src5`フォルダが表示されていない事を確認します。[Files]ウィンドウを表示し、このフォルダが存在しているが除外されている事を確認します。

図5-26　プロジェクトからのフォルダの除外



プロジェクトがビルドされたら、[Output]ウィンドウの[Build]タブに`src5`フォルダまたは`scr5.c`ファイルが表示されていない事を確認します。

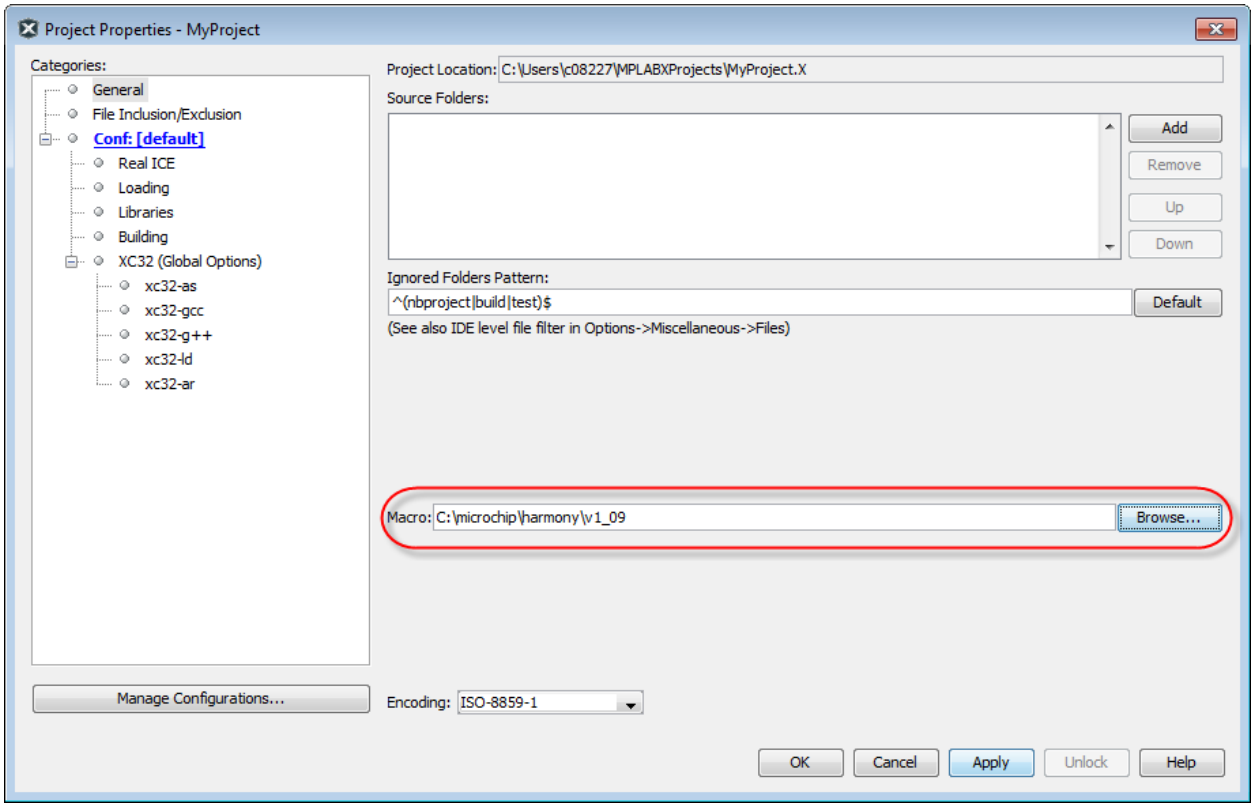
[Macro]

指定したソースファイルにリンクするために、[Add Existing Items]選択ダイアログで使うソースファイルへのパスを入力します。

例: MPLAB Harmonyファイル

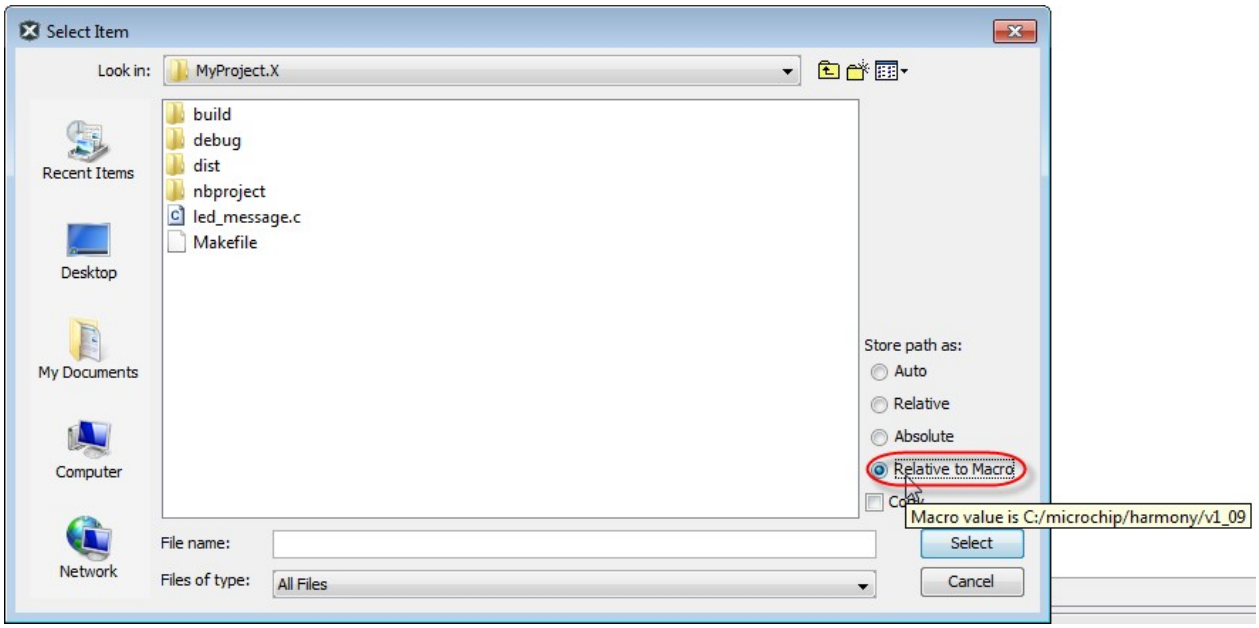
MPLAB Harmonyソースファイルへのパスを選択します。

図5-27　[Macro]用のパスの入力



[Projects]ウィンドウで[Source Files]フォルダを右クリックし、[Add Existing Items]のオプションの1つを選択します。このダイアログにはマクロからファイルを選択するオプションがあります。[Relative to Macro]の上にマウスカーソルを置くと、マクロ値が表示されます。

図5-28　マクロからファイルを選択する事を選択する



[Encoding]

プロジェクト エンコードを変更します。ここではコード構文の色分けを指定します。
これは、[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])、**[Fonts and Colors]**ボタン、**[Syntax]**タブで編集できます。

5.15 ストップウォッチの使用

ストップウォッチを使うと、2つのブレークポイント間の時間を調べる事ができます。

ストップウォッチの使い方

1. ストップウォッチを開始するブレークポイントを追加します。
2. ストップウォッチを停止するブレークポイントを追加します。
3. [Window] > [Debugging] > [Stopwatch]を選択し、ウィンドウの左にある**[Properties]**アイコンをクリックし、開始および停止ブレークポイントを選択します。
4. 再度プログラムをデバッグすると、ストップウォッチの結果が表示されます。

図5-29　ストップウォッチの設定

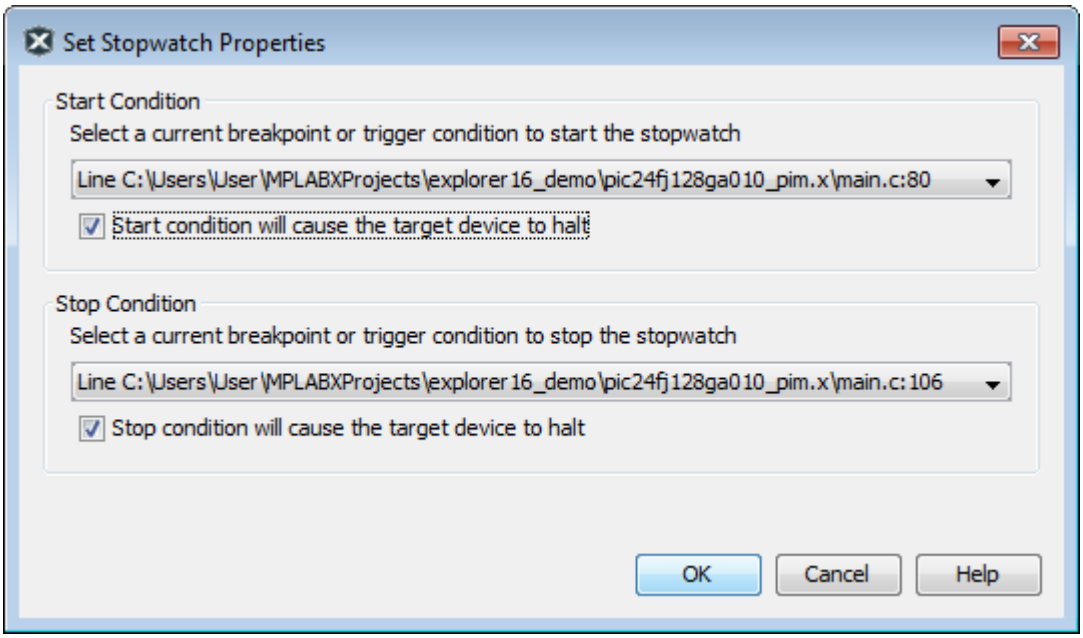
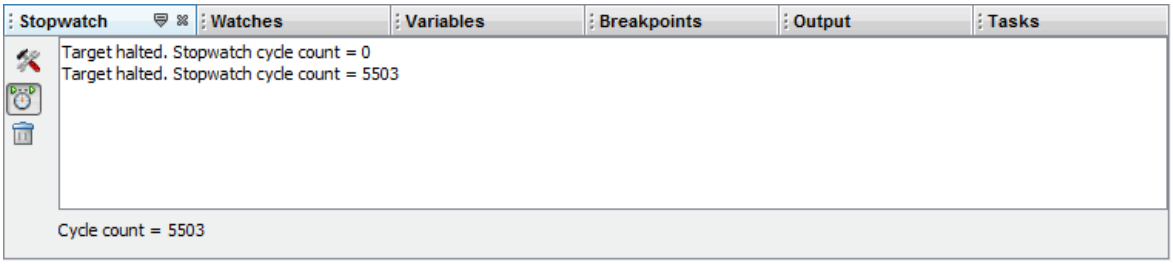


図5-30　[Stopwatch]ウィンドウの内容



[Stopwatch]ウィンドウの左側には以下のアイコンが表示されます。

表5-5　ストップウォッチ アイコン

アイコン	アイコンテキスト	説明
	Properties	ストップウォッチのプロパティを設定します。現在のブレークポイントまたはトリガの1つを始点、もう1つを終点として選択します。
	Reset Stopwatch on Run	実行時にストップウォッチの時間を0にリセットします。
	Clear History	[Stopwatch]ウィンドウをクリアします。

(続き)

アイコン	アイコンテキスト	説明
	Clear Stopwatch	(シミュレータのみ)デバイスリセット後にストップウォッチをリセットします。

5.16 [Disassembly]ウィンドウの表示

このウィンドウは逆アセンブルコードを表示します。このウィンドウは、[Window] > [Debugging] > [Output] > [Disassembly Listing File]と選択して開きます。ウィンドウの内容を表示するには、デバッグ セッションで一時停止する必要があります([Debug] > [Debug Project])。

この情報は、リンカが生成したリスティング ファイルにも含まれている場合があります。このファイルは[File] > [Open File]を選択し、
ProjectName.lstファイルを選択して開きます。

逆アセンブル ファイル全体を素早く表示するには、[Disassembly]ウィンドウで右クリックして「Disassembly Listing File」を選択します。

Note: [Disassembly]ウィンドウは各命令を逆アセンブルします。しかし、命令に関連したバンク切り換えの履歴は保持していません。従って、ウィンドウに表示されるSFR名はBank 0用です。

図5-31 変数をマウスオーバーした[Disassembly]ウィンドウ

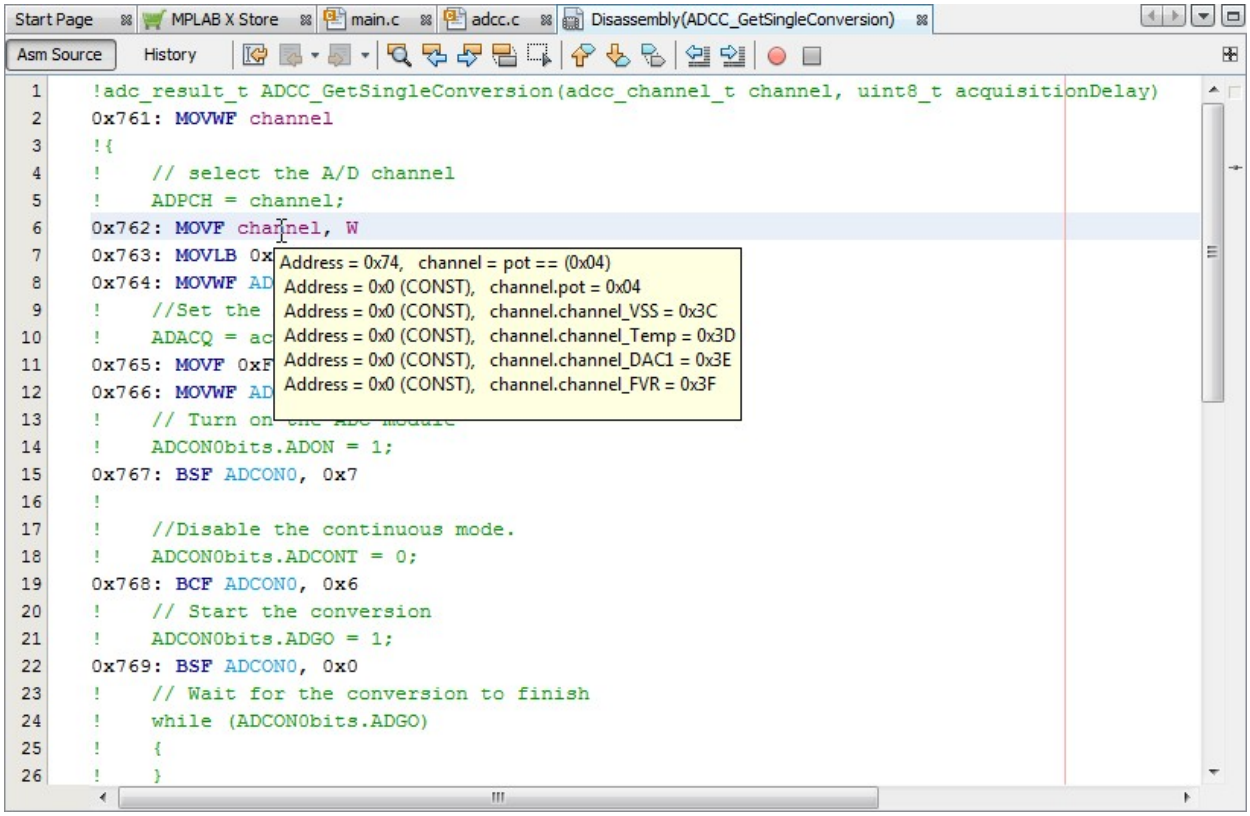


表5-6 [Disassembly]ウィンドウのコンテクスト メニュー

メニュー項目	説明
Step Instruction、Step Over、Run to Cursor	4.15「コードのステップ実行」を参照してください。
Cut、Copy、Paste	ウィンドウ内で選択したテキストの切り取り、コピー、貼り付け
Select in Projects	[Projects]ウィンドウを開いて、選択した内容を保存している文書を選択します。

(続き)	
メニュー項目	**説明**
Disassembly Listing File	逆アセンブル リスティング ファイルの内容を表示します。

5.17 コールスタックの表示

16/32ビットデバイスでは、[Call Stack]ウィンドウを使って、実行中のCコード内のCALLとGOTOを表示できます。このウィンドウはアセンブリコードには対応していません。コールスタックを使う場合、コード最適化はOFFにしておく事を推奨します。

[Call Stack]ウィンドウは、関数とその引数を、実行中のプログラムで呼び出された順番に一覧表示します。

コールスタックの表示方法

1. プログラムをデバッグ実行し、一時停止します。
2. [Window] > [Debugging] > [Call Stack]を選択して[Call Stack]ウィンドウを開きます。12.4「[Call Stack] ウィンドウ」も参照してください。コールスタックの詳細は、以下のNetBeansビデオをご覧ください。

https://www.youtube.com/watch?v=iS8OOUPmTUc

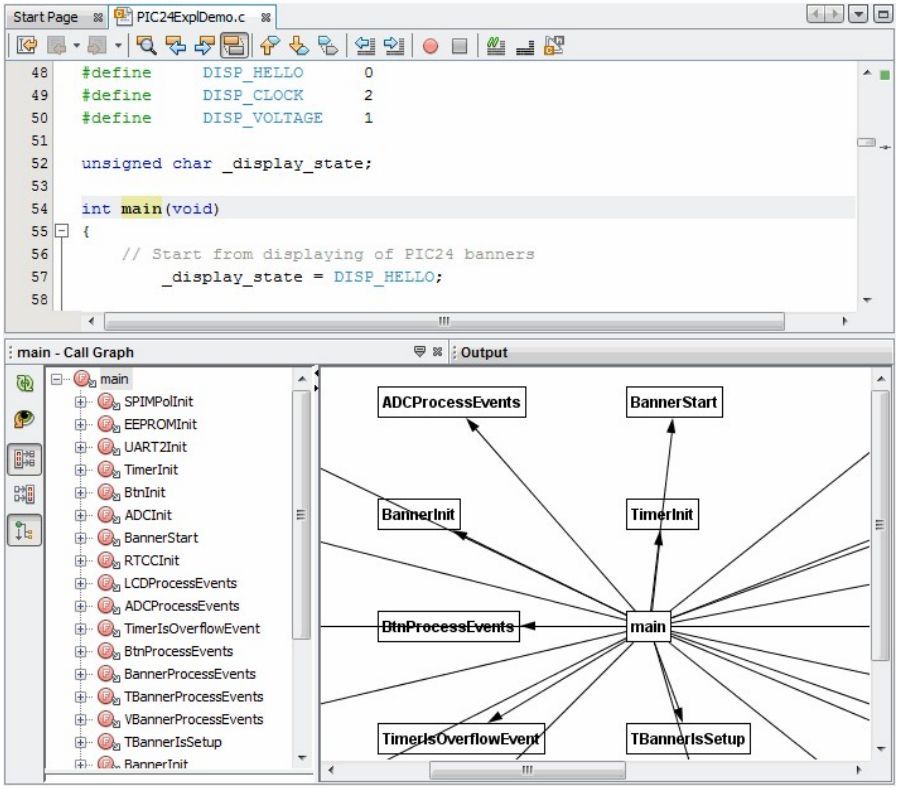
5.18 コールグラフの表示

[Call Graph]ウィンドウは、選択した関数からコールされている関数、または、選択した関数をコールしている関数をツリー構造で表示します。

コールグラフの表示方法

- 選択(強調表示)した関数を右クリックし、ドロップダウン メニューから[Show Call Graph]を選択します。

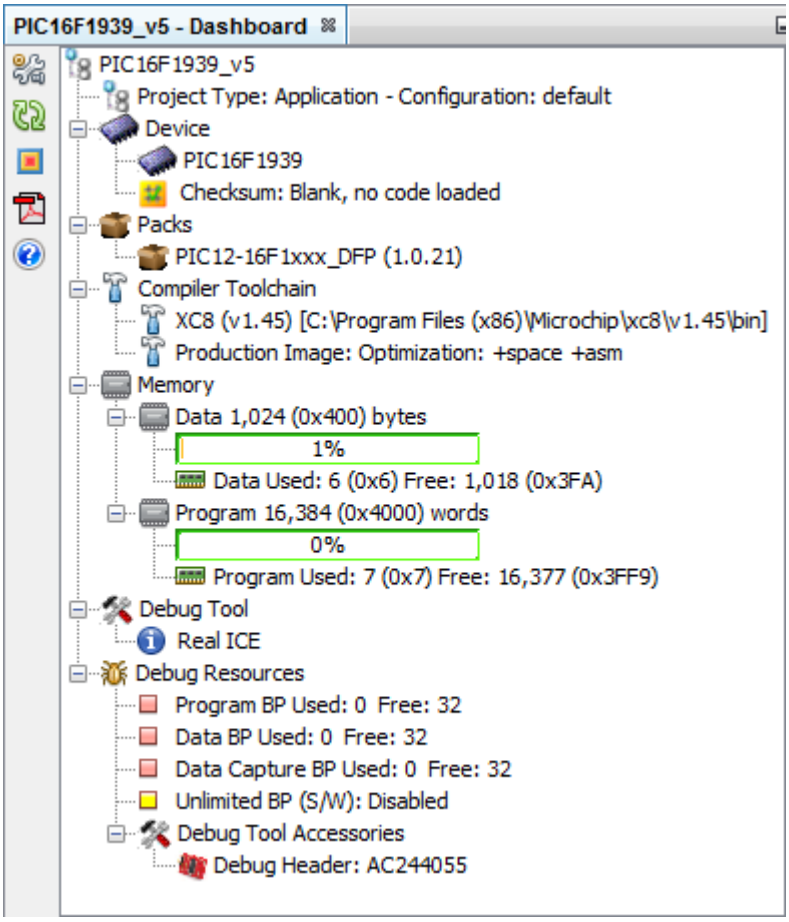
図5-32 main関数のコールグラフ



5.19 ダッシュボードの表示

[Window] > [Dashboard]を選択すると、プロジェクト情報を表示できます。

図5-33 ダッシュボードの表示



ダッシュボードは、以下のどちらかのプロジェクトを表示します。

- メイン プロジェクトが未選択の場合、[Projects]ウィンドウ内でアクティブなプロジェクト([Projects]ウィンドウ内でいずれかのプロジェクトをクリックすると、そのプロジェクトがアクティブになります)
- メイン プロジェクト([Projects]ウィンドウでいずれかのプロジェクトを右クリックし、[Set as Main Project]を選択します)
- 他のプロジェクトは、アクティブ/非アクティブに関係なく表示されません。

5.19.1 ダッシュボードのグループ

[Dashboard]ウィンドウのプロジェクト情報はグループごとに表示されています。

表5-7 ダッシュボードのグループ(続き)

グループ	定義と説明
Project Name	• **アクティブ/メイン プロジェクト**の名前 • **プロジェクト タイプ** (アプリケーションまたはライブラリ)と**プロジェクト コンフィグレーション**(既定値またはユーザ定義)
Device	• プロジェクト デバイス • 実行またはデバッグ実行中に生成された全ての**ステータスフラグ** • デバイスのメモリに読み込んだ内容の**チェックサム** チェックサムを表示するにはビルドする必要があります。6.14「チェックサム」を参照してください。

(続き)

グループ	定義と説明
 Packs	バージョン管理されたデバイス情報を保存するパック -.PICファイルやその他のファイル (デバイスによって異なります)
 Compiler Toolchain	• **プロジェクト ツールチェーン名**(例: XC8)、(ツールの**バージョン番号**)、[実行ファイルへの**パス**] • **イメージタイプ**(量産またはデバッグ)、ツールチェーンのライセンスタイプに対応する**最適化**レベル 最適化の詳細は言語ツールの文書を参照してください。 • **コンパイラ ライセンス***の情報: – license type: ワークステーションまたはネットワーク – license mode: FREE, STD, PRO – license seats: 利用できるシートの総数 – licenses available: 利用できるライセンスの数 – HPA – HPAが終了するまでの日数 – Expire – ライセンスが失効するまでの日数 – Press for status – [Output]ウィンドウにステータスを表示します。 * コンパイラ ライセンスを表示するにはMPLAB XC8 v1.34, MPLAB XC16 v1.25, MPLAB XC32 v1.40のいずれかを入手する必要があります。 コンパイラ ライセンスの詳細は以下を参照してください。 https://www.microchip.com/mplab/compilers
 Memory	• Usage Symbols disabled.のメッセージが出ている場合、メッセージをクリックして有効にします。 [Project Properties]ウィンドウの[Loading]カテゴリで、[Load symbols when programming or building for production (slows process)]にチェックを入れます。 • **メモリのタイプ**(DataまたはProject)とデバイスでの**利用可能容量** • **使用済みメモリ**は残りのメモリ容量の目安になります。 コンパイラはプログラム空間の使用率、コンフィグレーション ビット、ID位置、EEPROM(デバイスに内蔵されている場合)の詳細を記載したメモリ概要を出力します。 これらのメモリ空間の合計は、ワードサイズで考えると[Dashboard]の[Program Used]と一致します。メモリ容量が比較的少ないデバイスでは、マップファイルを確認する必要があります。
 Debug Tool	**ハードウェア デバッグツール** • デバッグツールの**名前**とシリアル番号 • デバッグツールの**接続**(既定値ではツールの接続は常時アクティブです)変更するには、[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択し、**[Embedded]**ボタンをクリックし、**[Generic Settings]**タブを開いて、[Maintain active connection to hardware tool]に移動します。 • **[Refresh Debug Tool Status]**ボタンをクリックすると、ハードウェア デバッグツールのファームウェア バージョンと現在の電圧レベルが表示されます。 **シミュレータ** • デバッグツールの**名前**(例: シミュレータ) • **シミュレートする周辺モジュールをクリックして**サポートされている周辺モジュールを表示します。

(続き)	
グループ	**定義と説明**
Debug Resources	**ハードウェア ブレークポイント** 現在**使用中**のブレークポイント数とプロジェクト デバイス向けに**使用可能**なブレークポイント数を示します。 **ソフトウェア ブレークポイント** プロジェクト デバイスでソフトウェア ブレークポイントが**サポートされている**かどうかを示します。 **デバッグツール アクセサリ** このプロジェクトで使っているデバッグツール アクセサリ (デバッグヘッダ等)の種類を示します。

5.19.2 [Dashboard]アイコン

[Dashboard]ウィンドウの左側のアイコンで、下表の各種機能にアクセスできます。

表5-8 サイドバーのアイコン

アイコン	機能
	プロジェクト プロパティ: [Project Properties]ウィンドウを表示します。
	デバッグツールのステータス表示を更新する: クリックすると、ハードウェア デバッグツールの詳細を表示します。
	ソフトウェア ブレークポイントをトグル(有効化/無効化)する: クリックするたびにソフトウェア ブレークポイントの有効/無効が切り換わります。この機能の状態はアイコンに表示されています。 • 中心が赤: 無効 • 中心が緑: 有効 • 中心が黒: サポートしていない – 使用中のデバイスまたはツールで利用できるブレークポイント数を超えている場合
	デバイス データシートを開く: Microchip社のウェブサイトからデバイスのデータシートを開きます。 https://www.microchip.com/ ローカルに保存したデータシートを開くか、ブラウザを開いてMicrochip社ウェブサイトでデータシートを検索できます。
	コンパイラのヘルプ: クリックすると、コンパイラ インストール フォルダの「docs」フォルダ内の文書の主索引(利用できる場合)が開きます。

5.20 プロジェクト レジスタの表示([I/O View])

[I/O View]ウィンドウ([Window] > [Debugging] > [IO View])で、現在のプロジェクトのターゲット デバイスのレジスタ概要を表示する事ができます。このウィンドウは設計時にクイック リファレンスとして機能し、プロジェクトがデバッグモードの時はレジスタ値を表示できます。

[I/O View]は上下ペインに分割され、上のペインには周辺モジュール、下のペインには選択した周辺モジュールのレジスタが表示されます。

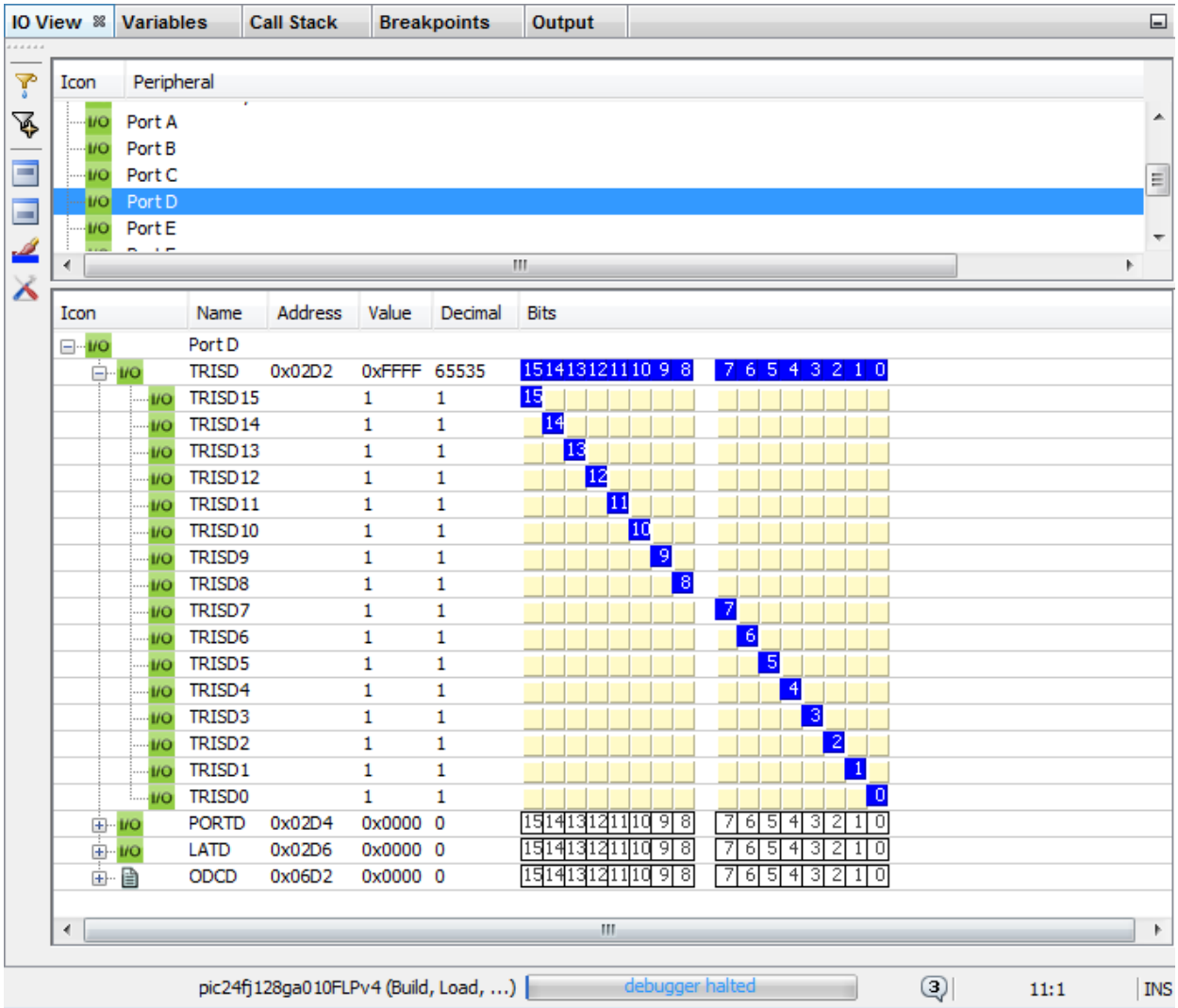
- 上のペインは、名前付きの編集値を持つレジスタをコンボで表示できます(例: Configコンボ)。
- 下のペインは、編集用のカスタムビット制御のあるレジスタワードと変更された色付きビットを表示します。

このウィンドウの詳細は12.8「[I/O View]ウィンドウ」を参照してください。

デバッグー時停止中の編集

デバッグモードを一時停止すると、ウィンドウ内で値を編集できます。

図5-34　デバッグ一時停止中の[I/O View]



ビットをクリックすると値がトグルします。レジスタ内の各ビットは別の列に表示されます。セットされたビットは既定値で暗い色になりますが、クリアされたビットは色がなくなります(既定値は白)。

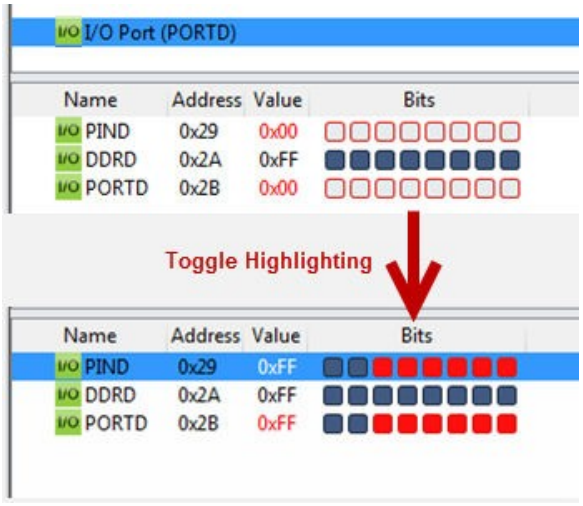
値フィールドをクリックし新しい値を書き込む事で、任意の値を変更できます。

一部の値とビットは読み出し専用のため変更できません。ビットによっては書き込み専用です。詳細は各デバイスの文書を参照してください。ビットまたは値がセットされるとデバイスから直ちに読み戻され、[I/O View]にはデバイスからの実際の値が表示されます。新しい値をセットしても[I/O View]が更新されない場合、そのレジスタは書き込み専用またはアクセス不可である可能性があります。

色の変更

前回の表示以降にレジスタが変更された場合、値とビットは赤で表示されます。前回以降にビットがセットされた場合、赤の塗りつぶしで表示されます。クリアされた場合、赤の枠線で表示されます。この機能はツールバーでトグルできます。

図5-35　[I/O View] - 変更されたビット表示のトグル



5.21　コードの改善

リファクタリングおよび/またはプロファイリングを使ってコードを改善します。

コードのリファクタリングは、コードの機能を変更せずにコードを単純化します。現在、Cコードでは以下を行えます。

- 複数ファイルを通して、関数の使用箇所を検出する
- 複数ファイルを通して、関数とパラメータの名前変更する

詳細は7.6「Cコードのリファクタリング」を参照してください。

コードのプロファイリングは、プログラム実行中にCPU使用率、メモリ使用率、スレッド使用率を常時監視するものです。プロファイリング ツールは、Cプロジェクト実行時に常時動作します。

NetBeans IDEメニューの機能(MPLAB X IDEでは通常使わないまたは表示されない機能)は、[Tools] > [Plugins] > [Installed] > [MPLAB Hidden Menu Filter]を選択し、その項目を選択して[Deactivate]をクリックすると表示できます。

5.22　ローカルファイル履歴によるソースコードの管理

MPLAB X IDEは、NetBeansプラットフォームの特長であるローカルファイル履歴機能を内蔵しています。この機能は、ローカル プロジェクトおよびファイル向けの、従来のバージョン管理システムのようなバージョン管理機能です。これにはローカルDIFFおよびファイル復元ツールが含まれます。

ファイルのローカル履歴を表示させるには、以下のどちらかを行います。

- [Editor]ウィンドウで**[History]**ボタンをクリックする
- [Project]または[File]ウィンドウでファイルを右クリックし、[History] > [Show History]を選択する

そのファイルの全変更履歴が表示されます。

図5-36　[Editor]ウィンドウ内のローカル履歴

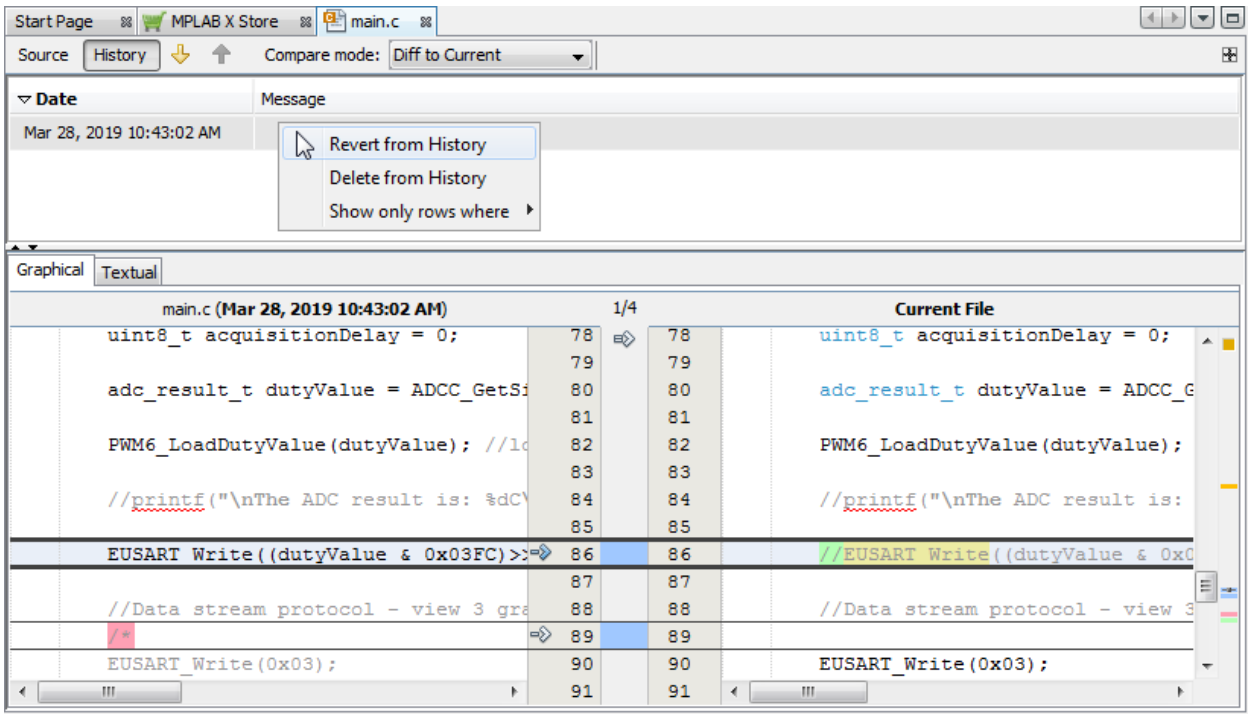


表5-9　ローカル履歴のコンテクスト メニュー

メニュー項目	概要
Revert from History	前のバージョンに戻します。[Revert to]ダイアログが開き、そのファイルの全変更履歴が表示されます。いずれかのバージョンを選択し、**[OK]**をクリックすると、そのバージョンが復元されます。
Delete from History	変更した項目を履歴から削除します。 履歴から削除した項目を元に戻すには、[Project]または[File]ウィンドウでファイルを右クリックし、[History] > [Revert Deleted]を選択します。
Show only rows where	[Date]列の下で右クリックして日付フィルタ(Date == またはDate < >)を選択します。 [Message]列の下で右クリックしてメッセージ フィルタ(Message == またはMessage < >)を選択します。

5.23　リビジョン管理システムによるソースコードの管理

多数のファイルを含む複雑なアプリケーションをチームで開発する場合、ソースコードの変更管理が必要です。MPLAB X IDEはソースファイルのバージョン管理機能を備えています。さらに、外部のリビジョン (ソースまたはバージョン)管理プログラムもサポートしています。

各種のリビジョン管理システム(RCS)をMPLAB IX DEと組み合わせて使う事ができます。一度設定すると、リビジョン管理操作はコンテクスト メニューからアクセスできます。

現在サポートしているソース/バージョン管理システムはGit、Subversion、Mercurialです。その他のシステムはプラグイン(例: CVS)を介してサポートできる場合があります。

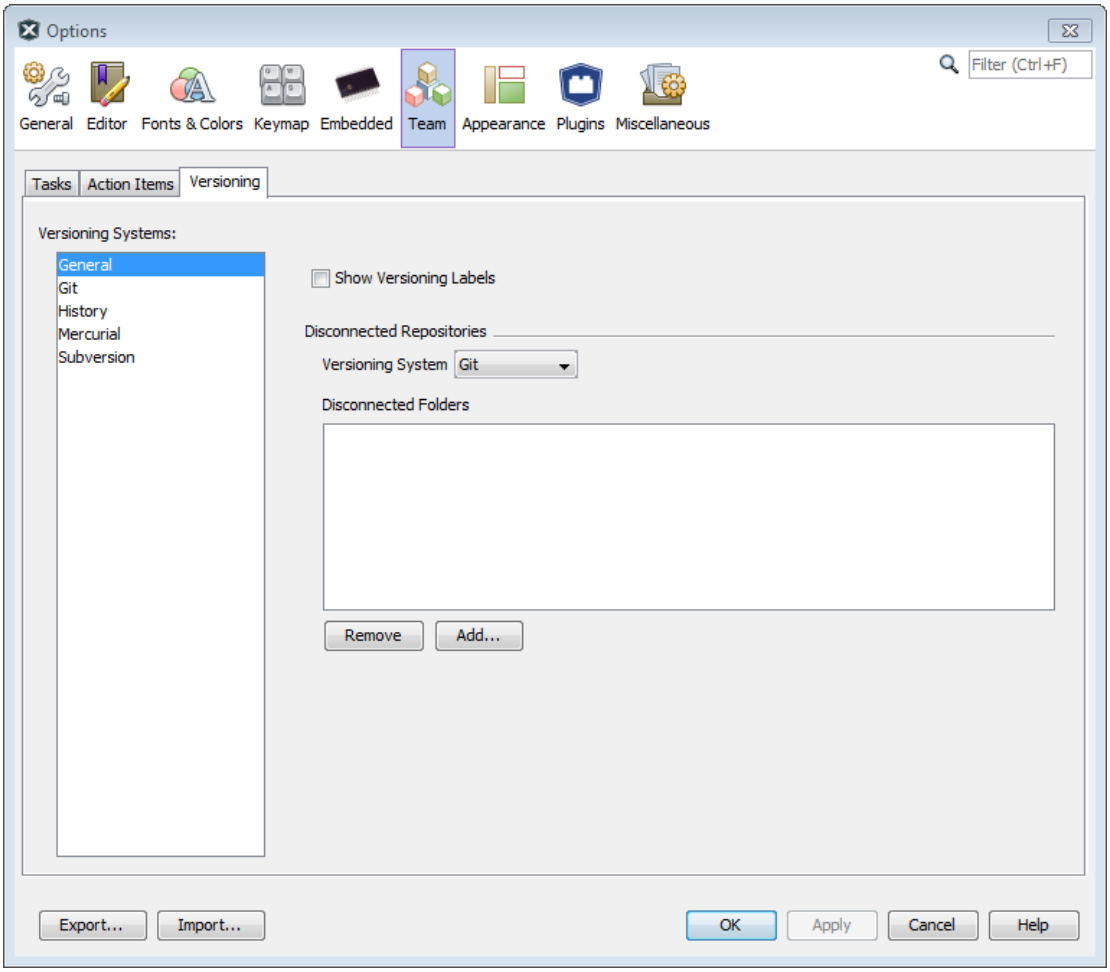
MPLAB X IDEプロジェクト用のリビジョン管理システムを使う際の注意事項は以下の通りです。

1. 全てのプロジェクト ファイルを同プロジェクト内に保存するようにします。プロジェクト外のファイルもレポジトリに含め、そのプロジェクトを検索できるようにします。
2. MPLAB X IDEでは当該プロジェクトのファイルのみコミットします。不要なファイルを含めると、プロジェクトが他のマシンでコンパイルされなくなる可能性があるため、これは重要です。「プロジェクト ファイルの保存」を参照してください。

3. [Merge Conflicts Resolver]ウィンドウを使ってチェックインする際の競合を解消しました。「リビジョン管理されたファイルの競合の解消」を参照してください。
4. 読み出し専用ファイルは、「読み出し専用ファイルを使ったプロジェクトのビルド」に従って確認します。ソース/バージョン管理の設定方法

1. [Team]メニューからバージョン管理プログラムのサブメニューを選択し、そのバージョン管理プログラムを設定します(下図参照)。
2. [Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択して、[Team]を選択し、[Versioning]タブでバージョン管理オプションを設定します。
3. [Window] > [Versioning]を選択して[Version Control]ウィンドウを開きます。

図5-37　バージョン管理のオプション



Gitの使い方

Gitの詳細は以下のリンクをご覧ください。

- Gitウェブサイト: http://git-scm.com
- Wikipedia: Git: https://en.wikipedia.org/wiki/Git
- Gitチュートリアル: https://git-scm.com/docs/gittutorial
- NetBeans IDEでのGitサポートの使用: https://netbeans.org/kb/docs/ide/git.html

Subversionの使い方

Subversionの詳細は以下のリンクをご覧ください。

- Apache™ Subversion®ウェブサイト: http://subversion.apache.org/
- Wikipedia: Apache™ Subversion®: http://en.wikipedia.org/wiki/Apache_Subversion
- Subversion®によるバージョン管理: http://svnbook.red-bean.com/

- NetBeans IDEでのSubversionサポートの使用: https://netbeans.org/kb/docs/ide/subversion.html

Mercurialの使い方

Mercurialの詳細は以下のリンクをご覧ください。

- Mercurialウェブサイト: http://mercurial.selenic.com/
- Wikipedia: Mercurial: http://en.wikipedia.org/wiki/Mercurial
- Hginit: Mercurial tutorial: http://hginit.com/
- NetBeans IDEでのMercurialサポートの使用: https://netbeans.org/kb/docs/ide/mercurial.html

CVSの使い方

MPALB X IDEでCVSを使うには、CVSプラグインをインストールする必要があります。5.26「プラグインツールの追加」を参照してください。

- CVS - Concurrent Versions System: http://www.nongnu.org/cvs/
- Wikipedia: Concurrent Versions System: http://en.wikipedia.org/wiki/Concurrent_Versions_System
- Source Forge: What is CVS?: http://sourceforge.net/apps/trac/sourceforge/wiki/CVS
- NetBeans IDEでのCVSサポートの使用: https://netbeans.org/kb/docs/ide/cvs.html

その他のリビジョン管理システムの使い方

その他のリビジョン管理システムはプラグインとして追加できます。5.26「プラグインツールの追加」を参照してください。

5.23.1 プロジェクト ファイルの保存

リポジトリに保存する必要があるプロジェクト ファイル

下表に、プロジェクト ファイルをバージョン管理リポジトリにコミットする必要があるかどうかを示します。

表5-10　リポジトリに保存するプロジェクト ファイル

フォルダまたはファイル	コミットの要否
プロジェクト フォルダ	
Makefile	✔
ソースファイル	✔
buildフォルダ	✘
distフォルダ	✘
nbprojectフォルダ	
configurations.xml	✔
project.properties	✔
project.xml	✔
Makefile-*	✘
Package-*	✘
privateフォルダ	✘

(続き)

フォルダまたはファイル	コミットの要否
緑のチェック: プロジェクトのイメージを生成する必要があります。 赤のX印: これらのフォルダ/ファイルは再生成されるため保存する必要はありません。	

プロジェクトの構成の詳細は「[Files]ウィンドウの表示」を参照してください。

5.23.2 リビジョン管理されたファイルの競合の解消

チェックインしようとするファイルと、リビジョン システムのリポジトリ内の他の人が変更した同じファイルとの間に競合が発生する場合があります。

これらの問題の解消方法は以下の例を参照してください。

例: `configurations.xml`の競合

`configurations.xml`ファイルの競合の解消方法

1. 競合は以下によって通知されます。
   1.1. 警告ダイアログ
   1.2. [Output]ウィンドウ
2. [Projects]ウィンドウの**[Files]**タブをクリックし、`nbproject`フォルダを展開します。`configurations.xml`という名前のファイルを見つけます(競合を示す赤いフォントで表示)。
3. `configurations.xml`ファイルを右クリックして[RCS] > [Resolve Conflicts]を選択します(RCSは使用中のリビジョン管理システムの名前)。これにより[Merge Conflicts Resolver]ウィンドウが開きます。
4. [Merge Conflicts Resolver]ウィンドウを使って競合を解消します。
5. エラーが解決したらプロジェクトを閉じます。右クリックのコンテクスト メニューが動作しない場合、[File] > [Close Project]を使ってプロジェクトを閉じます。
6. 再度プロジェクトを開きます。競合は解消済みのはずです。

5.23.3 読み出し専用ファイルを使ったプロジェクトのビルド

ファイルがリポジトリにチェックイン済みであり、一部のファイルが読み出し専用のプロジェクトをビルドするには以下の手順で行います。

1. 最小限のファイルをソース管理にコミットします(5.23.1「プロジェクト ファイルの保存」参照)。
2. ファイル`nbproject/configurations.xml`と`nbproject/project.xml`を書き込み可能にします。

5.24 コード開発とエラー追跡の連携

MPLAB X IDEは、チームサーバによるグループ間で連携したコード開発をサポートします。[Team]メニューを使うと、アカウントへのログイン、プロジェクトの作成またはロード、プロジェクトの共有、リソースの取得、チャットメッセージの送信、コンタクトリストの表示が可能です。問題追跡システムを使うとバグ追跡で連携できます。

サポートするメニュー項目は以下の通りです。

- [Window] > [Services] – [Services]ウィンドウは実行時リソースへのメイン エントリポイントです。IDEに登録されているサーバ、データベース、Webサービス等の重要な実行時リソースの論理ビューを表示します。詳細は[Help] > [Help Contents]で[Services]ウィンドウを検索してください。
- [Team] > [Find Task] – [Find Tasks]ウィンドウ内で、プロジェクト タスクと評価基準を選択し、**[Search]**をクリックします。
- [Team] > [Report Task] – [Report a New Task]ウィンドウ内で新しいプロジェクト タスクを選択し、問題の詳細を記述し、**[Submit]**をクリックします。

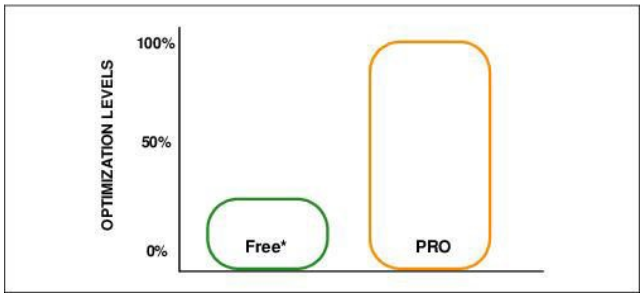
チーム プロジェクトと問題追跡の詳細は以下のNetBeansヘルプトピックを参照してください。
https://netbeans.org/kb/docs/ide/team-servers.html
これらツールの詳細は以下を参照してください。

- Git – http://git-scm.com
- Mercurial – http://mercurial.selenic.com/
- Subversion – http://subversion.apache.org/

5.25 MPLAB XCコンパイラのFreeライセンスとPROライセンスの比較

Microchip社ウェブサイトからMPLAB XC Cコンパイラをダウンロードすると、コンパイラはPROライセンスとして60日間実行され、その後Freeライセンスになります。最適化機能を継続して使う場合はPROライセンスを購入できます。

図5-38 各ライセンスの最適化レベル

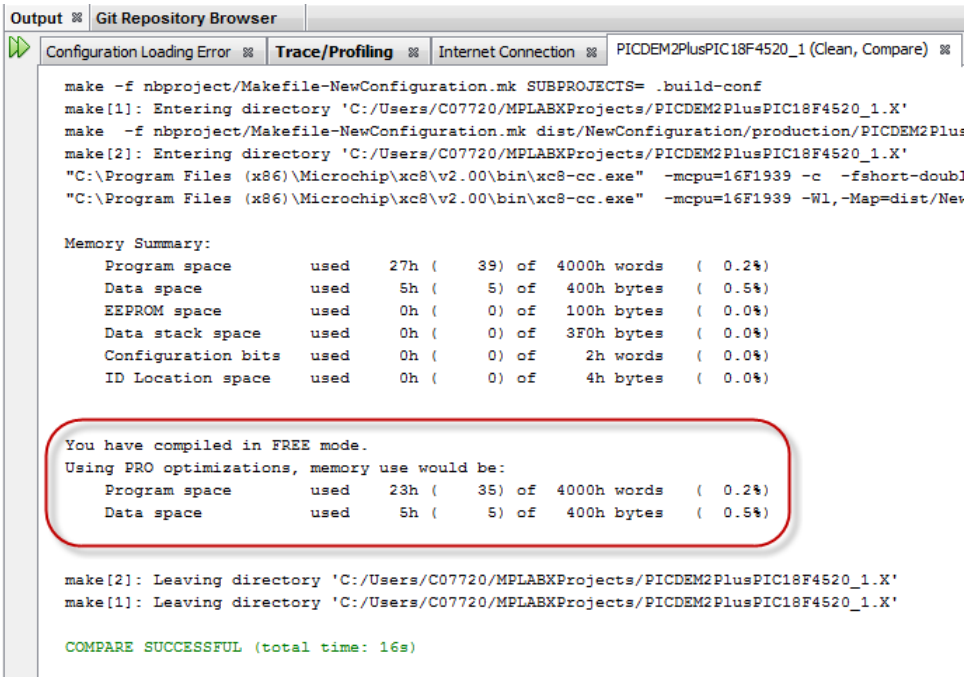


PROライセンスを購入すべきかどうか判断するには、MPLAB X IDEでアプリケーションの比較ツールを実行します。ビルドアイコンのオプション プルダウン メニューから[Build with PRO comparison]または[Clean and Build with PRO comparison]を選択します。以下に出力例を示します。

Note: コードが大き過ぎてFreeモードでビルドできない場合、この比較は機能しません。しかし、MPLAB XC8コンパイラの場合は--MAXIPICオプションを使って試行できます。

これらのオプションは選択された最適化レベルでプロジェクトをビルドしますが、機能しないPROビルドも作成して2つの結果を比較します。元のビルドは出力データとして表示されます。PROビルド部分は削除されます。

図5-39 PROの比較



5.26 プラグインツールの追加

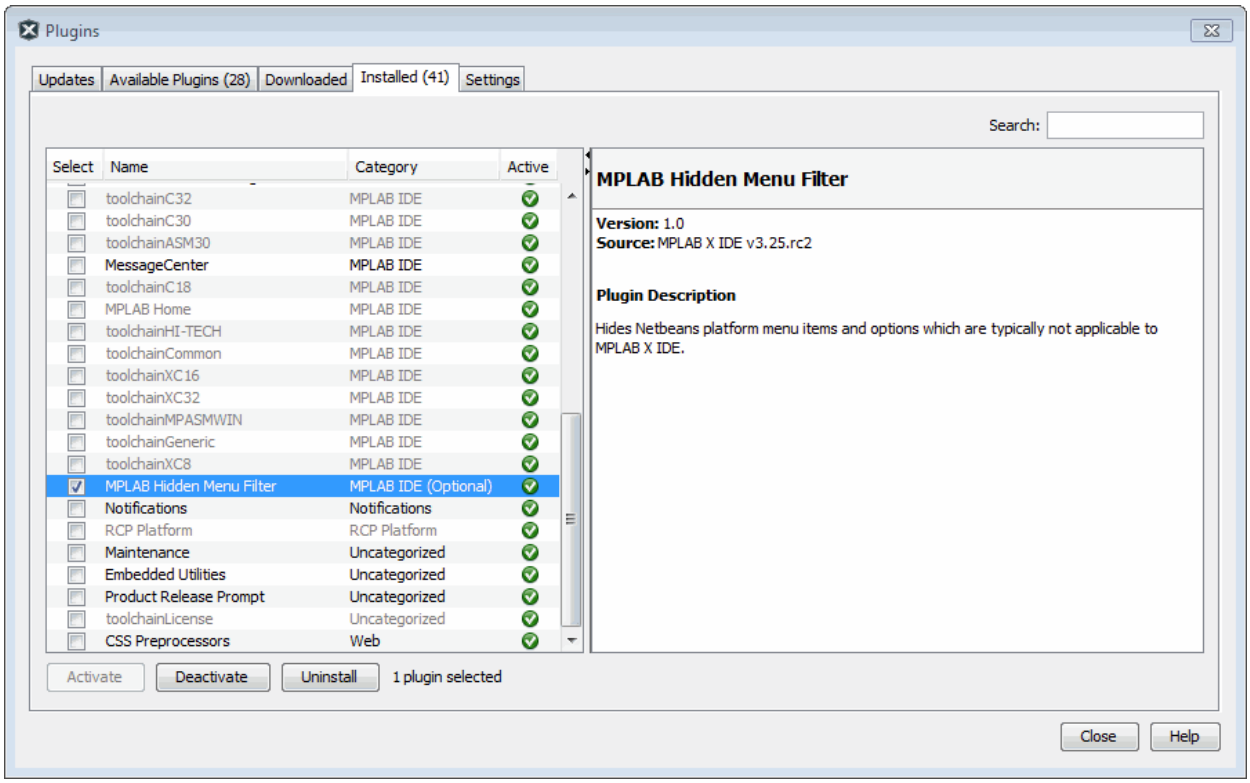
DMCI等MPLAB X IDEのプラグインツールは、IDEのプラグイン マネージャまたはEmbedded Code Sourceウェブサイトから追加できます。

インストールしたプラグインの表示

MPLAB X IDEには多数のプラグインがインストールされています。灰色表示でない項目は無効にするまたはアンインストールする事ができますが、[Category]に(Optional)と表示されていない場合は推奨しません。

インストールされているプラグインを表示するには、[Tools] > [Plugins]を選択し、**[Installed]**タブをクリックします。

図5-40　インストールされているMicrochip社製プラグインツール



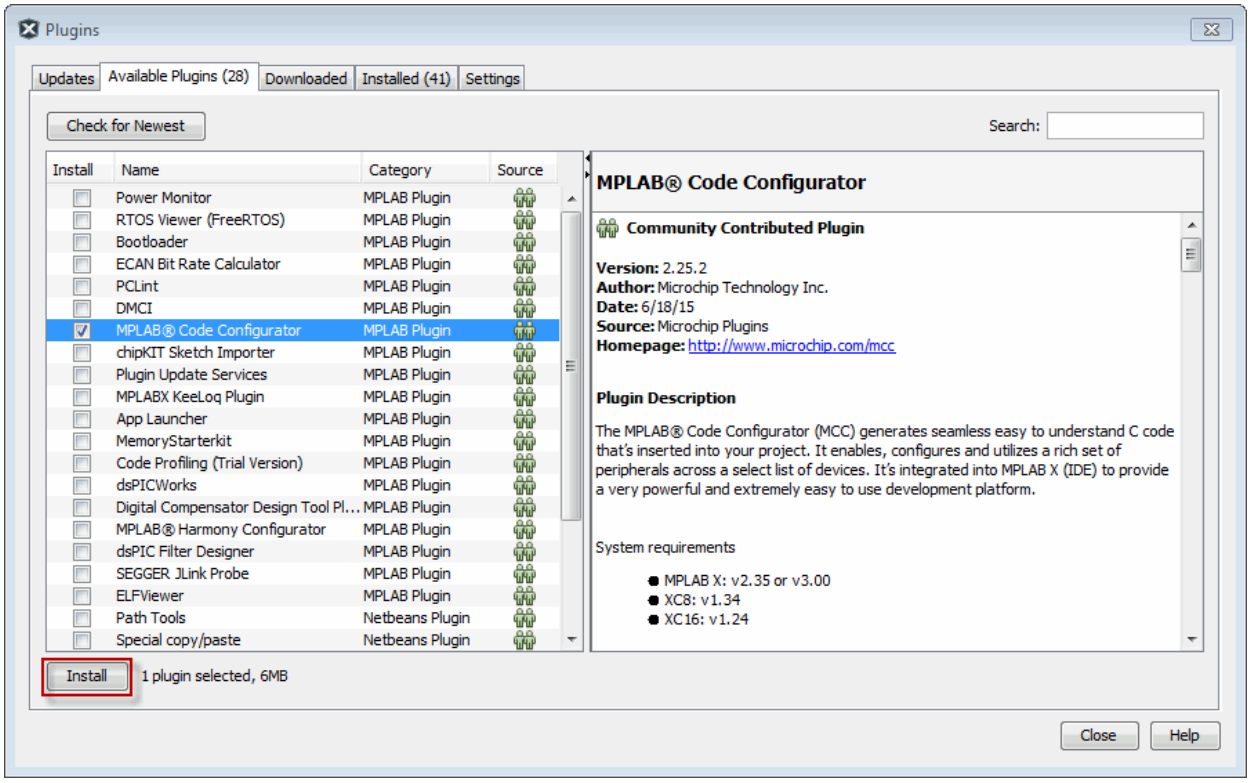
プラグイン マネージャからのプラグインの追加

利用可能なプラグインを表示およびインストールする方法

1. MPLAB X IDEで[Tools] > [Plugins]を選択し、**[Available Plugins]**タブをクリックします。
2. チェックボックスにチェックを入れてプラグインを選択し、[Install]をクリックします(下図参照)。
3. 画面の指示に従って、プラグインをダウンロードおよびインストールします。
   Note:　プラグインによっては、インストールされる機能が他のプラグイン内のモジュールに依存するものがあります。そのような場合、プラグイン マネージャは警告を表示します。
4. インストール後、プラグインは[Installed]タブに表示されます。このプラグインに対して無効にする、再度有効にする、アンインストールする、のいずれかの操作を実行できます。
5. [Tools] > [Embedded]を選択して、追加したツールを確認します。追加したツールが表示されない場合、MPLAB X IDEを一度終了してから再起動すると表示される場合があります。

[Help]ボタンをクリックすると、プラグインのインストールに関するヘルプを表示できます。

図5-41　利用可能なMlicrochip社製プラグインツール

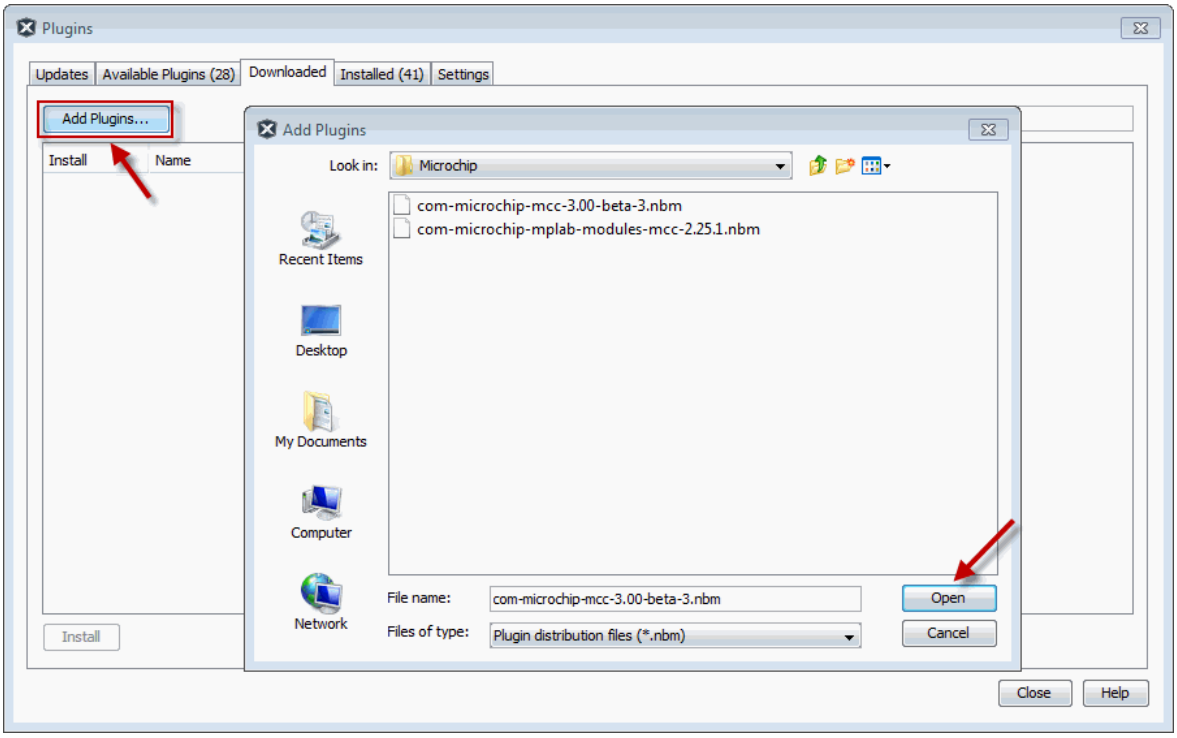


組み込みコード集からのプラグインの追加

利用可能なプラグインを表示、ダウンロード、インストールする方法

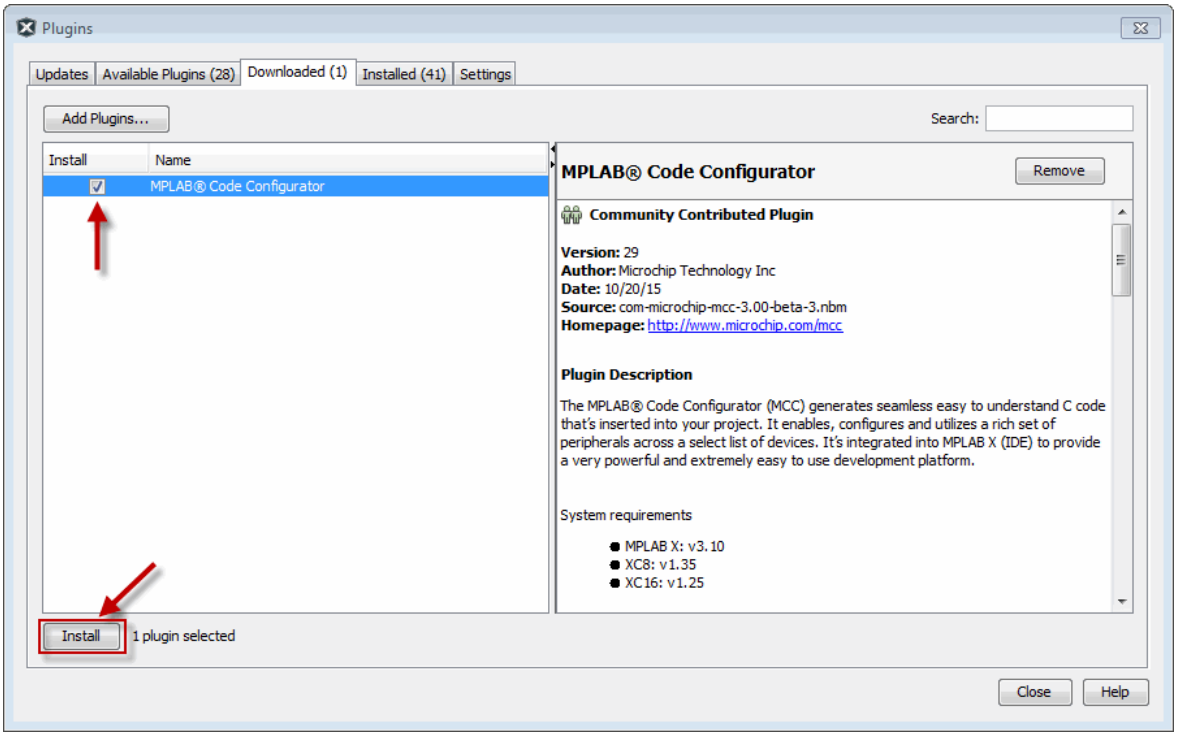
1. 以下のEmbedded Code Sourceウェブサイトを開きます。
   http://www.embeddedcodesource.com/codedeveloper/microchip-technology
2. プラグインを選択し、ウェブページの指示に従ってローカルフォルダにダウンロードします。
3. ダウンロードしたZIPファイルを解凍して.nbmファイルを取り出します。
4. MPLAB X IDEで[Tools] > [Plugins]を選択し、**[Downloaded]**タブをクリックします。
5. **[Add Plugins]**をクリックします。.nbmファイルを検索、選択して開きます。

図5-42　ダウンロード済みMicrochip社製プラグインツール



6. プラグイン名の隣のチェックボックスがチェックされている事を確認し、[Install]をクリックしてプラグインをインストールします。

図5-43　ダウンロード済みMicrochip社製プラグインツールのインストール



7. 画面の指示に従って、プラグインをダウンロードおよびインストールします。
Note:　プラグインによっては、インストールされる機能が他のプラグイン内のモジュールに依存するものがあります。そのような場合、プラグイン マネージャは警告を表示します。

8. インストール後、プラグインは[Installed]タブに表示されるます。このプラグインに対して無効にする、再度有効にする、アンインストールする、のいずれかの操作を実行できます。
9. [Tools] > [Embedded]を選択して、追加したツールを確認します。追加したツールが表示されない場合、MPLAB X IDEを一度終了してから再起動すると表示される場合があります。

プラグインの更新

使用中のプラグインに更新版が存在するかどうかを判断する方法

1. MPLAB X IDEで[Tools] > [Plugins]を選択し、[Updates]タブをクリックします。
2. [Check for Updates]をクリックします。
3. 更新版が表示される場合、プラグイン名がチェックされている事を確認し、**[Update]**をクリックします。
4. 画面の指示に従って、プラグインを更新します。

MPLAB X IDEの新バージョンをインストールしても、インストール済みのプラグインは更新されません。プラグインは、そのリリースに付けられたバージョンのインターフェイスでテストされています。全てのプラグインがバージョン間で移行可能とは限らず、そのまま流用される事はありません。これはNetBeansに対してもあてはまります。従って、新しいバージョンのMPLAB X IDEをインストールした場合、プラグインの再インストールが必要な場合があります。

更新センターの設定

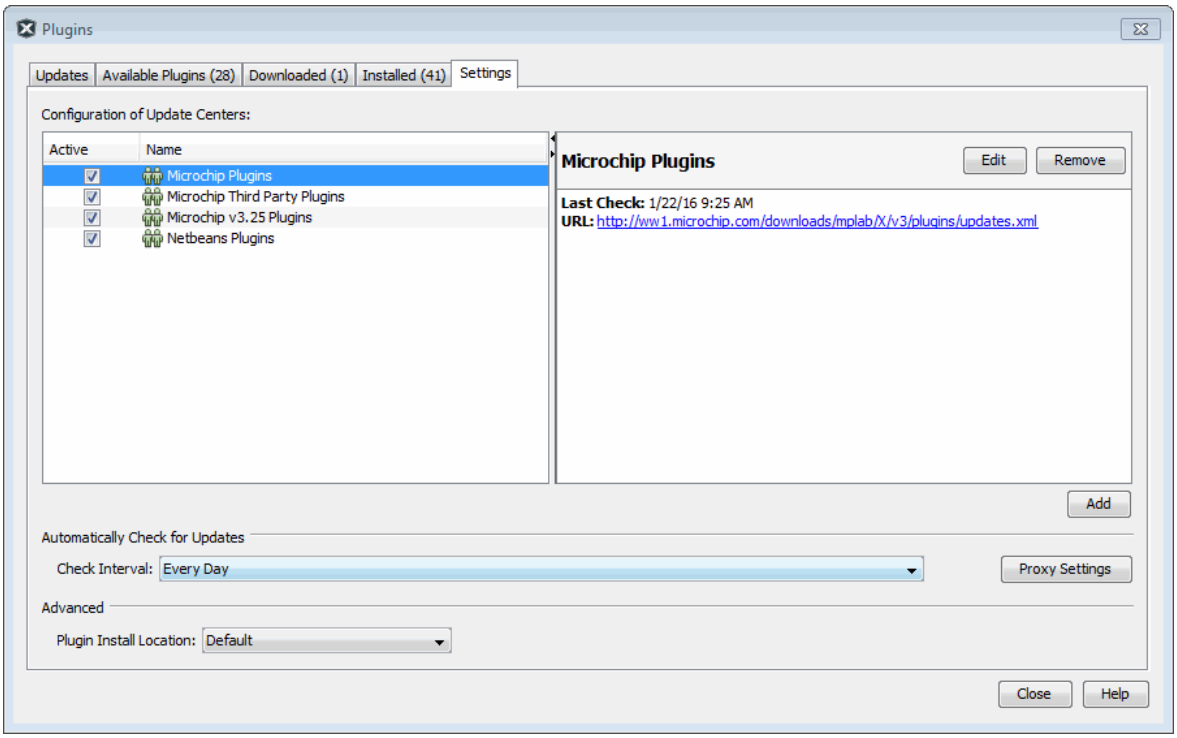
[Plugins]ウィンドウに利用可能なプラグインを表示させるには、更新センターを設定する必要があります。MPLAB X IDEには2つのMicrochip更新センターがあらかじめ設定されています。

- lhttp://ww1.microchip.com/downloads/mplab/X/plugins/updates.xml
- http://ww1.microchip.com/downloads/mplab/X/thirdpartyplugins/updates.xml

プラグイン マネージャで別の更新センターを設定する方法

1. [Tools] > [Plugins]を選択し、[Settings]タブをクリックします。
2. [Add]ボタンをクリックして、[Update Center Customizer]ダイアログを開きます。
3. 更新センターの名前を入力します。
4. 更新センターのURLを入力します。
5. **[OK]**をクリックします。

図5-44　Microchip更新センターの設定



インストール済みプラグインの設定

MPLAB X IDEにプラグインをインストールすると、設定が必要な場合があります。プラグイン設定オプションは、[Tools] > [Options]の[Plugins]に表示されます。「[Tools Options]ウィンドウ、[Plugins]」を参照してください。

プラグインコードの保存場所

プラグインコードはMPLAB X IDEのユーザ コンフィグレーション データと一緒に保存されています。8.7「ユーザ コンフィグレーション データの表示」を参照してください。

6. 高度な作業とコンセプト

本章ではより高度な作業の方法と便利なコンセプトを説明します。

表6-1　高度な作業とコンセプトの内容

1 高速化	1. MPLAB X IDEの高速化(IDEの動作が遅い場合) 2. ビルド時間の短縮(MPLAB X IDEで並列makeを使う)
2 プロジェクト関連作業	1. 複数プロジェクトの使い方(複雑なアプリケーション開発時) 2. 複数コンフィグレーションの使い方(同じプロジェクトコードで複数プロジェクト プロパティを選択可能にする) 3. デュアルコア プロジェクトの作成(dsPIC33CHデュアルコア デバイスを使う場合) 4. ユーザMakefileプロジェクトの作成(MPLAB X IDEで外部makefilesを使う場合) 5. ソースファイル フォルダにリンクしたリソースの使用(プロジェクト ソースベースまたはライブラリ バージョンの変更(MPLAB Harmony)) 6. サードパーティ製ツールの使い方(MPLAB X IDEプロジェクト内) 7. コードカバレッジの使用(実行されたコードの割合をテスト) 8. ログデータ(実行とデバッグの問題を記録する) 9. MPLAB X IDEプロジェクトのパッケージ化(プロジェクト内のファイルを圧縮) 10. ターゲットとのハードウェア ツールの接続により、デバッグ機能が決まる場合があります。「ハードウェア ツールの接続とデバッグ」を参照してください。
3 コンセプト	1. チェックサムの定義 2. コンフィグレーションの種類

6.1 MPLAB X IDEの高速化

MPLAB X IDEの動作が遅い場合、以下を検討します。

PCヒープの増加

mplab_ide.confファイルを編集すると、MPLAB X IDEに割り当てられているメモリ容量を変更できます。このファイルは編集前にバックアップしておく事を推奨します。このファイルを編集すると、変更内容はMPLAB X IDEを次回起動した時に有効になります。

- **Windows 64ビット** - `C:\Program Files (x86)\Microchip\MPLABX\vx.xx\mplab_platform\etc`
- **Windows 32ビット** - `C:\Program Files\Microchip\MPLABX\vx.xx\mplab_platform\etc`
- **Linux** - `/opt/microchip/mplabx/vx.xx/mplab_platform/etc`
- **macOS** - `/Applications/microchip/mplabx/vx.xx/Contents/Resources/mplab_platform/etc`

vx.xxはMPLAB X IDEのバージョン番号です。

以下の行は既定値を収めています。

```
default_options="-J-Dnb.FileChooser.useShellFolders=false -J-Dcrownking.stream.verbosity=very-quiet -J-Xms256m -J-Xmx512m -J-XX:PermSize=128m -J-XX:MaxPermSize=384m -J-XX:+UseConcMarkSweepGC -J-XX:+CMSClassUnloadingEnabled"
```

太字の部分は以下の通りです。

-Xms256m: JVMが256 MB以上のヒープを割り当てる事を意味します。

-Xmx512m: JVMが512 MB以下のヒープを割り当てる事を意味します。

-XX:PermSize=128m: JVMが、ヒープに収まらないデータの追加に必要な空間として128 MBを割り当てる事を意味します。

-XX:MaxPermSize=384m: JVMが、ヒープに収まらないデータの追加に必要な空間として384 MB未満を割り当てる事を意味します。

以下のエラーが発生しない限り、PermSizeまたはMaxPermSizeを変更する必要はありません。
`java.lang.OutOfMemoryError: PermGen space.`

最も重要な部分は`-Xmx512m`です。この値はMPLAB X IDEが使うヒープ量の最大値を指定しています。一見、大きなヒープ量を設定した方が良いように見えるかもしれません。しかし、MPLAB X IDEにメモリを使う事は他のアプリケーションとシステム機能に使うメモリが少なくなる事を意味しています。

メモリモニタを有効にすると、IDEが使うメモリの量を監視できます。ツールバー領域の空いた所を右クリックして[Memory]を選択します。または、[View] > [Toolbars] > [Memory]を選択します。

`mplab_ide.conf`の値を変えない限り、上限値は512 MBです。`-Xmx512m`の設定は128 MB刻みで変更する事を推奨します。大容量のメモリを使える場合、128 MBを超える刻みで増やす事もできます。しかし、システムの他の部分のために十分なメモリが残っている事を確認してください。

[Change Debug]ウィンドウの使用

デバッグツールを使うとMPLAB X IDEが遅くなる場合、以下を試します。

- 必要なウィンドウのみを表示させます。MPLAB X IDEデバッガは、デバッグツールの使用中、開いているウィンドウを全て監視するためです。
- デバッグのステップ実行中の[Stack]ウィンドウの更新頻度を低くします。[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])ウィンドウの**[Embedded]**ボタンをクリックし、**[Generic Settings]**タブの[Disable auto refresh for call stack view during debug sessions]と[On mouse over structure and array expressions, evaluate integral members only]にチェックを入れます。

6.2 ビルド時間の短縮

PCによっては、並列make(12.16.2「[Project Options]タブ参照)を使ってプロジェクトのビルド時間を短縮できる場合があります。並列makeをサポートしていない言語ツールもあります。

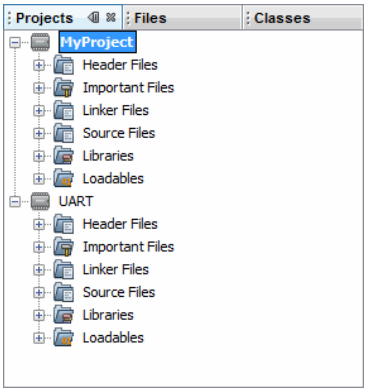
オペレーティング システム(OS)の変更を検討する方法もあります。OSによっては、より高速にファイルをアクセスできます。MPLAB X IDEはWindows, Linux, Mac OSをサポートしているため、その中でどれが適しているかを検討します。

6.3 複数プロジェクトの使い方

MPLAB X IDEでは、複数のプロジェクトを使って作業できます。

MPLAB X IDEで複数のプロジェクトを開いて[Projects]ウィンドウに表示できます。このウィンドウの詳細は本書の12.15「[Projects]ウィンドウ」を参照してください。

図6-1 [Projects]ウィンドウに表示した複数のプロジェクト



複数プロジェクトを開く

MPLAB X IDEではプロジェクトを新規作成する([File] > [New Project])または複数のプロジェクトを開く([File] > [Open Project])事ができます。または、以下のようにショートカットで--openディレクティブを使えます。

```
"C:\Program Files
(x86)\Microchip\MPLABX\v3.15\mplab_platform\bin\ mplab_ide.exe" --
open "C:\MPLAB_X_Projects\MyProject1"
```

次に、複数のショートカットを作成して、様々な場所から複数のプロジェクトを開く事ができます。

プロジェクト タイプの定義

以下で説明するように、各プロジェクトはアクティブ(または開発のフォーカス)にする事ができます。ただし、メインプロジェクトは常に1つだけです。

メイン プロジェクト

プロジェクト名を右クリックして[Set as Main Project]を選択すると、プロジェクトをメイン プロジェクトとして選択できます。メイン プロジェクトの名前は太字で表示されます。この場合、全てのツールバー機能が1つのメインプロジェクトで実行されます。

アクティブなプロジェクト
メイン プロジェクトを設定せずに作業する事もできます。アクティブなプロジェクトとは、メイン プロジェクトがない時にフォーカスされるプロジェクトです。

[Projects]ウィンドウ内でいずれかのプロジェクトをクリックすると、そのプロジェクトがアクティブになります。その場合IDEは、作業中のプロジェクトをコンテクストで判断します。すなわち、あるファイルにフォーカスするようにエディタを設定すると、そのファイルを含むプロジェクトがアクティブ プロジェクトになります。アクティブプロジェクトは、全ての操作([Debug Project]アイコンのクリック等)を受けます。

プロジェクト コードの開発

プロジェクトのコードの開発とデバッグの方法

1. [Projects]ウィンドウ内のプロジェクトをクリックして[Debug] > [Debug Project]を選択し、プロジェクトのデバッグを開始します。
2. 複数のデバッグ セッションを同時に実行する場合、[Sessions]ウィンドウを開くと([Window] > [Debugging] > [Sessions])現在実行中のデバッグ セッション間で切り換える事ができます。

デバッグ セッションを切り換えると[Watches]ウィンドウ、変数、メモリ表示が現在選択しているデバッグ中のプロジェクトのものに切り換わります。ステータスビットも現在のプロジェクトのものに切り換わります。ダッシュボードは、デバッグ プロジェクトまたは[Project]ウィンドウ内のプロジェクトのいずれであれ、最後に選択したプロジェクトの内容を表示します。これは設計仕様に基づいた動作です。

[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])、**[Embedded]**ボタン、**[Generic Settings]**タブでオプション[Maintain active connection to hardware tool]を選択しておくと、セッション/プロジェクト切り換え時に再接続が行われます。これは設計仕様に基づいた動作です。

複数プロジェクトのグループ化

複数プロジェクトを使うもう1つの方法はグループ化です。[File] > [Project Group]を選択し、[Create New Group]ダイアログ ボックスでプロジェクト グループを選択するか新規に作成します。

6.4 複数コンフィグレーションの使い方

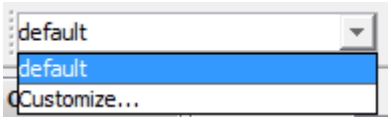
MPLAB X IDEでは、同一プロジェクトで複数のビルド コンフィグレーションを使えます。これは、複数のプラットフォームでコンパイル可能なコード(Microchip社のアプリケーション ライブラリ デモプロジェクト等)にとって便利です。

新規プロジェクトを作成すると、「default」という名前のコンフィグレーションが作成されます。追加コンフィグレーションは、以下のセクションの手順に従って作成します。

6.4.1 [Manage Configurations]ダイアログ

ユーザ コンフィグレーションの作成は、以下のどちらかの手順で始めます。

- ツールバーのドロップダウン メニューから[Customize]を選択して、[Project Properties]ウィンドウを開く

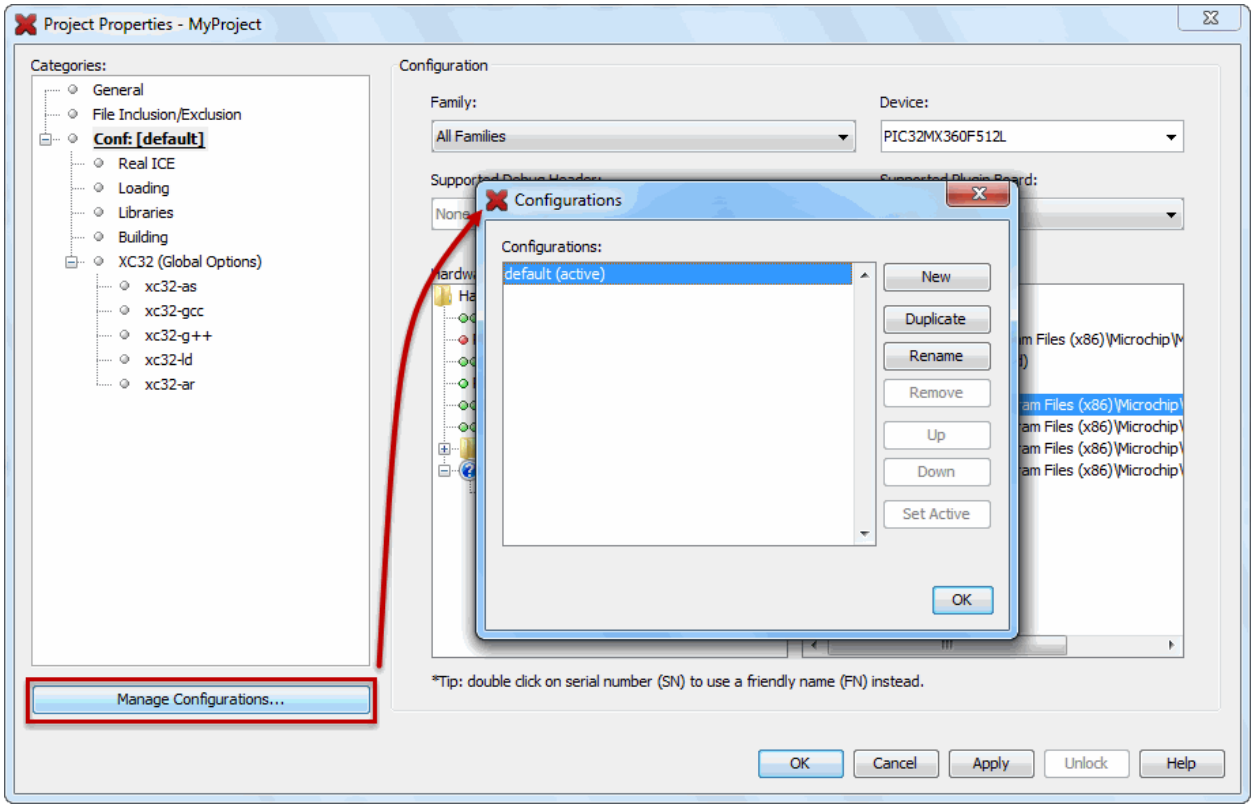


- プロジェクト名を右クリックして[Properties]を選択し、[Project Properties]ダイアログを開く

[Project Properties]ウィンドウで**[Manage Configurations]**をクリックして、[Configurations]ダイアログを開きます。このウィンドウでは、既存コンフィグレーションの名前を変更するか、新規コンフィグレーションを追加するか、既存コンフィグレーションを複製できます。名前にスペースがある場合は全てアンダースコアに置き換えられます。

1つのプロジェクトに対して複数のコンフィグレーションを作成した場合、**[Manage Configurations]**ボタンまたはドロップダウン メニューを使って、いずれか1つのコンフィグレーションをアクティブにします。

図6-2　[Project Properties] – [Configurations]ダイアログ

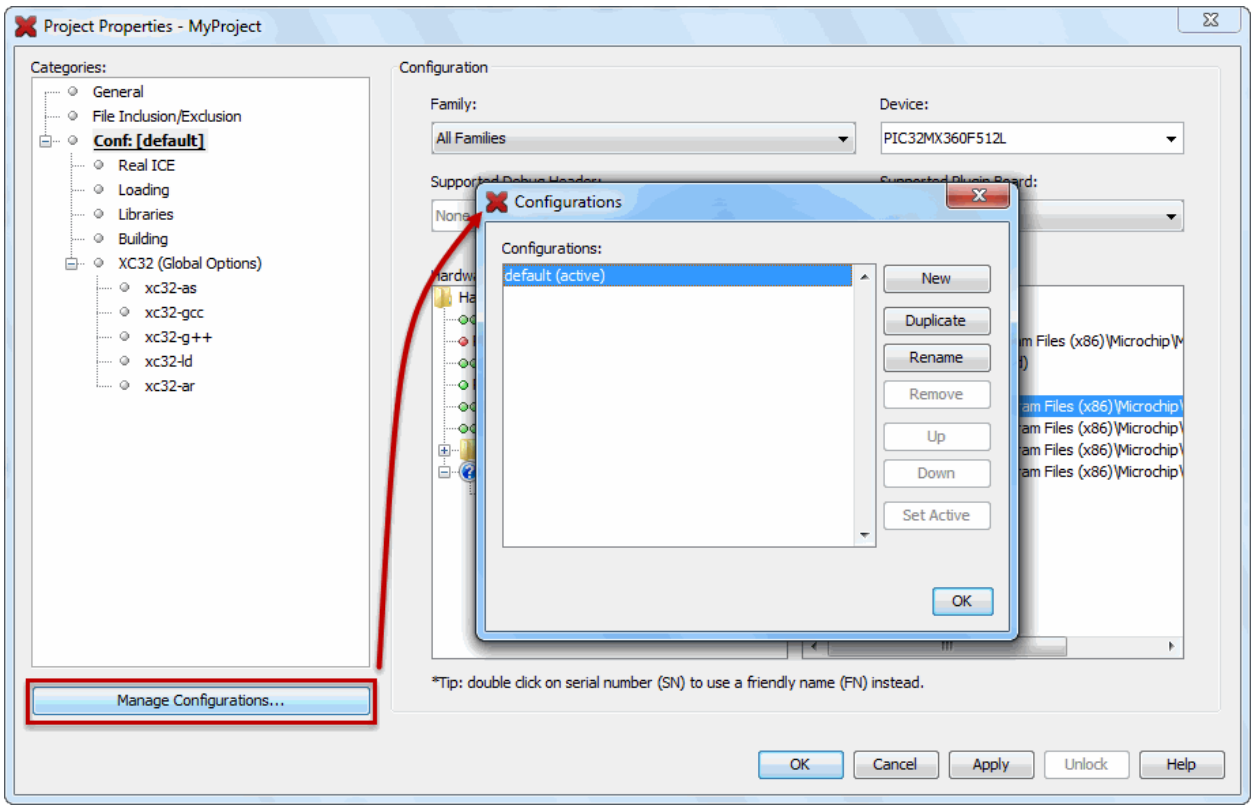


6.4.2　新規コンフィグレーションの追加

プロジェクトに新規コンフィグレーションを追加する方法

1. [Project Properties]ウィンドウで**[Manage Configurations]**をクリックします。
2. [Configurations]ダイアログで、プロジェクト コンフィグレーションを選択して**[New]**をクリックします。
3. [New Configuration Name]ダイアログに名前を入力します(例: 「My Test」(下図参照)。**[OK]**をクリックしてダイアログを閉じます。名前にスペースは使えないため、[Configurations]ではこの名前は「My_Test」に変更されます。
4. **[OK]**をクリックして[Project Properties]ウィンドウに戻ります。以上で、「Debug」コンフィグレーション(Conf: My_Test)が表示されるはずです。

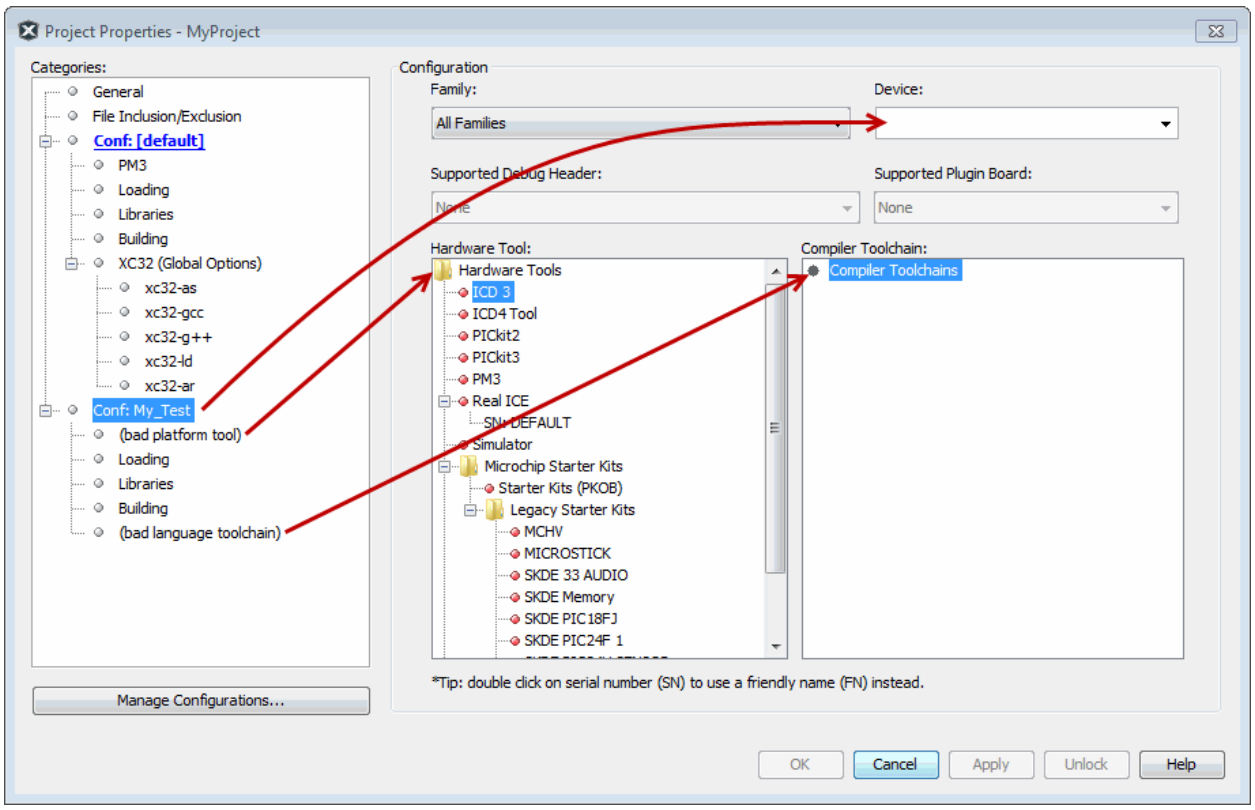
図6-3　新規コンフィグレーションの作成



新規コンフィグレーションを追加した場合、複数の項目を[Project Properties]ウィンドウで割り当てる必要があります(下図参照)。

1. デバイス – ハードウェア ツールとコンパイラのサポートを確認するため、これを最初に選択する必要があります。
2. ハードウェア ツール
3. コンパイラ ツールチェーン

図6-4　新規コンフィグレーション



6.4.3　複製したコンフィグレーションの追加

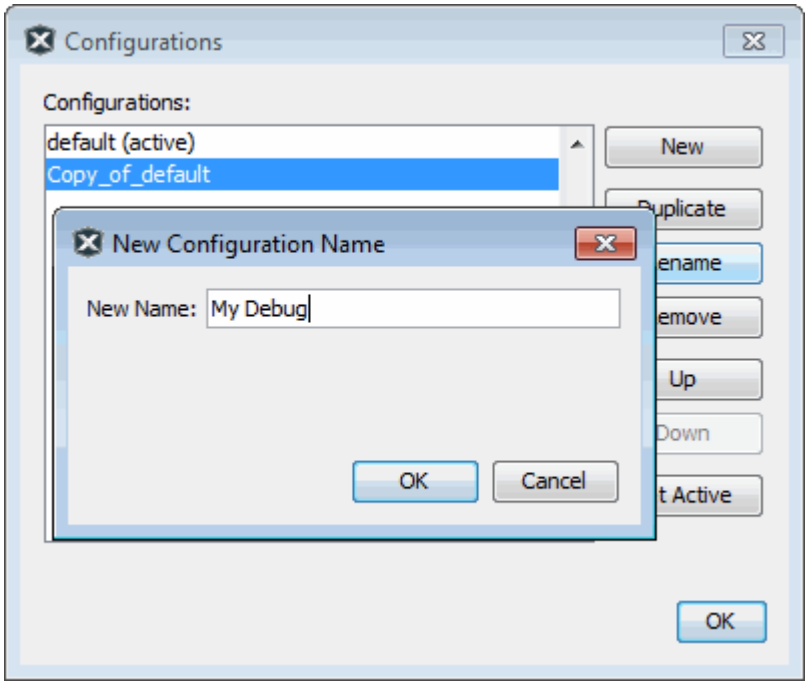
既存コンフィグレーションの複製を追加し、これを編集して使う事ができます。

複製コンフィグレーションを作成する理由の1つは、デバッグ用コンフィグレーションを作成するためです。MPLAB X IDEはMicrochip社製ツールと組み合わせて使うデバッグ用マクロ(4.13「コードのデバッグ」の[Debug Macros Generated]参照)を提供しています。しかし、専用のコンフィグレーションを作成する事で、ユーザ独自のデバッグ用マクロを使えるようになります。またはサードパーティのツールで同じデバッグ機能を使えるようになります。

デバッグ コンフィグレーションの設定方法

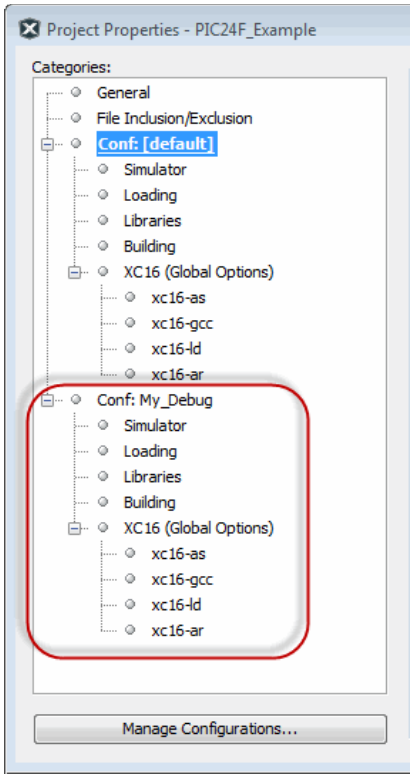
1. [Project Properties]ウィンドウで**[Manage Configurations]**をクリックします。
2. [Configuration]ダイアログで、プロジェクト コンフィグレーションを選択して**[Duplicate]**をクリックします。
3. **[Rename]**をクリックし、[New Configuration Name]ダイアログで名前(例: 「My Debug」)を入力します(下図参照)。**[OK]**をクリックしてダイアログを閉じます。名前にスペースは使えないため、[Configurations]ではこの名前は[My_Debug]に変更されます。

図6-5　デバッグ コンフィグレーションの作成



4. **[OK]**をクリックして[Project Properties]ウィンドウに戻ります。以上で、デバッグ コンフィグレーション(Conf: My_Debug)が表示されるはずです(下図参照)。

図6-6　デバッグ コンフィグレーション



デバッグと同様に、コード内のマクロを使って他のコンフィグレーションに切り換えるには、6.4.6「コンフィグレーション マクロ」を参照してください。

6.4.4 コンフィグレーション内のファイル管理

プロジェクト ファイルは、作成したコンフィグレーションに個別に、またはグループとして割り当てる事で管理できます。

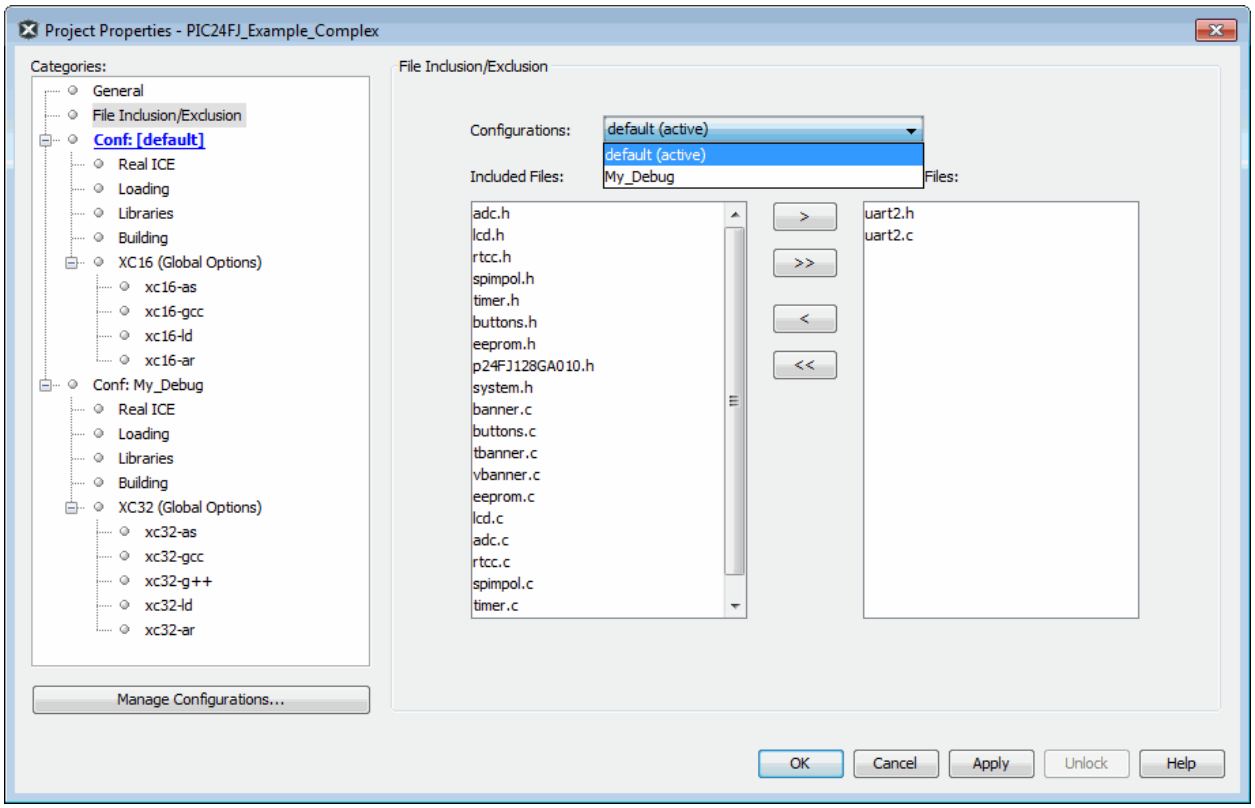
[Projects]ウィンドウ

ツールバーの[Set Project Configuration]ドロップダウン ボックスからコンフィグレーションを選択します。次に、1つまたは複数のファイルを選択して(<Shift>+クリックまたは<Ctrl>+クリック)右クリックすると、コンテクスト メニューが表示されます。[Exclude file(s) from current configuration]を選択します。1つまたは複数のファイルを再び含めるには、コンテクスト メニューの[Include file(s) from current configuration]を選択します。

[Project Properties]ウィンドウ

多くのファイルに適用するコンフィグレーションを設定する場合、[Projects]ウィンドウでプロジェクト名を右クリックして[Project Properties]ウィンドウを開きます。[File Inclusion/Exclusion]カテゴリを選択します。[Configurations]ドロップダウンボックスからコンフィグレーションを選択し、このコンフィグレーションに含めるまたは除外するファイルを指定します。

図6-7 [Project Properties] – [Configurations]



6.4.5 コンフィグレーション名

一部の特殊文字はコンフィグレーション名に使えません。これは、ソースコード内のコンフィグレーション名を定義マクロ（「コンフィグレーション マクロ」）として使うためです。

ハイフン(-)、アンパサンド(&)等の特殊文字は自動的にアンダースコア(_)に置換されます。

6.4.6 コンフィグレーション マクロ

新規または複製コンフィグレーションを追加する場合、独自のコンフィグレーション関連マクロを作成するか、コード内の#ifdef文または他の文の自動生成マクロを使えます。これらはMPLAB XC Cコンパイラのプリプロセッサ マクロです。

ユーザ定義マクロ

独自のプリプロセッサ マクロの定義方法

1. [Project Properties]ウィンドウで、プロジェクト コンフィグレーションをクリックしてアクティブにします。そのコンフィグレーションの下で、ツールチェーン内のコンパイラをクリックします。
2. [Preprocessing and messages]オプション カテゴリで、マクロを定義できるオプションを選択し、関連するテキストボックスをクリックします。
3. ポップアップ ダイアログでマクロ名を入力し、**[OK]**をクリックします。以上で、このマクロはアクティブ コンフィグレーションに関連付けられました。
   マクロには、コンフィグレーションと同じ名前(例: 6.4.3「複製コンフィグレーションの追加」の「My_Debug」)か、任意の名前(例: 「DebugConfig」)を付ける事ができます。

   マクロ名(および以下で説明する文字列)は大文字と小文字を区別します。

以上で、プロセッサマクロは以下のコンディショナル テキスト内で使えるようになりました。

```
// If My_Debug configuration is active
#ifdef DebugConfig
fprintf(stderr,"This is a debugging message\n");
#endif
```

さらに、コンフィグレーション名と一致する文字列値をマクロに割り当てると、通常のコードで使えます。

```
#ifdef DebugConfig
#define CFG_NAME “My_Debug”;
// other defines
#endif
#ifdef DefaultConfig
#define CFG_NAME “My_Default”;
// other defines
#endif
:
int main(int argc, char** argv)
{
printf(CFG_NAME);
return{EXIT_SUCCESS};
}
```

CFG_NAMEマクロは常にアクティブ コンフィグレーションの名前を保持します。

自動生成マクロ

新規または複製コンフィグレーションを作成すると(例: My_Test)、関連するマクロ(例: XPRJ_My_Test)も作成されます。マクロには、コンフィグレーション名(「My_Test」)と一致する文字列値が含まれます。

前のセクションのように#ifdef文用のマクロか、以下のようにコンパイル時にマクロの値を使う事ができます。

```
int main(int argc, char** argv)
{
printf(CFG_NAME);
return{EXIT_SUCCESS};
}
```

CFG_NAMEマクロは自動的にアクティブ コンフィグレーションの名前を保持します。

6.5 デュアルコア プロジェクトの作成

デュアルコア デバイス(例: dsPIC33CH DSC)には独立した動作が可能なマスタコアとスレーブコアがあり、アプリケーション開発時はプログラミングとデバッグを別々に行えます。両方のプロセッサ (マスタとスレーブ) サブシステムにはそれぞれ専用の割り込みコントローラ、クロック ジェネレータ、ICD、ポートロジック、I/O MUX、PPSを備えています。デュアルコア デバイスはシングルダイ上に別々の2つのdsPIC® DSCがあるのと同じです。

デュアルコア デバイスには、プロジェクトの様々な作成および使用方法に対応するいくつかの動作モードがあります。

マスタのみのプログラミングとデバッグ

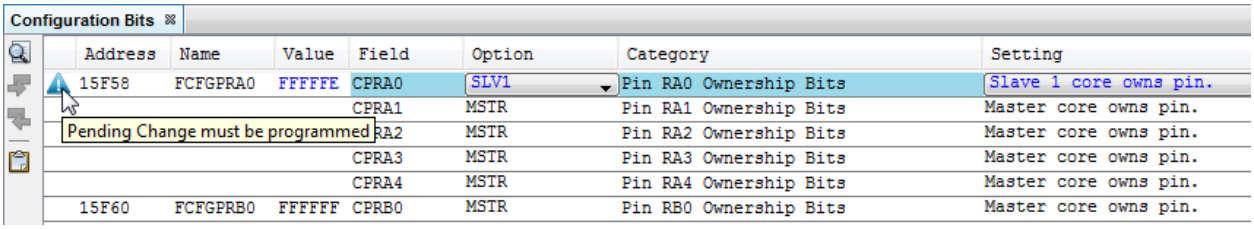
デュアルコア デバイスは、既定値ではマスタのみのモードです。MSI(マスタスレーブ インターフェイス)では、MSI1マスタ制御レジスタ(MSI1CON)のスレーブ イネーブルビット(SLVEN)が「0」にセットされています。

このデバイスモード用のプロジェクトは、他のMicrochip社製デバイスのプロジェクトと同じです。詳細は「新規プロジェクトの作成」を参照してください。

スレーブのみのプログラミング

スレーブ プロジェクトを独立して動作させる場合、スレーブ用のポートを割り当てる必要があります。従って、コンフィグレーション ビットの設定(スレーブに割り当てるポート等)でマスタをプログラミングするマスタ プロジェクト(例: MasterStub)が必要です。詳細は「[Configuration Bits]ウィンドウでのコンフィグレーション ビットの設定」を参照してください。

図6-8　マスタ コンフィグレーション ビット



スレーブのみのプログラミング モードには2つのプロジェクトが必要です。

1. 「Create a New Project」に従ってMasterStubを作成し、スレーブを有効にします(SLVEN = 「1」)。コンフィグレーション ビットは前述のようにセットします。
2. 「Create a New Project」に従ってSlaveOnlyプロジェクトを作成します。ただし、デバイスは「S1」バージョンを使います(例 dsPIC33CH128MP508S1)。

最初にMasterStubをプログラミングし、次にSlaveOnlyをプログラミングします。

スレーブのみのデバッグ

前のセクションで作成したSlaveOnlyプロジェクトをデバッグするには、コンフィグレーションの設定でバックグラウンド デバッグ(S1DEBUG)を有効にします。

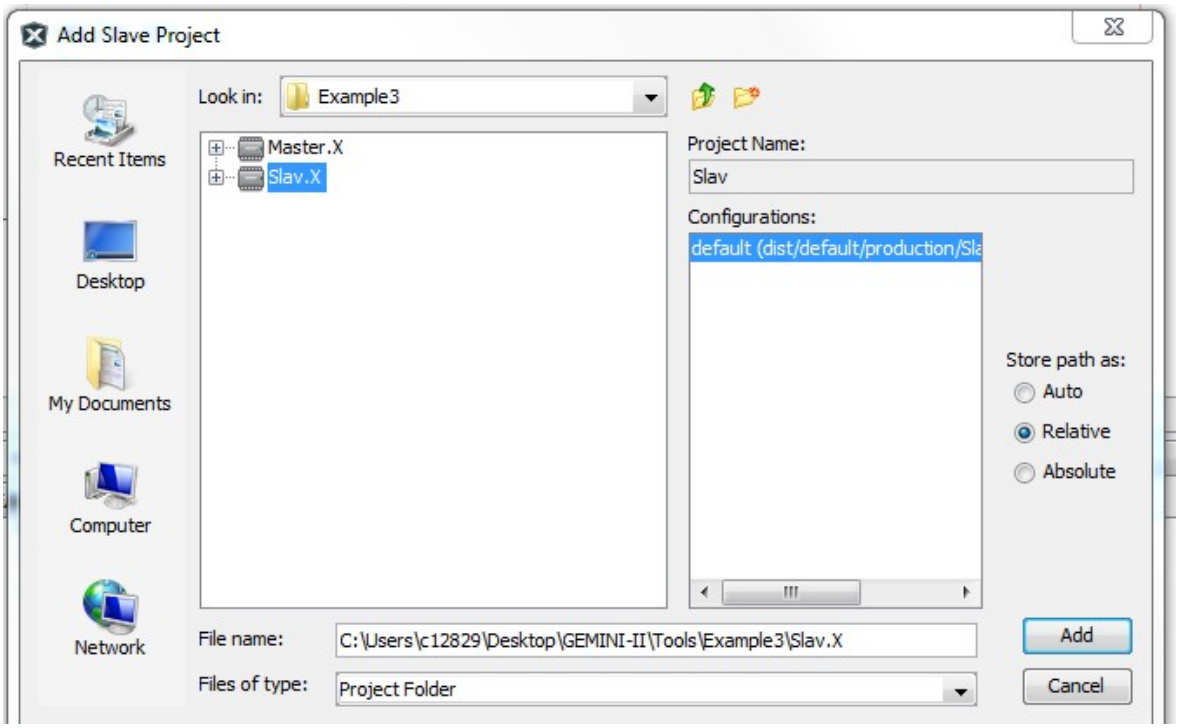
```
// FS1ICD
#pragma config S1ICS = PGD2 // ICD Communication Channel Select bits
                            // (Communicate on PGEC2 and PGED2)
#pragma config S1DEBUG = ON // Background Debug (Enabled)
 :
```

マスタおよびスレーブのプログラミングとデバッグ

このモードでは、マスタ プロジェクトがスレーブ プロジェクトにリンクされます。つまり、スレーブコードはマスタ内に存在し、マスタはこのコードをスレーブに転送する必要があります。

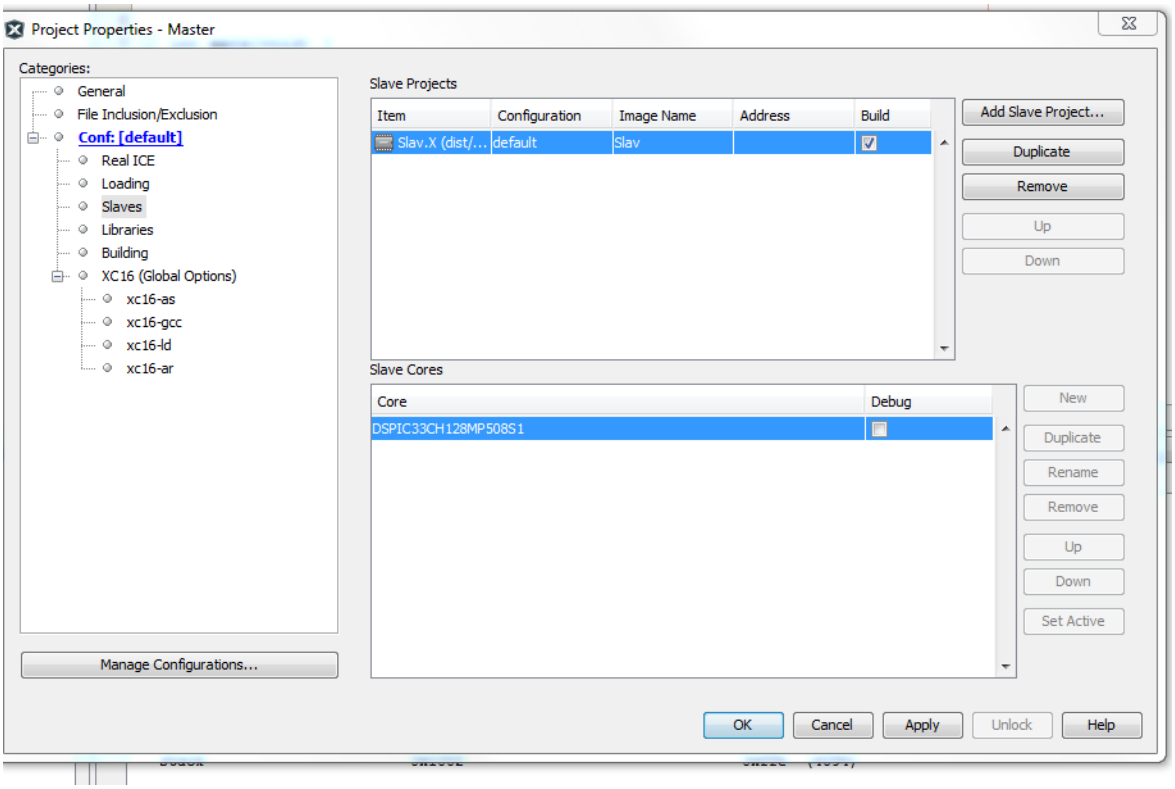
1. 「スレーブのみのプログラミング」で説明した通り、マスタおよびスレーブ プロジェクトを作成します。
2. [Projects]ウィンドウで、マスタ プロジェクトに「Slaves」という名前のフォルダがあります。このフォルダを右クリックして[Add Slave Project]をクリックします。次に、ダイアログでスレーブ プロジェクトを選択して**[Add]**をクリックします。

図6-9　[Add Slave Project]ダイアログ



3. マスタ プロジェクトを右クリックして[Properties]を選択し、[Project Properties]ウィンドウを開きます。[Slaves]カテゴリをクリックし、[Build]チェックボックスにチェックを入れます。**[Apply]**をクリックします。

図6-10　マスタおよびスレーブ プロジェクト



4. マスタコードに以下を追加します。

```
#include "Slav.h"
//The function to transfer the code from slave to master and start the slave is below.
 _program_slave(1,0,Slav);
 _start_slave();
```

5. マスタをビルドしてプログラミングする事により、マスタおよびスレーブのプロジェクトがビルドされ、マスタおよびスレーブのコアがプログラミングされます。

6.6 ユーザMakefileプロジェクトの作成

外部makefileを使うプロジェクトを作成します。これは、プロジェクトがMPLAB X IDEの外にビルドされているが、MPLAB X IDEを使ってデバッグを実行する場合に便利です。

makefileプロジェクトを作成するには、[New Project]ウィザードを使います。

6.6.1 [New Project] - ユーザMakefile

[New Project]ウィザードを開く方法については4.1.2「[New Project]ウィザードの起動」を参照してください。

[New Project]ウィザードが起動し、プロジェクトの新規作成を支援します。

Step 1. Choose Project: [Microchip Embedded]カテゴリを選択し、プロジェクト タイプ[User Makefile Project]からプロジェクトを選択します。

Step 2. Select Device: プロジェクトで使うデバイスを[Device]ドロップダウン リストから選択します。選択肢を絞り込むために、最初にデバイスファミリを選択します。

Step 3. Select Header: このステップは、選択したデバイスでヘッダが利用可能である場合に表示されます。デバッグ用にヘッダが必要かどうか、デバイスがデバッグ回路を内蔵しているかどうかは、以下のどちらかの文書で確認します。確認後にヘッダを使うかどうかを選択します。

- 『Processor Extension Pak and Debug Header Specification』
- 『エミュレーション拡張パック(EEP)およびエミュレーション ヘッダ ユーザガイド』

Step 4. Select Tool: アプリケーション開発に使うツールをリストから選択します。
選択したデバイスに対するツールのサポートレベルが、ツール名の横のマークで色分け表示されます。緑は完全サポート、黄はベータサポート、赤は未サポートを意味します。

Step 5. プラグインボードの選択: このステップは、選択したデバイスでプラグインボードが利用可能である場合に表示されます。[Supported Plugin Board]のドロップダウン リストで利用可能なプラグインボードを調べます。

Step 6. Select Project Name and Folder: 新規プロジェクトの名前と保存場所を選択します。表1を参照してください。

Step 7. Create User Makefile Project: データを入力してmakefileプロジェクトを設定します(表2参照)。このウィンドウでの一般的なデータ入力の方法は以下の通りです。

- パスにはスラッシュ「/」を推奨します。代わりにバックスラッシュ「\」も使えますが、エスケープする必要があるかもしれません「\\」。
- パスとファイル名にスペースを使わない事を推奨します。しかし、このウィンドウでパスおよびファイル名を入力する際にどうしてもスペースを使わなくてはいけない場合、エスケープ スペース (スペースの前にバックスラッシュ)を使います。スペースはGNU Makeにおける区切り文字であるため、スペースを使うと問題が生じる事が分かっています。

Step 8. Finish: 作成したプロジェクトが[Projects]ウィンドウ内に表示されます。

プロジェクト作成後、[Project Properties]ウィンドウの[Makefile]カテゴリでMakefileプロジェクトの設定を変更します。プロジェクトをHEXファイルとしてエクスポートする方法は12.15「[Projects]ウィンドウ – [Project]メニュー」を参照してください。

表6-2 表1: [Select Project Name and Folder]の設定項目

項目	説明
Project Name	プロジェクトの名前を指定します。
Project Location	プロジェクトの保存場所を入力または参照します。

(続き)

項目	説明
Project Folder	プロジェクト名とプロジェクトの保存場所に基づいて、プロジェクト フォルダの保存場所を表示します。 プロジェクト フォルダと作業フォルダの関係は6.6.2「ユーザMakefileプロジェクト フォルダおよび作業フォルダ」を参照してください。
Set as main project	作成されるプロジェクトをメイン プロジェクトに設定します。既定値でチェックが入っています。
Overwrite existing project	このチェックボックスは、プロジェクト名が既に使われている場合にアクティブになります。
Also delete sources	このチェックボックスは、プロジェクトが既に存在し、ソースコードが含まれている場合にアクティブになります。
Use project location as project folder	プロジェクトの保存場所と同じフォルダ内にプロジェクトを作成します。 これは、MPLAB IDE v8プロジェクトをインポートする際、MPLAB X IDEのプロジェクトを同じフォルダにする場合(ビルドも同様にする場合)に便利です。 MPLAB IDE v8が.mcpファイルを使ってプロジェクト情報を保存するのに対し、MPLAB X IDEはフォルダを使います。従って、MPLAB IDE v8プロジェクト(.mcpファイル)とフォルダを共有させる事ができるMPLAB X IDEプロジェクトは1つだけです。
Encoding	既定値でISO-8859-1に設定されます。通常これを変更する必要はありません。

表6-3　表2: [Create User Makefile Project]の設定項目

項目	説明
Working Directory	外部プロジェクトの保存場所を指定します。これは各種コマンド (ビルド、デバッグビルド、クリーン)が実行される既定値の保存場所です。 プロジェクト フォルダと作業フォルダの関係は6.6.2「ユーザMakefileプロジェクト フォルダおよび作業フォルダ」を参照してください。 このフォルダはユーザに代わってフォーマットの問題を処理するため、[Browse...]クリック時のフォルダを参照する事を推奨します。
Build command	MPLAB X IDE(アイコンまたはメニュー)でビルドまたは[Run]コマンドが要求された場合、代わりにこのコマンドに従ってビルドを実行します。 ビルドパス*: Windowsの場合、引用符を使います。LinuxまたはMac OSの場合、スペースはエスケープする必要があります。
Debug build command	MPLAB X IDE(アイコンまたはメニュー)で[Debug]コマンドが要求された場合、代わりにこのコマンドに従ってデバッグビルドを実行します。 デバッグビルド パス*: Windowsの場合、引用符を使います。LinuxまたはMac OSの場合、スペースはエスケープする必要があります。 以下も参照してください。 http://microchipdeveloper.com/mplabx:work-outside-compile-for-debug
Clean command	MPLAB X IDE(アイコンまたはメニュー)で[Clean]コマンドが要求された場合、代わりにこのコマンドに従ってクリーンを実行します。 クリーンパス*: Windowsの場合、引用符を使います。LinuxまたはMac OSの場合、スペースはエスケープする必要があります。
Image name	量産イメージファイル(HEX)のパスと名前を入力します。イメージパス*: スペースは全てエスケープする必要があります。
Debug image name	デバッグイメージ ファイル(ELFまたはCOF)のパスと名前を入力します。デバッグイメージ パス*: スペースは全てエスケープする必要があります。
User Include Paths	ユーザ インクルード ファイルを入力または参照します。インクルードパスはダイアログ(図6-11)で管理します。

(続き)

項目	説明
User Macros	マクロを追加または「解析および追加」します。 [Edit]: マクロ名を入力します。またはマクロを含むファイルを参照します。マクロはダイアログ(図6-12)で管理します。 [Parse]: マクロを自動的に解析します(MPLAB XCコンパイラのみ)。ダイアログの指示に従います。完了後、表示されている現在のマクロを選択して置換するか、現在のマクロの一覧に付け加えます。このダイアログの使用例は6.6.3「Makefileプロジェクトの例」を参照してください。

図6-11　[Select Include Paths]ダイアログ

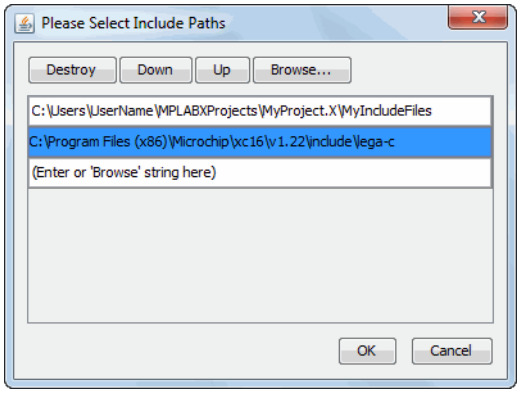
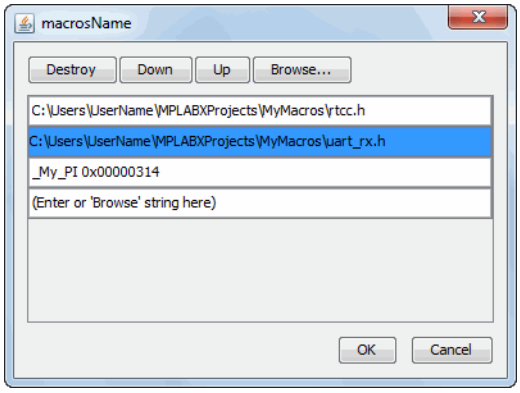


図6-12　[macrosName]ダイアログ



6.6.2　ユーザMakefileプロジェクト フォルダおよび作業フォルダ

プロジェクト フォルダは`Makefile`と`nbproject/Makefiles`を保存しています。MPLAB X IDEはプロジェクト フォルダでmakefileを実行します。次に作業フォルダに移動し、「Build command」または「Debug build command」を実行します。これによりGNU Makeが実行され、「Build command」として指定した処理が実行されます。

プロジェクト フォルダは、絶対パスで保存場所を定義する場合、作業フォルダに対してどこにでも配置できます。しかし、この方法を使うとプロジェクトの移植性が損なわれます。プロジェクトを移植しやすくするには、作業フォルダはプロジェクト フォルダに対して相対パス(例: プロジェクト フォルダの1レベル下)で指定します。

6.6.3　Makefileプロジェクトの例

この例では、MPLAB X IDEでユーザmakefileプロジェクトを作成する方法を示します。このmakefileプロジェクトは、バッチファイルとIDE外で作成したユーザコード上のmakeのどちらかを実行します。このユーザコードは、1つのファイル(`t.c`)から成ります。このファイルに記述されているシンプルなコードを以下に示します。

例: `t.c`のコード

```
#include <xc.h>
int main(void)
{ while(1) {
```

```
}
return 0;
}
```

本コードは、量産向けにはt_production.hexにリンクし、デバッグ向けにはt_debug.elfにリンクします。

この例では、t.c、バッチファイル(makeme.bat)、ユーザmakefile (Makefile)は以下のフォルダ(Windowsの場合)に保存されていると仮定します。

```
C:\Users\MCHP\MPLABXProjects\makestuff\myOwnCodeWithItsOwnMakefile
```

使用中のPC上でこの例を再現する場合、MCHPをユーザのUserNameで置き換えます。

- [User Makefile Project] – バッチファイルからコードを作成
- [User Makefile Project] – ユーザMakefileからコードを作成
- [User Makefile Project] – 完了
- [User Makefile Project] – マクロの解析

6.6.3.1 [User Makefile Project] – バッチファイルからコードを作成

コードをコンパイルするため、バッチファイル(makeme.bat)の引数としてproduction、debug、cleanのいずれかを使います。productionの場合、t_production.hexがビルドされます。debugの場合、t_debug.elfがビルドされます。cleanの場合、.elfおよび.hexファイルが消去されます。

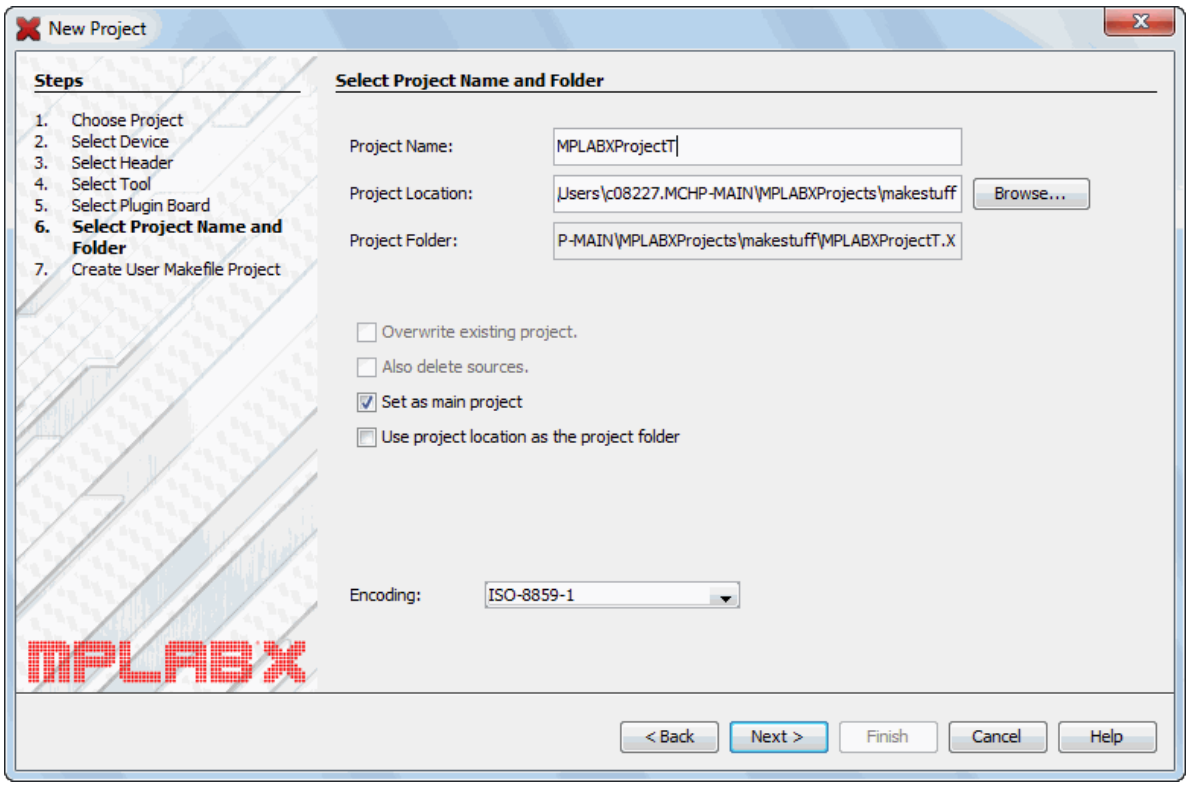
例: makeme.bat

```
::assumes xc32-gcc and xc32-bin2hex are on the path
rem @echo off
set COMPILER=xc32-gcc
set BIN2HEX=xc32-bin2hex
set PROCESSOR=32MX360F512L
if "%1" == "production" goto production
if "%1" == "clean" goto clean
:debug
%COMPILER% -mprocessor=%PROCESSOR% -g -D_DEBUG -o t_debug.elf t.c
goto end
:production
%COMPILER% -mprocessor=%PROCESSOR% -o t_production.elf t.c
%BIN2HEX% t_production.elf
goto end
:clean
del /F *.elf
del /F *.hex
goto end
:end
```

MPLAB X IDEでユーザmakefileプロジェクトを作成するには、6.6.1「[New Project] - ユーザMakefile」の手順を実行します。

1. **Choose Project:** [Microchip Embedded]、[User Makefile Project]を選択します。
2. **Select Device:** デバイスとしてPIC32MX360F512Lを選択します。
3. **Select Header:** このデバイスに利用できるヘッダはありません。
4. **Select Tool:** SNでハードウェア ツールまたは[Simulator]を選択します。この例では「Simulator」を使います。
5. **Select Plugin Board:** この例ではヘッダを使いません。
6. **Select Project Name and Folder:** この例では「Project Folder」はmyOwnCodeWithItsOwnMakefileと同じフォルダに作成します。従って、「Project Location」はC:\Users\MCHP\MPLABXProjects\makestuffとし、「Project Name」はMPLABXProjectTとします。

図6-13　[User Makefile Project] – プロジェクト名とフォルダの選択



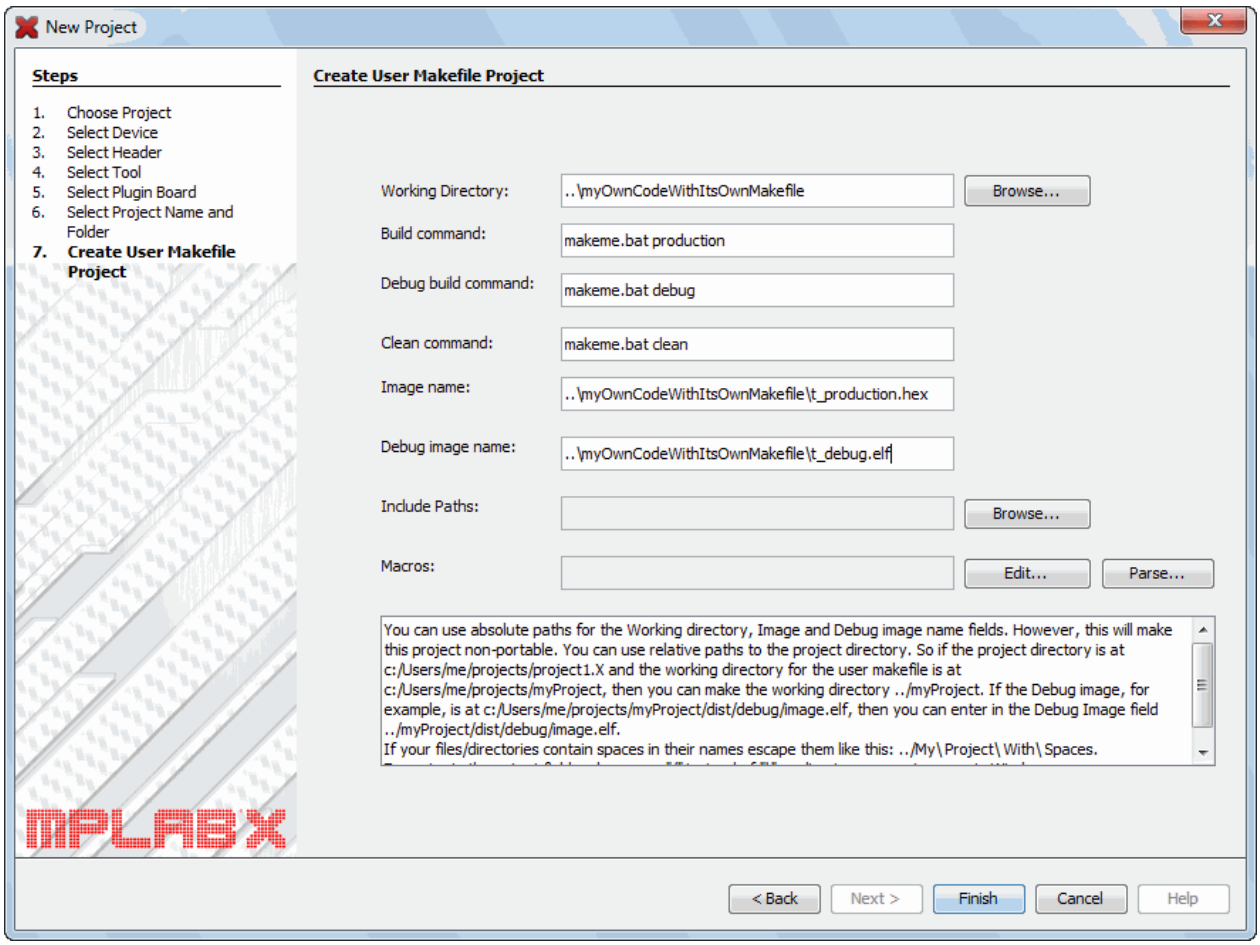
7. **Create User Makefile Project:** 情報を入力してmakefileプロジェクトを設定します。
作業フォルダでは、各種コマンド (バッチファイルのビルド、デバッグビルド、クリーン)が実行されます。このフォルダは絶対パスとして設定できます。

   ```
   C:\Users\MCHP\MPLABXProjects\makestuff\myOwnCodeWithItsOwnMakefile.
   ```

   しかし、この方法を使うとMPLAB X IDEプロジェクトの移植性が損なわれます。従って作業フォルダ、イメージ名、デバッグイメージ名はMPLAB X IDEプロジェクト フォルダからの相対パスで入力します。

   [Finish]をクリックするとプロジェクトが作成されます。

図6-14　[User Makefile Project] – バッチファイルからコードを作成



6.6.3.2　[User Makefile Project] – ユーザMakefileからコードを作成

コードをコンパイルするため、ユーザmakefile (`Makefile`)は引数として`production`、`debug`、`clean`のいずれかを使います。`production`の場合、`t_production.hex`がビルドされます。`debug`の場合、`t_debug.elf`がビルドされます。`clean`の場合、`.elf`と`.hex`ファイルが消去されます。

例: ユーザMakefileのコード

```
# Assumes xc32gcc and xc32bin2hex are on the path
COMPILER=xc32gcc
BIN2HEX=xc32bin2hex
PROCESSOR=32MX795F512L
production: t_production.hex
debug: t_debug.elf
t_debug.elf: t.c
$(COMPILER) mprocessor=$(PROCESSOR) -g -D_DEBUG -o t_debug.elf
t.c
t_production.hex: t_production.elf
$(BIN2HEX) t_production.elf
t_production.elf: t.c
$(COMPILER) mprocessor=$(PROCESSOR) o t_production.elf t.c
clean:
del /F *.hex
del /F *.elf
```

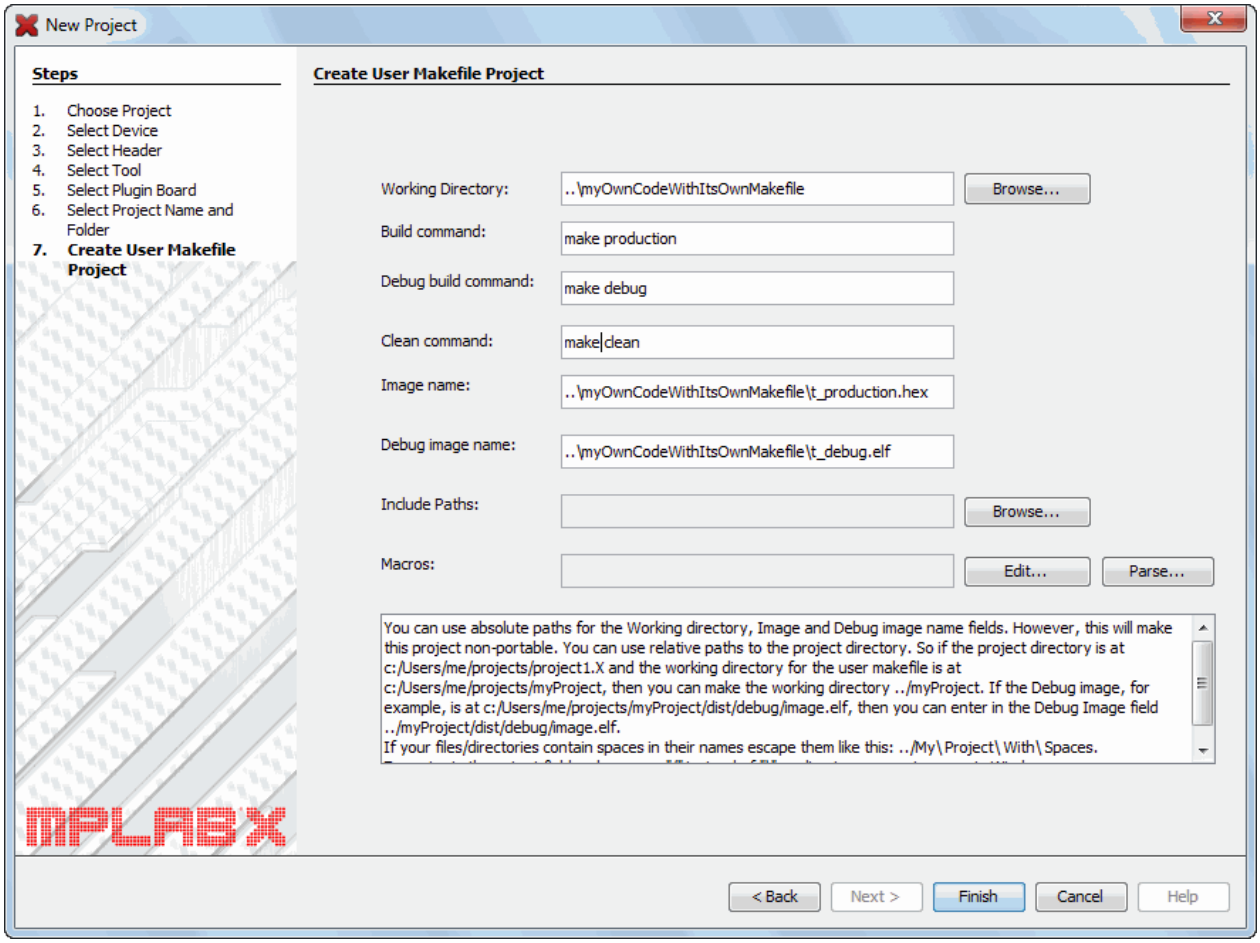
MPLAB X IDEでユーザmakefileプロジェクトを作成するには、「[User Makefile Project] – バッチファイルからコードを作成」の手順1～6を実行します。Step 7は下図に従って設定します。

作業フォルダでは、各種コマンド (ユーザMakefileのビルド、デバッグビルド、クリーン)が実行されます。作業フォルダ、イメージ名、デバッグイメージ名はMPLAB X IDEプロジェクト フォルダからの相対パスで入力します。

MPLAB X IDEはビルド、デバッグビルド、クリーンにGNU Makeを使います。「ユーザMakefileプロジェクト フォルダおよび作業フォルダ」に従って、MPLAB X IDEがプロジェクト フォルダ内でMakefileを実行すると、作業フォルダがカレントフォルダになります。作業フォルダ内でmakeはmakefileを検索し、ユーザ定義のMakefileを見つけます。このMakefileはビルド、デバッグビルド、クリーンの方法を定義します。

Note: 「Build command」等のコマンドを定義するもう1つの方法は、-fオプションを使ってユーザmakefileを指定する事です(例: `make -f Makefile production`)。**[Finish]**をクリックするとプロジェクトが作成されます。

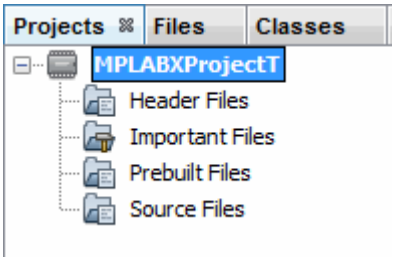
図6-15 [User Makefile Project] – ユーザMakefileからコードを作成



6.6.3.3 [User Makefile Project] – 完了

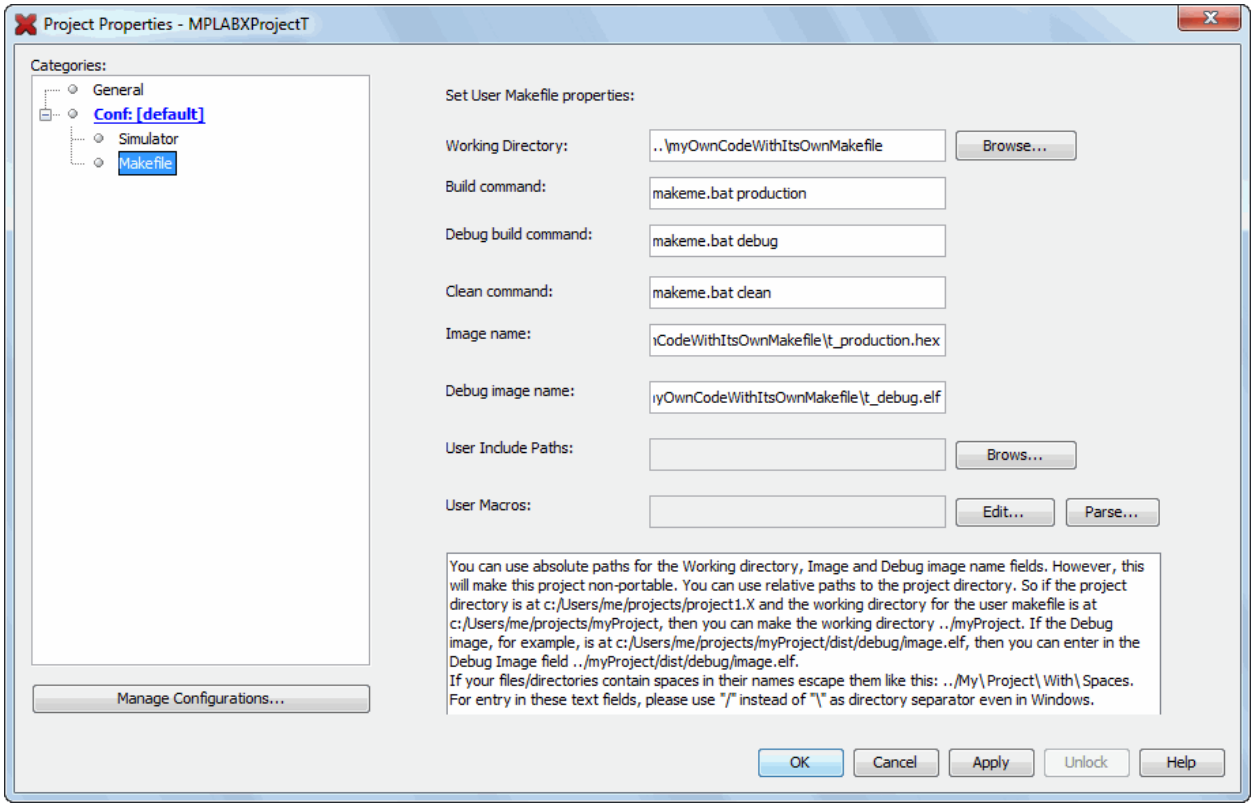
プロジェクト作成後、[Projects]ウィンドウに以下が表示されます。プロジェクト名を右クリックしてプロジェクトをビルドできます。

図6-16　[User Makefile Project] – プロジェクト ツリー



設定を変更する場合、プロジェクト名を右クリックして[Properties]を選択します。

図6-17　[User Makefile Project] – [Project Properties]ダイアログ



この例では、ビルドの結果(`t_production.hex`)とデバッグビルドの結果(`t_debug.elf`)は作業フォルダ`myOwnCodeWithItsOwnMakefile`に保存されています。MPLAB X IDEではMPLABXProjectTプロジェクト フォルダのみが表示されるため、ビルド後のIDEの[Projects]ウィンドウでは変化が見られません。

6.6.3.4　[User Makefile Project] – マクロの解析

MPLAB XC Cコンパイラのマクロを解析し、makefileプロジェクトに追加できます。これを行うには、引数を指定してコマンドラインでコンパイラを呼び出し、出力全体を[Parse macros]ダイアログの最初のパネルにコピーする必要があります。

コマンドラインでコンパイラを実行するにはCファイルが必要です。この例では、以下のコードをファイル`xcmain.c`に貼り付けます。

```
#include <xc.h>
void main(void)
{ return;
}
```

コンピュータがMPLAB XCコンパイラへのパスを認識している事を確認します。その後、コマンドラインで以下のコマンドを実行します。

```
xc32-gcc -dM -E xcmain.c > macros32.txt
```

MPLAB XC32++ではなくMPLAB XC32 Cコンパイラを使うため、xc32-gccが実行ファイルです。オプション`-dM`は、前処理の最後に有効なマクロ定義のリストだけを出力するようプリプロセッサに指示します。オプション`-E`は、前処理後に実行を停止させます。リダイレクト(>)文字は、出力をファイル`macros.txt`に書き込みます。

その他のMPLAB XC Cコンパイラでは以下を使えます。

```
xc16-gcc -dM -E xcmain.c > macros16.txt
xc8 --pre -v --chip=8BitMCU xcmain.c > macros8.txt
```

`8BitMCU`は使用中の8ビットデバイスに置き換えます。`macros8.txt`ファイルに対しては、マクロをファイル内の他のテキストから分離するための編集も必要です。

[Parse macros]ダイアログは以下の手順で開きます。

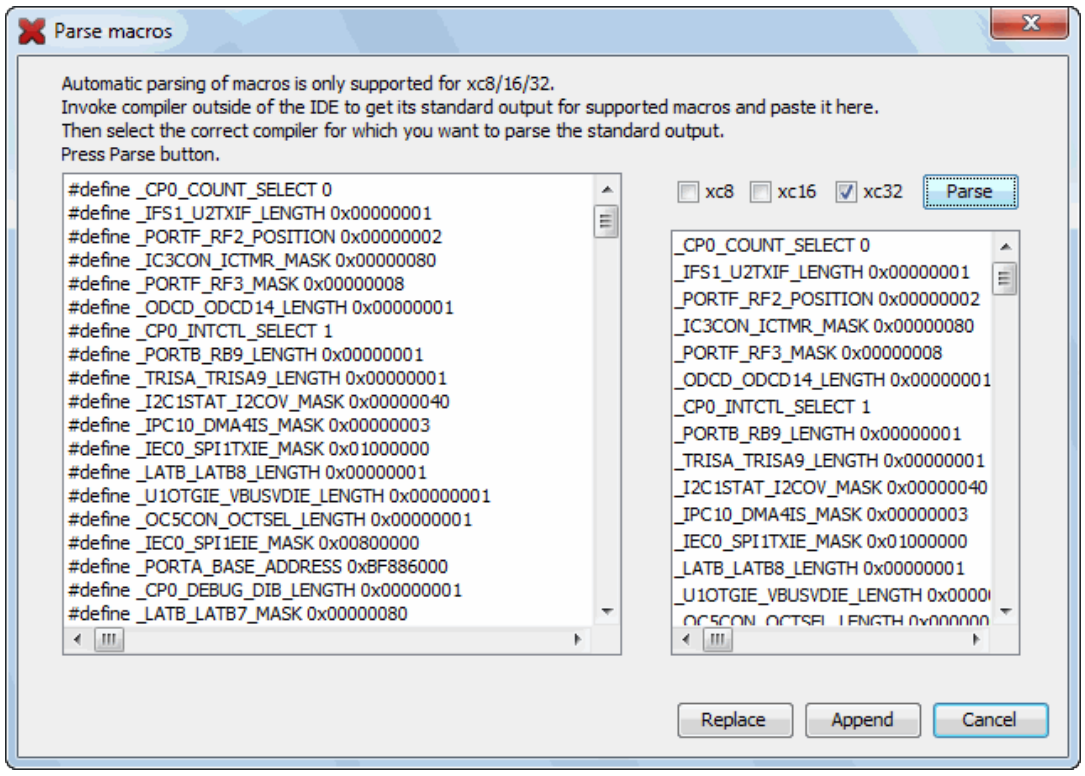
1. プロジェクト名の上で右クリックして[Properties]を選択します。
2. [Categories]の下で[Makefile]を選択します。
3. [Set User Makefile properties]の下で[User Macros]に移動し、**[Parse]**ボタンをクリックします。マクロの解析

は以下の手順で行います。

1 `macros.txt`の内容をコピーしてダイアログの左パネルに貼り付けます。
2 使用中のMPLAB XC Cコンパイラにチェックを入れる(この場合は「xc32」)。
3 **[Parse]**ボタンをクリックします。

[User Macros]テキストボックスの現在の内容を置換するには**[Replace]**、追加するには**[Append]**、変更しない場合は**[Cancel]**ボタンをクリックします。

図6-18 [Parse Macros]ダイアログ



6.7 ソースファイル フォルダにリンクしたリソースの使用

マクロを使ってソースベース(またはライブラリ バージョン)をポイントし、別のソースベースに移動するように切り換えられます。これは、MPLAB Harmonyアプリケーションで特に便利です。

MPLAB X IDEでプロジェクトを開き、[File] > [Project Properties]を選択します。[General]カテゴリをクリックし、[Macro]の横の[Browse]をクリックします。ソースベースを保存しているフォルダを見つけて開きます。**[OK]**をクリックします。プロジェクトでは、このソースベースのファイルを「Relative to Macro」として開きます。

別のソースベースに切り換えるには、その新しいソースベースを保存しているフォルダを[Macro]で参照して選択します。新しいソースベースと一致する古いソースベースのファイルは全て、新しいソースベースを参照するようになります。

6.8 サードパーティ製ハードウェア ツールの使い方

サードパーティ製ハードウェア ツールを使う場合、以下の要件があります。

- 必要なUSBドライバを入手してインストールします。USBドライバとインストール説明書は、通常サードパーティから入手できます。
- このツールのMPLAB X IDEプラグインをインストールします。詳細は5.26「プラグインツールの追加」を参照してください。

インストールが完了すると、[Project Properties]ウィンドウにハードウェア ツールが表示され選択できるようになります(4.4「プロジェクト プロパティの表示と変更」参照)。

6.9 コードカバレッジの使用

コードカバレッジは、コードのどの部分が実行されているかを表示します。コードカバレッジの出力を表示するには、コードカバレッジをサポートしているMPLAB XC CコンパイラをMPLAB X IDEで使う必要があります。コードカバレッジは以下のバージョン以降でサポートしています。

- MPLAB X IDE v5.25
- MPLAB XC8 v2.10
- MPLAB XC16 v1.40
- MPLAB XC32 v2.30

MPLAB X IDEでは、コンパイラ カテゴリの下の[Project Properties]ウィンドウでコードカバレッジを有効にできます。有効にしたら、テストコードを完全に実行して停止/一時停止します。[Window] > [Debugging] > [Code Coverage]を選択し、コード全体に対して実行されたコードの割合を**[Code Coverage]**タブウィンドウで確認します。

MPLAB Code Coverageアドオン ライセンス(SW006026- COV)を購入すると、コードカバレッジの詳細を表示できます。このライセンスを購入すると、コンパイラのフルコード カバレッジ機能がアクティベートされ、以下が表示されます。

- コードカバレッジを表示したテキスト
- [Code Coverage]タブウィンドウでの詳細レポート

詳細は以下を参照してください。

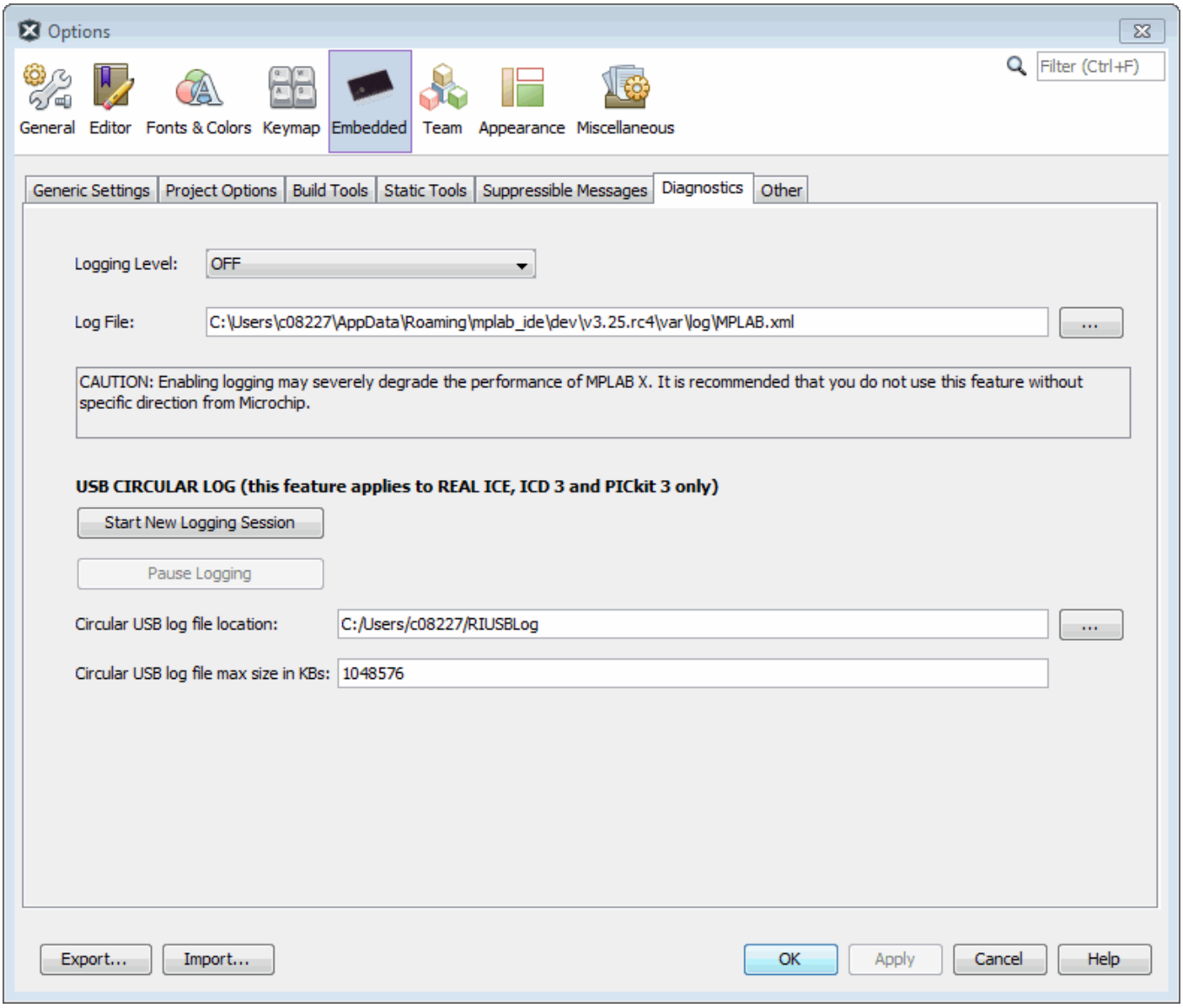
http://www.microchip.com/mplab/codecoverage

6.10 ログファイル

ログファイルは、プログラム実行状況の把握と問題の解決に役立ちます。

MPLAB X IDEに問題やエラーが発生して技術サポートに問い合わせる場合、2つのログファイル(MPLAB X IDEのログファイルとNetBeansプラットフォームのログファイル)のデータが必要です。その他の補足情報も必要です。

図6-19　データロギング



6.10.1　MPLAB® X IDEのログファイル

MPLAB X IDEのログファイル機能は、Java Loggerクラスに基づいてデータを生成します。

ログファイルの設定方法

1. [Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択して**[Embedded]**ボタンをクリックし、**[Diagnostics]**タブを選択します。
2. ドロップダウン ボックスからログレベルを選択します。
   Note:　ログレベルが高いほど収集データは増えますが、アプリケーション実行速度は低下します。
3. ログファイルを保存する場所(パス)を選択します。

データを記録する方法

1. ログレベル「Finest」を設定します。
2. ログファイルの名前と保存場所を書き留めます。
3. 問題やエラーが発生するまでステップを繰り返します。
4. 生成されたログファイルとその他必要な情報を技術サポートに送信します。

6.10.2 NetBeansプラットフォームのログファイル

NetBeansのログファイルは、NetBeansプラットフォームの実行に関する情報を記録します。

データを記録する方法

1. [View] > [IDE log]を選択して[Output]ウィンドウにログファイルを開きます。
2. [Output]ウィンドウを右クリックして[Clear]を選択します。
3. 問題やエラーが発生するまでステップを繰り返します。
4. ウィンドウを右クリックし、[Save As]を選択してテキストをファイルに保存します。
5. ファイルとその他必要な情報を技術サポートに送信します。

6.10.3 その他のサポート情報

ログは有益ですが、技術サポートでは問題の解決にあたりその他の情報も必要です。例えば以下のような情報です。

- ログファイルに対応する、実際に実行したステップ
- デバッグツールおよび言語ツールの種類とバージョン等、ツールの詳細
- その他のMPLAB X IDEプロジェクト データ

6.10.4 ログ収集に関する注意事項

ログ収集時は以下に注意します。

- ログファイルはできるだけ短く、焦点を絞ります。つまり、エラーが発生するまで延々とコードを実行して長時間記録せず、エラー発生から記録するようにします。
- ハードウェア ツールの場合、USB通信の問題を判別するには循環ログが有益です。

6.11 MPLAB X IDEプロジェクトのパッケージ化

プロジェクトを右クリックして[Package]を選択すると、プロジェクトをzipファイル(`.zip`)にパッケージ化できます。MPLAB X IDEはファイルをZIPに圧縮できますが、解凍する事はできません。従って、プロジェクトの解凍には別のプログラムが必要です。

この機能は、プロジェクトをパッケージする際にプロジェクト ファイルを解析し、パッケージに含めるべきプロジェクト ファイルのフォルダを決定します。すなわち、プロジェクト フォルダと、その相対パスを使うフォルダ内のファイルだけをパッケージに含めます。

パッケージにはソースファイル、`makefile`、「`nbproject`」フォルダが含まれます。

ヘッダファイルは、プロジェクトに追加していない限り含まれません。従って、ユーザ作成のヘッダファイルが必要な場合、解凍済みのプロジェクトはビルドできません。

6.12 ハードウェア ツールの接続とデバッグ

ハードウェア デバッグツールとターゲット間の接続によって、利用可能なデバッグ機能とデバイス関連のデバッグ機能が決まります。

ハードウェア ツールとターゲットの接続 - PIC MCU

表6-4 ハードウェア ツールとターゲットの接続 - PIC MCU

接続	ハードウェアのサポート(1)	デバッグのサポート (2,3,4)	トレースのサポート(1)	サポート速度
標準(ICSP)通信	**Real ICE**, **ICD4**, **ICD3**: モジュラケーブル(6ピンRJ-11) **Snap**, **PK4**, **PK3**: リボンケーブル(6ピンSIL)	Basic, Advanced	**Real ICE**: ネイティブトレース	15 MIPS以下 PIC32: デバイス依存

(続き)

接続	ハードウェアのサポート(1)	デバッグのサポート(2,3,4)	トレースのサポート(1)	サポート速度
高速(LVDS)通信	**Real ICE**: パフォーマンス パック (AC244002)	Basic, Advanced	**Real ICE**: ネイティブ トレース **Real ICE**: SPIトレース	15 MIPS超 PIC32: デバイス依存
	Real ICE: パフォーマンス パック + アイソレータ ユニット(AC244005)	Basic, Advanced	なし	15 MIPS超 PIC32: デバイス依存
JTAG	**Real ICE**: JTAGアダプタ (AC244007) **SJL**	Basic	なし	PIC32: デバイス依存
ロジックポート	**Real ICE**: ロジックポート プローブ(4ピン)	N/A	**Real ICE**: I/Oポート トレース	デバイス依存
	Real ICE: トレース インターフェイス キット (AC244006)	N/A	**Real ICE**: 命令トレース (PIC32 MCU、一部のエミュレーション ヘッダ)	デバイス依存
標準(ICSP)通信 + ロジックポート	**Real ICE**: 電力モニタ(AC244008)	Basic	なし	15 MIPS以下 PIC32: デバイス依存

1. ツール略語については最後の表を参照してください。
2. サポートはデバイス依存です。デバイスによっては使えないデバッグ機能があります。
3. 基本的なデバイスサポート(Basic): Run、Halt、Reset、Step Into、Step Over、ソフトウェア/ハードウェア ブレークポイント
4. 高度なデバイスサポート(Advanced): データキャプチャ、ランタイム ウォッチ等

表6-5　ハードウェア ツールとターゲットの接続 - AVR MCU

接続	ハードウェア サポート(1,2)	デバイスのサポート(3)	デバッグのサポート(4,5,6)	トレースのサポート	サポート速度
JTAG	AI、PK4	全AVR MCU	Basic	なし	32 kHz～7.5 MHz
aWire	AI	AVR 32ビットMCU	Basic	なし	7.5 kbit/s～7 Mbit/s
PDI 2線インターフェイス	AI、PK4	AVR XMEGA® MCU	Basic	なし	32 kHz～7.5 MHz
debugWIRE	AI, PK4	AVR 8ビットMCU	Basic	なし	4 kbit/s～0.5 Mbit/s
UPDI	AI, PK4	AVR 8ビットMCU	Basics	なし	最大750 kbit/s
SPI	AI, PK4	AVR 8ビットMCU	プログラミングのみ	N/A	8 kHz～5 MHz
TPI	AI, PK4	tinyAVR 8ビット MCU	プログラミングのみ	N/A	デバイス依存

(続き)

接続	ハードウェア サポート (1,2)	デバイスのサポート (3)	デバッグのサポート (4,5,6)	トレースのサポート	サポート速度

1. ツールの略語については表6-7を参照してください。
2. ICD4とPK4にはAC102015デバッガ アダプタボードが必要です。
3. デバイスサポートは現在進行中であり、現時点で利用できない場合があります。
4. サポートはデバイス依存です。デバイスによっては使えないデバッグ機能があります。
5. 基本的なデバイスサポート(Basic): Run、Halt、ソフトウェア/ハードウェア ブレークポイント
6. プログラミングのみ: この接続ではデバッグできません。

表6-6　ハードウェア ツールとターゲットの接続 - SAM MCU

接続	ハードウェア サポート (1,2)	デバイスのサポート (3)	デバッグのサポート (4,5,6)	トレースのサポート	サポート速度
JTAG	ICD4	SAM MCU	Basic	なし	32 kHz～7.5 MHz
SWD	ICD4	SAM MCU	Basic, Advanced	ITM - 3 MB/s	32 kHz～10 MHz

1. ツールの略語については表6-7を参照してください。
2. ICD4とPK4にはAC102015デバッガ アダプタボードが必要です。
3. デバイスサポートは現在進行中であり、現時点で利用できない場合があります。
4. サポートはデバイス依存です。デバイスによっては使えないデバッグ機能があります。
5. 基本的なデバッグサポート(Basic): Run、Halt、ソフトウェア/ハードウェア ブレークポイント
6. 高度なデバッグサポート(Advanced): Basicへの追加機能、デバイス依存

表6-7　ツールの略語

略語	ツール名
Snap	MPLAB Snapインサーキット デバッガ/量産プログラマ
PK4	MPLAB PICkit 4インサーキット デバッガ/量産プログラマ
PK3	PICkit 3インサーキット デバッガ/開発用プログラマ
ICD4	MPLAB ICD 4インサーキット デバッガ/量産プログラマ
ICD3	MPLAB ICD 3インサーキット デバッガ/量産プログラマ
RI	MPLAB REAL ICEインサーキット エミュレータ/量産プログラマ
AI	Atmel-ICEインサーキット デバッガ/プログラマ
SJL	SEGGER J-Linkデバッグプローブ (サードパーティ)

デバイス依存デバッグ機能の詳細は以下を参照してください。

`<MPLAB X IDE install>docs/FeatureSupport/HWToolDebugFeatures.html`

6.13　IDEスクリプトの使用

MPLAB X IDEにはIDEの挙動をプログラミングにより制御できるスクリプト機能があります。Microchip社のMPUプロジェクトで使えるように開発すると、MCUプロジェクトでも使えます。

スクリプトについて

スクリプトはJythonで書かれています。スクリプトはAPIへアクセスでき、以下が可能です。

- デバッグ セッションの状態(実行、停止等)を制御

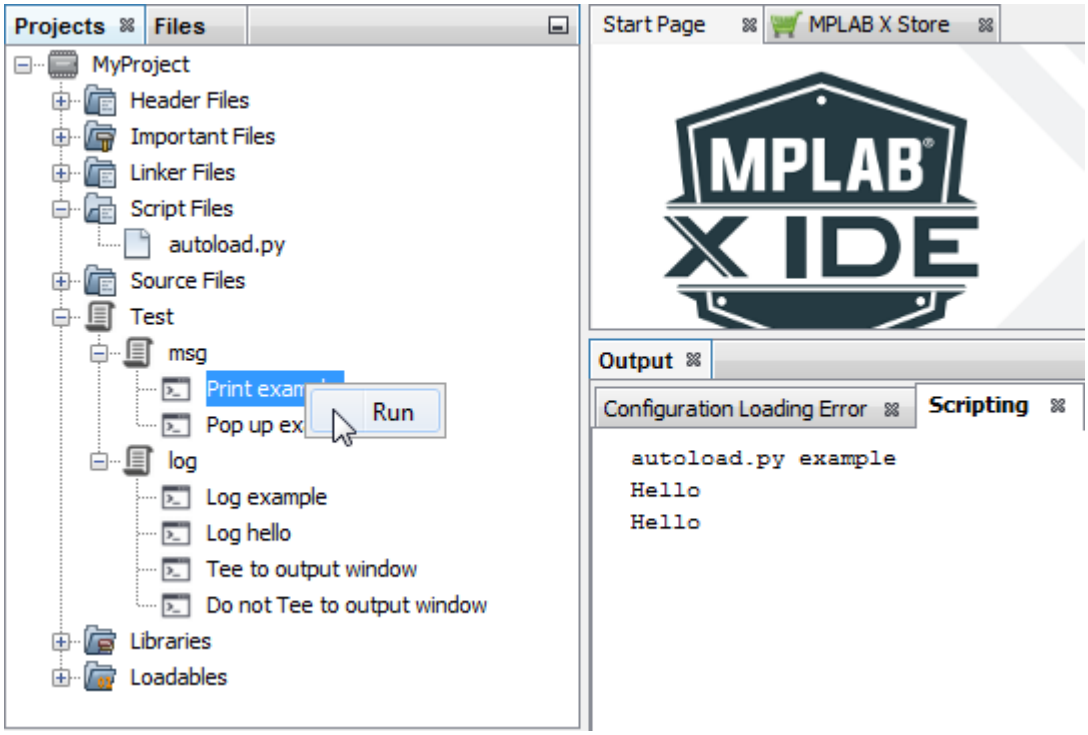
- デバッグ中にデバイスメモリへアクセス
- 停止を発生させるかどうかを決定する関数へのコールバックを使ってブレークポイントを設定(これにより、条件付きブレークポイントを実装可能)
- [Projects]ウィンドウに操作を配置(クリックするとスクリプトのメソッドを実行)
- デバッグイベントのフックを提供(特定のイベントが発生したら関数を呼び出す)

スクリプトの使い方

スクリプト機能へは以下の手順でアクセスします。

1. スクリプトを使うプロジェクトを閉じます。
2. `autoload.py`という名前のファイルを作成します。APIを説明する自己文書化`autoload.py`は、`<MPLAB X IDE Install folder>\docs\example_code`に保存されています。
3. このファイルをプロジェクト フォルダにコピーします。必要に応じてファイルを編集します。
4. プロジェクトを再度開くと、プロジェクトツリーに「Script Files」フォルダが表示されます。ファイルautoload.pyはこのフォルダにあります。
5. 必要に応じて「Tests」のスクリプトを実行します。

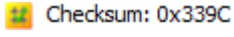
図6-20　プロジェクト内の未編集autoload.py



6.14　チェックサム

チェックサムとは、アプリケーション コードと、MCUまたはDSCの設定から得られる16進数です。チェックサム生成方法の詳細はデバイスのプログラミング仕様を参照してください。

チェックサムはプロジェクトのビルド時に生成されます。生成されたチェックサムは[Dashboard]ウィンドウに表示されます(5.19「ダッシュボードの表示」)。



チェックサムはビルドエラーの検出に役立ちます。例として、あるプロジェクトをあるコンピュータでビルドした場合について考えます。同じプロジェクトを別のコンピュータにコピーしてビルドした時に、チェックサムが同じであればプロジェクトが正常にコピーされた事を意味します。チェックサムが異なれば、コピーに失敗した事を意味します。

チェックサムをエラー検出に使う場合、プロジェクト コードを開発する際以下に注意する必要があります。コンパイラマクロの__TIME__と__DATE__を使ってはいけません。ファイルパスを変更する場合、マクロの__FILE__とassert()を使ってはいけません。

デバイスによって、チェックサムをユーザIDとして使う場合があります。
4.10.3「ユーザIDメモリへの非保護チェックサムの挿入」

6.15 コンフィグレーション

「コンフィグレーション」という用語は様々な文脈で使われます。下表にそれらの用語と定義を示します。

表6-8 コンフィグレーションの文脈

用語	定義
コンフィグレーション ビット コンフィグレーション ヒューズ コンフィグレーション ワード コンフィグレーション バイト コンフィグレーション レジスタ	これらの用語は全て、デバイスを設定するために使うデバイスメモリの物理領域を指します。利用可能なコンフィグレーションは、デバイスのデータシートを参照してください。
[Configuration Bits]ウィンドウ	プロジェクトで設定したデバイスで利用可能なコンフィグレーションを表示するウィンドウです。このウィンドウの詳細は、4.19「[Configuration Bits]ウィンドウでのコンフィグレーション値の設定」を参照してください。
プロジェクト コンフィグレーション	プロジェクトのビルド設定です。プロジェクト作成後、コンフィグレーション名は「[default]」となっています。この名前は変更できます。また、1つのプロジェクトに対して複数のコンフィグレーションを作成できます。詳細は6.4「複数コンフィグレーションの使い方」を参照してください。 このデータはconfigurations.xmlファイルに保存されます。
プロジェクト コンフィグレーション タイプ	特定タイプのプロジェクト向けの設定です。プロジェクト コンフィグレーション タイプは、「アプリケーション」として設定したスタンドアロンプロジェクトと「ライブラリ」として設定したライブラリ プロジェクトにのみ適用されます。詳細は4.10.1「プロジェクト コンフィグレーション タイプの変更」を参照してください。
ユーザ コンフィグレーション データ	ユーザが作成したMPLAB X IDEのコンフィグレーションです。8.7「ユーザ コンフィグレーション データの表示」を参照してください。

7. エディタ

MPLAB X IDEはNetbeansプラットフォームをベースとしているため、コードの編集にはNetBeansプラットフォームのエディタを使います。

NetBeansソースエディタの詳細は以下を参照してください。

https://docs.oracle.com/cd/E50453_01/doc.80/e50452/working_nbeans.htm#A1151635

エディタに関するCコンパイラ情報は、以下のNetBeansヘルプトピックでご覧ください。

https://netbeans.org/kb/docs/cnd/navigating-editing.html

7.1 エディタの使い方

エディタを使うには、[File] > [New File]を使ってファイルを作成します。または[File] > [Open File]を使って既存のファイルを開きます。以下のセクションでは、その他の操作と機能について説明します。

デスクトップの制御

エディタに関連するデスクトップ項目には以下のようなものがあります。

- [File]メニュー – [Editor]ウィンドウでファイルを開きます(10.1.1「[File]メニュー」参照)。
- [Edit]メニュー – 編集コマンドにアクセスします(10.1.2「[Edit]メニュー」)。
- エディタ ツールバー – こちらも編集コマンドにアクセスします。各ファイルのエディタ ウィンドウの上にあります(10.2.10「エディタ用ツールバー」参照)。
- ウィンドウ内で右クリックして開くコンテクスト メニュー – その他のコマンドを利用できます。

表示の制御

下表に基本的な表示の制御機能を示します。その他のフォーマット設定機能については7.2「エディタのオプション」を参照してください。

操作	説明
ソースエディタのウィンドウを最大化する	以下のいずれかを行います。 • ソースエディタのファイルのタブをダブルクリックする • ソースエディタのウィンドウを選択し、<Shift>+<Esc>を押す • [Window] > [Configure Window] > [Maximize]を選択する
最大化したソースエディタを元のサイズに戻す	以下のいずれかを行います。 • ソースエディタのファイルのタブをダブルクリックする • <Shift>+<Esc>を押す • [Window] > [Configure Window] > [Restore]を選択する
行番号を表示する	[View] > [Show Line Numbers]を選択する
2つのファイルを同時に表示する	1. 2つ以上のファイルを開く 2. 1つのファイルのタブをクリックし、ファイルを配置するウィンドウの側面にドラッグする ウィンドウの配置場所を示す赤いプレビュー ボックスが表示された状態でマウスボタンを放すと、エディタ ウィンドウがそこに配置されます。 タブをドラッグする場所によって、ウィンドウは水平または垂直に分割できます。
1つのファイルの表示画面を分割する	1. ソースエディタのファイルのタブを右クリックし、[Clone]を選択する 2. コピーしたファイルのタブをクリックし、コピーしたファイルを配置するウィンドウの部分にドラッグする

(続き)

操作	説明
開いているファイル間を移動する	1. <Ctrl>を押す 2. <Ctrl>を押したままで<Tab>を押すと、選択ボックスが表示される 3. 移動するファイルを選択する
コードを自動的にフォーマットする	ソースエディタ内で右クリックし、[Format]を選択する テキストを選択している場合、そのテキストのみがフォーマットされます。テキストを選択していない場合、そのファイル全体がフォーマットされます。

基本的なエディタ機能

下表に、頻繁に使われるエディタ機能をまとめます。

エディタ機能	参考文献
Unicodeのサポート	既定値ではISO-8859-1文字エンコードを使います。これを変更するには以下を行います。 [Projects]ウィンドウでプロジェクト名を右クリックして[Properties]を選択する 左側の[Categories]の下の列で[General]を選択する ページ下部の[Encoding]を変更する
構文に基づくコードの色分け	[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])、**[Fonts and Colors]**ボタン、**[Syntax]**タブ 色分けは、プロジェクト作成時のエンコード設定で決まります。
コード入力中のエラー表示	以下のNetbeansヘルプトピックを参照してください。 https://netbeans.org/kb/docs/java/editor-codereference.html#coloring
シンボル、エラー等の色付きマーカによる識別	[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])、**[Fonts and Colors]**ボタン、**[Annotations]**タブ
候補とヒントを表示する スマートコード補完	[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])、**[Editor]**ボタン、**[Code Completion]**タブ
デバッグモードでシンボルにカーソルを合わせて値を表示する	デバッグ停止時にシンボルの上にマウスカーソルを置くと値が表示されます。 **Note:** この方法でマクロの値を表示する事はできません。デバッグ中にエディタ内のマクロを右クリックして(macOSでは<Cmd>+<Alt>+<Click>)、[Macro Expansion]ウィンドウを使います。
アセンブリとCコードの折り畳みおよび展開表示	7.5「コード折り畳み」を参照してください。
関数(例: `delay(x)`)を右クリックして、その関数が使われている箇所を検索する。検索を関数内のみに制限する事もできます(例: ローカル変数`i`の検索)。	以下のNetBeansヘルプトピックを参照してください。 http://wiki.netbeans.org/Find_Usages_in_Compiled_Dependencies
関数(例: `delay(x)`)を右クリックして、コールグラフを表示する コールグラフの横のボタンを使って順番の入れ換え等を行う	「コールグラフの表示」を参照してください。
ソースコード内でキーワード(例: `//TODO`)を使ってコメントを作成し、それらの要処理事項をスキャンして[Action Items]ウィンドウに自動的に追加する	以下のNetBeansヘルプトピックを参照してください。 https://netbeans.org/features/ide/index.html [Options]ウィンドウに関しては、[Team]: [Action Items]を参照してください。

(続き)

エディタ機能	参考文献
バージョン管理システムがなくてもローカルファイル履歴機能を使って以前のバージョンを復元する	5.22「ローカルファイル履歴によるソースコードの管理」
[Go to File]、[Go to Type]、[Go to Symbol]、[Go to Header]、[Go to Declaration]等を使って、素早く移動する	「[Navigate]メニュー」を参照してください。
リファクタリング オプション(関数と変数の名前変更、全関数の検索等)を使ってコードの構造を改良する	7.6「Cコードのリファクタリング」を参照してください。

エディタのトラブルシュート

エディタ内でハイライトされたコードのエラーは、コンパイラのマニュアルを参照してください。エラーのハイライト表示に関する詳細は、NetBeansヘルプトピックを参照してください。

https://netbeans.org/kb/docs/java/editor-codereference.html#coloring

個々のエラーは9.5「エラー」に記載しています。

7.2 エディタのオプション

エディタ オプションの設定方法

1. [Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択して[Options]ダイアログを開きます。
2. **[Editor]**ボタンをクリックします。タブをクリックしてエディタ機能

を設定します。下表に各タブとそのオプションを示します。

表7-1 [General]タブ

項目	説明
Code Folding	コード折り畳みを有効にする場合にチェックを入れます。また折り畳むコードの種類を選択します。コード折り畳みの詳細は7.5「コード折り畳み」を参照してください。 以下のNetBeansヘルプトピックも参照してください。 https://netbeans.org/kb/docs/java/editor-codereference.html
Camel Case Behavior	キャメルケース ナビゲーションを有効にする場合にチェックを入れます。

表7-2 [Formatting]タブ

項目	説明
Language	フォーマット設定を適用するプログラミング言語を選択します。
Category	カテゴリを選択します。現在は[Tabs and Indents]のみです。
Expand Tabs to Spaces	タブをスペースに変換する場合、チェックを入れます。
Number of Spaces per Indent	インデント1段をスペース何文字にするか入力します。
Tab Size	タブのサイズを入力します。
Right Margin	右余白のサイズを入力します。

表7-3　[Code Completion]タブ

項目	説明
Language	コード補完機能を適用するプログラミング言語を選択します。
Completion options	コード補完機能を有効にする場合、チェックを入れます。コード補完の詳細は以下のNetBeansヘルプトピックを参照してください。 https://netbeans.org/kb/docs/java/editor-codereference.html

表7-4　[Code Templates]タブ

項目	説明
Language	テンプレートを適用するプログラミング言語を選択します。
Templates	上で指定した言語に対するテンプレート情報を入力します。 **Abbreviation**: エディタに入力する省略形を入力します。 **Expanded Text**: 上の省略形と[Expand template on]で選択したキーを入力すると、ここで指定するテキストに展開します。 **Description**: テンプレート項目に説明を追加します(任意)。コード テンプレートの詳細は以下のNetBeansヘルプトピックを参照してください。 https://netbeans.org/kb/docs/java/editor-codereference.html
Expand template on	省略形を展開するトリガとなるキーを選択します。

表7-5　[Hints]タブ

項目	説明
Language	ヒントを適用するプログラミング言語を選択します。
[Hint]ウィンドウ	[Hint]ウィンドウでは、エラー、警告等どれをどのように表示するかを指定します。 ヒントの使い方は以下のNetBeansヘルプトピックを参照してください。 https://netbeans.org/kb/docs/java/editor-codereference.html

表7-6　[Macros]タブ

項目	説明
Macros	エディタマクロを作成、編集、削除します。 **マクロを新規作成する**には、**[New]**ボタンをクリックしてマクロ名を入力します。その後、リストでそのマクロを選択し、マクロコード エディタでコードを引用符で囲んで入力します。 **マクロのキーボード ショートカットを設定する**には、リストからマクロを選択し、**[Set Shortcut]**ボタンをクリックし、ダイアログ ボックスにショートカットを入力します。このショートカットを使ってマクロを実行します。 **マクロを削除する**には、リストからマクロを選択して**[Remove]**ボタンをクリックします。 **マクロコードを変更する**には、リストからマクロを選択してエディタでコードを編集します。
Macro Code	上のマクロ名をクリックして、マクロコードを引用符で囲んで入力します。マクロのキーワードの一覧は、以下のURLを参照してください。 http://wiki.netbeans.org/FaqEditorMacros **Note:** 通常、エディタでゼロから入力するよりも、操作を記録してマクロとして追加する方が簡単です。マクロの記録には、[Edit] > [Start/Stop Macro Recording]を使います。

7.3 コードのハイパーリンク

コードのハイパーリンクを使うと、リンク内容の関連情報に移動できます。

Cコードのハイパーリンク

ハイパーリンク ナビゲーションを使うと、関数、変数、定数のいずれかの呼び出しからその宣言にジャンプできます。

以下のどちらかの方法により、ハイパーリンクを使えます。

- <Ctrl> (Windows, Linux)または<Command> (Mac)を押しながら関数、変数、定数のいずれかの上にマウスカーソルを置きます。その要素に関する情報とハイパーリンクが表示されます。ハイパーリンクをクリックすると宣言にジャンプします。<Alt>+<左矢印>を押すと、呼び出し元に戻ります。
- マウスカーソルを識別子の上に置き、<Ctrl>+<B>または<Command>+<B>を押すと、エディタは宣言にジャンプします。<Alt>+<左矢印>を押すと、呼び出し元に戻ります。

<Alt>+<左矢印>または<Alt>+<右矢印>を押すと、カーソル位置の履歴で後方または前方に移動します。

ASMコードのハイパーリンク

アセンブリ ソースファイルに含まれているヘッダファイルを参照するには、#include文で参照されているファイル名の上にマウスカーソルを置き<Ctrl> (Windows, Linux)または<Command> (Mac)を押します。マウスをクリックすると、エディタのファイルタブ内にインクルード ファイルが開きます。

7.4 エディタの赤の感嘆符

ソースエディタは、ソースコード中の構文およびセマンティック エラー(例: セミコロンの欠落、未解決の識別子)をハイライト表示します。エラーは赤の下線で表示されます。また、エラーを含む各行の隣のエディタ ウィンドウ左側の余白に赤の感嘆符「グリフ」  が表示されます。グリフの上にマウスカーソルを置くとエラーメッセージが表示されます。

Note: エディタに表示されるエラーが全て本当のエラーとは限りません。認識できないシンボルである可能性もあります。コードをビルドしてみてビルドに成功するかどうか確かめます。詳細はセクション「未認識のシンボルによるエラー」を参照してください。

エラー条件とその他のマーク表示

エラーは、以下のいずれかを実行した場合に発生する可能性があります。

- 新規のコードを入力した
- 既存のコードを編集した
- 既存のファイルをプロジェクトに追加した
- 別の場所にファイルを移動した
- プロジェクトのプロパティまたはソースファイルを変更した

ファイル内にエラーがある場合、エディタ ウィンドウ右側のエラーストライプに赤のマークが表示されます。エラーストライプは、現在表示されている行だけではなくファイル全体を表しています。エラーストライプのマークをダブルクリックすると、マークが表す行にジャンプします。

未認識のシンボルによるエラー

コンパイラのプリパーサまたはパーサによって認識されないシンボルは、エラーとして表示されていても実際にはエラーではない(コードはビルドされる)場合があります。従って、このタイプのエラーが疑われる場合、プロジェクトのビルドを試みます。

例としては9.5「エラー」で以下を参照してください。

- For 8-bit code, #asm and #endasm cause red bangs in the Editor window.
- For 16-bit code, MPLAB XC16 packed attribute statement causes red bangs in the Editor window.

7.5 コード折り畳み

コードの一部を折り畳んで非表示にする事で、コードを読みやすくします。選択したオプションに応じて、Cまたはアセンブリコードの一部を折り畳み(非表示)または展開(表示)できます。

コード折り畳みは既定値で有効になっています。ほとんどのCまたはアセンブリコードでは、折り畳む事ができるコード部分はコード左の[-]および[+]アイコンで示されています。

コード折り畳みの有効化と設定

コード折り畳みは以下の手順で有効化および設定します。

1. [Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択して[Options]ダイアログを開きます。
2. **[Editor]**ボタンをクリックします。
3. **[General]**タブをクリックし、[Use code folding]にチェックを入れます。
4. その下で折り畳むセクションを指定します。折り畳みを実行するコードタイプが表示されない場合、カスタムの折り畳み機能を設定できます。

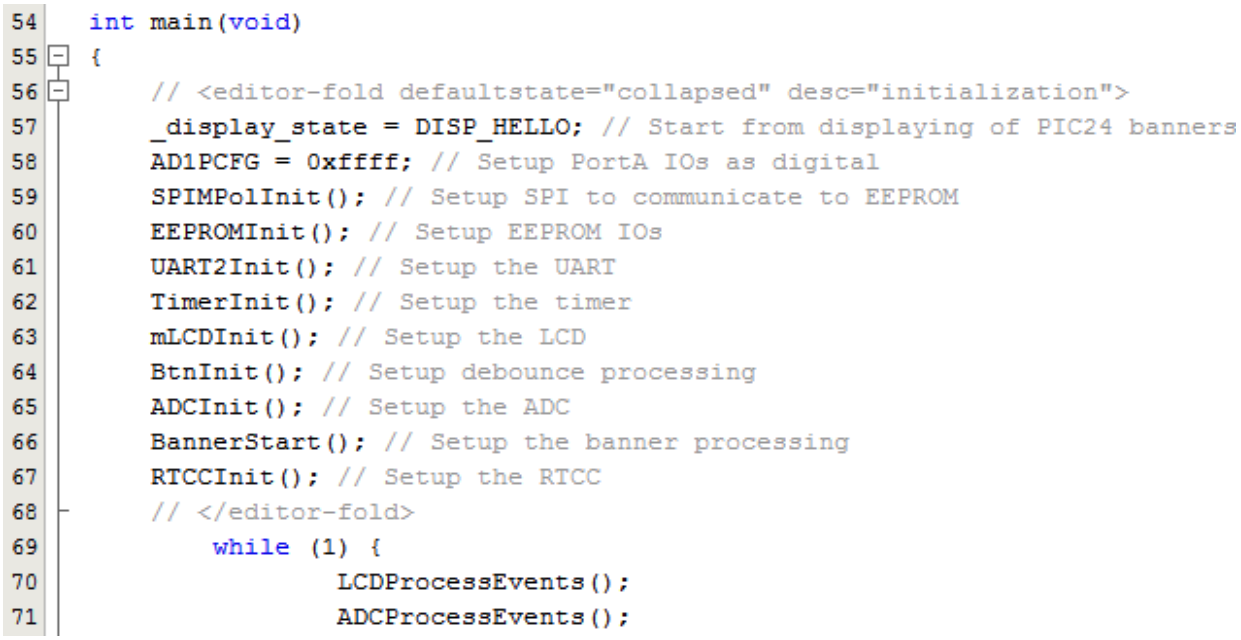
コード折り畳み/展開

エディタで、メソッドの隣の[-]アイコンをクリックすると、コードは折り畳まれます。折り畳まれたメソッドの隣の[+]アイコンをクリックすると、コードは展開されます。折り畳まれたコードは省略記号の箱で表されます。省略記号の箱の上にカーソルを置くと、IDEは折り畳まれたメソッドを表示します。

図7-1　展開されたコード

```
54  int main(void)
55  {
56      // <editor-fold defaultstate="collapsed" desc="initialization">
57      _display_state = DISP_HELLO; // Start from displaying of PIC24 banners
58      AD1PCFG = 0xffff; // Setup PortA IOs as digital
59      SPIMPolInit(); // Setup SPI to communicate to EEPROM
60      EEPROMInit(); // Setup EEPROM IOs
61      UART2Init(); // Setup the UART
62      TimerInit(); // Setup the timer
63      mLCDInit(); // Setup the LCD
64      BtnInit(); // Setup debounce processing
65      ADCInit(); // Setup the ADC
66      BannerStart(); // Setup the banner processing
67      RTCCInit(); // Setup the RTCC
68      // </editor-fold>
69          while (1) {
70                  LCDProcessEvents();
71                  ADCProcessEvents();
```

図7-2　折り畳まれたコード



コード折り畳み – インライン アセンブリとMPLAB C18 C

_asmおよび_endasmディレクティブを使っている場合、コード折り畳みはMPLAB C18のインライン アセンブリコードに対して機能しません。その回避策は、コードブロックをコード/ファイルの末尾付近に配置する事です。

コード折り畳みのカスタマイズ – C

Cコードをカスタム設定で折り畳む方法は以下の通りです。

- 折り畳むコードの前後に以下のコメントを入力します。

```
// <editor-fold defaultstate="collapsed" desc="user-description">
C code block to fold
// </editor-fold>
```

または

- `fcom`と入力して<Tab>キーを押すと、自動的に上のコメントが挿入されます。コ

メントが入力されたら、自分のコード向けにカスタマイズできます。

`defaultstate`	「`collapsed`」または「`expanded`」のどちらかを指定します。
`desc`	折り畳むコードブロックの説明を入力します。折り畳んでも、参考のためにこの説明だけは表示されたままになります。

上の「コード折り畳み/展開」に、カスタムコード折り畳み例を示しています。

詳細は以下のNetBeansヘルプトピックを参照してください。

https://ui.netbeans.org/docs/ui/code_folding/cf_uispec.html

コード折り畳みのカスタマイズ – ASM

アセンブリコードをカスタム設定で折り畳む方法は以下の通りです。

- MPASMアセンブラまたはMPLAB XC16アセンブラのコードの場合、以下のコメントをコードの前後に入力します。

```
; <editor-fold defaultstate="collapsed" desc="user-description">
ASM code block to fold
; </editor-fold>
```

```
// <editor-fold defaultstate="collapsed" desc="user-description">
ASM code block to fold
// </editor-fold>
```

- MPLAB XC32アセンブラのコードの場合、以下のコメントをコードの前後に入力します。

または

- MPASMアセンブラまたはMPLAB XC16アセンブラのコードの場合、`fcom;`と入力して<Tab>キーを押すと、自動的に上のコメントが挿入されます。
- MPLAB XC32アセンブラのコードの場合、`fcom//`と入力して<Tab>キーを押すと、自動的に上のコメントが挿入されます。

アセンブリコードは、Cコードと同じように折り畳まれます。コード折り畳みの例は、上のCコードの例を参照してください。

カスタマイズしたコード折り畳み/展開

コードブロックを展開または折り畳むには、[View]メニューを選択するか、[Editor]ウィンドウを右クリックして、以下を選択します。

1. [Code Folds] > [Collapse Fold] - コードブロックを非表示(折り畳み)
2. [Code Folds] > [Expand Fold] - コードブロックを表示(展開)
3. [Code Folds] > [Collapse All] - 折り畳み設定したコードブロックの全てを非表示
4. [Code Folds] > [Expand All] - 折り畳み設定したコードブロックの全てを表示

上の「コード折り畳み/展開」に、カスタムコード折り畳み例を示しています。

7.6 Cコードのリファクタリング

リファクタリングは、プログラムの挙動を一切変更する事なくコードを再構成します。リファクタリングによりコードが読みやすくなり、内容の理解と編集が容易になります。式をリファクタリングした場合、同じ結果が得られる必要があるのと同様に、リファクタリング後のプログラムも元のプログラムと機能的に等価である事が必要です。

一般的に、以下を目的としてコードをリファクタリングします。

- コードの編集や機能の追加を容易にする
- コードを単純化して理解しやすくする
- 不要な繰り返しを取り除く
- コードを他の用途またはより一般的な用途で使えるようにする
- コードの性能を改善する

IDEのリファクタリング機能は、ユーザが加えようとしている変更を評価してコードを再構成し、その変更によってアプリケーションのどの部分が影響を受けるのかを示します。これにより、必要な全ての箇所でコードを変更してコードを単純化できます。例えば、クラス名を変更する場合、IDEはコード内でその名前が使われている箇所を全て検出し、その名前が使われている箇所を全て変更するようユーザに提示します。

[Refactor]メニュー

IDEのリファクタリング機能は、ユーザがある箇所でコードの構造を変更した時に、その変更を反映してコード全体を更新します。

リファクタリング機能は、[Refactor]メニュー(10.1.6「[Refactor]メニュー」参照)から利用できます。

リファクタリングした変更の取り消し/やり直し

[Refactor]メニュー内のコマンドを使って適用した変更は、[Undo/Redo]ツールバー メニューのボタンで取り消す(またはやり直す)事ができます。

	**Undo** - IDEは、リファクタリングの影響を受けた全てのファイル内の全ての変更を元に戻します。

	**Redo** - IDEは、影響を受けた全てのファイル内のリファクタリングした変更を繰り返します。

リファクタリングを実行した後に、その影響を受けたファイルのいずれかを変更した場合、リファクタリングの[Undo/Redo]は利用できなくなります。

関数の使用箇所の検索

[Find Usages]コマンドを使うと、プロジェクトのソースコード内で特定の関数が使われている全ての箇所を検出できます。

プロジェクト内で関数が使われている箇所を検出する方法

1. [Navigator]ウィンドウまたは[Source Editor]ウィンドウ内で関数を右クリックし、[Find Usages]を選択します(<Alt>+<F7> (macOSでは<Ctrl>+<F7>)を押す)。
2. [Find Usages]コマンドは、指定された関数を呼び出しているコード行を表示します。[Find Usages]ダイアログ ボックス内で[Find]をクリックすると、
[Usages]ウィンドウに、その関数が見つかったファイルの名前と、そのファイル内で関数が使われているコード行が表示されます。

見つかった関数使用行のいずれかを選択して、その位置へジャンプする方法

- [Usages]ウィンドウの左ペイン内で、上向きまたは下向き矢印ボタンを使って、関数使用行を選択します。
- コードの1行をダブルクリックしてファイルを開くと、カーソルはそのコード行を指します。

その他のIDE検索機能

以下のIDEツールにより、プロジェクト内で特定テキストが使われている全ての箇所を検索できます。

- **テキストの検索と置換:** エディタで開いたソースファイル内で、特定テキストを検索するには、[Edit] > [Find]を選択して[Find]ダイアログ ボックスを開くか、[Edit] > [Replace]を選択して[Replace]ダイアログ ボックスを開きます。これらのコマンドは、文字列がCコードエレメントであるかどうかに関係なく、一致する文字列を全て検索します。
- **プロジェクト内の検索:** [Find]コマンドと同様に、[Find in Projects]コマンドも、関数名かどうかに関係なく、一致する文字列を検索します。[Edit] > [Find in Projects]を選択して[Find in Projects]ダイアログ ボックスを開き、検索文字列を入力します。

[Navigator]ウィンドウ内で関数をダブルクリックすると、ソースファイル内でその関数が宣言されている箇所を検索できます。その関数が別のソースファイル内で宣言されている場合、関数を右クリックしてコンテクスト メニューから[Navigate] > [Go To Declaration]を選択します。

関数またはパラメータの名前変更

関数またはパラメータの名前を変更し、プロジェクト全体を通してその関数またはパラメータへの参照を更新するには、以下の手順を実行します。

1. ソースエディタ内で関数またはパラメータを右クリックし、コンテクスト メニューから[Refactor] > [Rename]を選択します。
2. [Rename]ダイアログ ボックスに新しい名前を入力します。
3. 必要に応じて**[Preview]**をクリックします。すると、[Refactoring]ウィンドウ内の[Source Editor]の下段に、変更対象のコード行が表示されます。チェックを外すとその行は変更されません。
4. **[Do Refactoring]**をクリックして、変更を適用します。

関数またはパラメータにカーソルを合わせて<Ctrl>+<R>を押して新しい名前を入力すると、直接簡単に名前を変更できます。名前の変更作業を終了するには**[Escape]**を押します。

Cコードの移動、コピー、安全な削除

これらの機能はC++コード専用です。10.1.6「[Refactor]メニュー」を参照してください。

7.7 C/C++コードのエラー ディレクティブ

コードで#errorディレクティブを使う場合、以下のように文をカッコで囲む事を推奨します。

```
(#error This line shown only when there is an error.)
```

囲まれていない場合、構文パーサはCコンパイラと同じ方法で`#error`ディレクティブを扱い、そのトラックで停止します。従って、エディタではこれを示すため、そのディレクティブ直後のテキストが灰色表示されます。

8. プロジェクト ファイルおよびフォルダ

MPLAB X IDEでは通常、[File]ペイン内にプロジェクト ファイルとフォルダを操作するためのウィンドウ([Projects]、[Files]、[Favorites]、[Classes])を表示します。

本章で以下のプロジェクト ファイルおよびフォルダ関連トピックを確認する事を推奨します。

- パス/ファイル/フォルダ名の制約
- パス/ファイル/フォルダ プロジェクトに関する推奨事項
- ユーザ コンフィグレーション データの表示
- MPLAB IDE v8プロジェクトのインポート – 相対パス
- プロジェクトの移動、コピー、リネーム
- プロジェクトの削除

8.1 [Projects]ウィンドウの表示

[Projects]ウィンドウは、いわばプロジェクトの正面入口です。このウィンドウは重要なプロジェクト内容を表示します。このウィンドウの詳細は12.15「[Projects]ウィンドウ」を参照してください。

プロジェクトには少なくともソースファイルを追加する必要があります。MPLAB X IDEは、各種の既定値ファイル(ヘッダファイル、リンカスクリプト)を自動的に見つけます。ユーザはライブラリ ファイル、プリコンパイル済みオブジェクト ファイル、編集済みのヘッダおよびリンカスクリプト ファイルを追加できます。ビルドに含めないファイルは「Important Files」の下に保存できます。

図8-1 [Projects]ウィンドウ – 単純なCコード プロジェクト

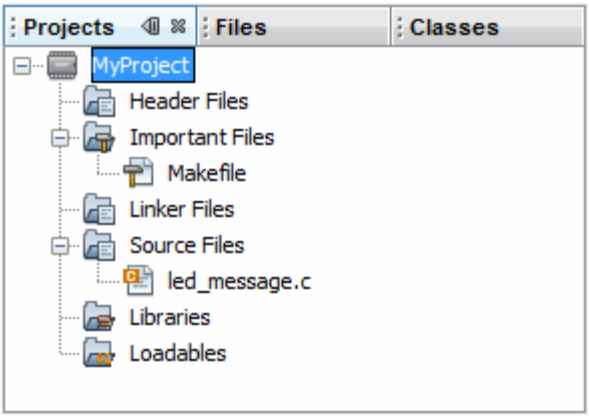


表8-1 [Projects]ウィンドウのフォルダ定義

仮想フォルダ	保存するファイル
Header Files	ヘッダファイル(.h、.inc)
Important Files	その他の重要ファイル(makefile等)や データシートPDF等の文書ファイルをここに含める事ができます。
Linker Files	リンカファイル(.ld、.gld、.lkr)
Source Files	ソースファイル(.c、.cpp、.asm)
Libraries	ライブラリ ファイル(.a、.lib)
Loadables	プリコンパイル済みオブジェクト ファイル(.o)

NetBeansヘルプトピックも参照してください。[Help] > [Help Contents]を選択し、[Search]タブに「Projects Window C or C++」と入力します。リストで最初に表示される[Projects]ウィンドウを見つけます。

8.2 [Files]ウィンドウの表示

[Files]ウィンドウには、[Projects]ウィンドウに表示されないファイルとフォルダを含め、フォルダベースでプロジェクトの内容を全て表示できます。

図8-2 [Files]ウィンドウ – シンプルなCコード プロジェクト

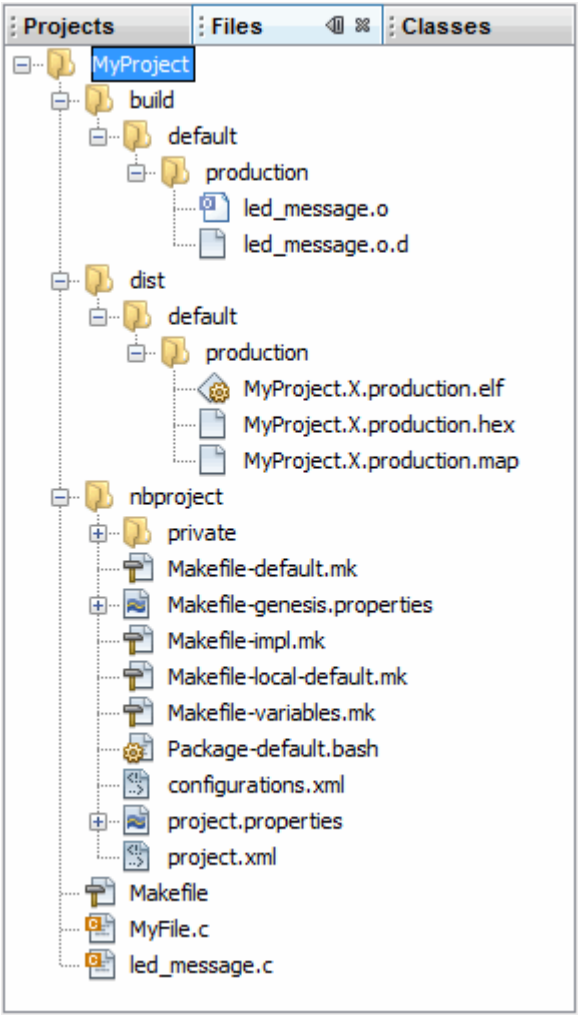


表8-2 [Files]ウィンドウのフォルダ定義

フォルダ	概要
MyProject	MakefileファイルとCコードまたはアセンブリのアプリケーション ファイルを保存するプロジェクト フォルダです。Makefileは、プロジェクトのマスタmakefileです。このファイルはプロジェクト作成時に生成され、それ以降は一切変更されません(再生成されません)。GNUのmakeに詳しければ、このファイルを編集してもかまいません。しかし、MPLAB X IDEでは、Makefile自体を編集する代わりに、プリビルド/makeステップとポストビルド/makeステップを追加する方法を使えます。
build(1)	中間生成ファイルを保存するフォルダです。プロジェクト コンフィグレーション、使い方、保存場所に応じたサブフォルダにファイルを保存しています。 このフォルダは以下のファイルを保存しています。 • 実行ファイル(.o) • 依存関係ファイル(.o.d) • HI-TECH®の中間生成ファイル(.p1)

(続き)

フォルダ	概要
dist(1)	出力ファイルを保存するフォルダです。プロジェクト コンフィグレーション、使い方、保存場所に応じたサブフォルダにファイルを保存しています。 このフォルダは以下のファイルを保存しています。 • 実行ファイル(.hex) • ELFまたはCOFオブジェクト ファイル(.elf、.cof) • ライブラリ ファイル(.a、.lib)
nbproject	makefileとメタデータ ファイルを保存するフォルダです。以下のファイルを保存しています。 • プライベート フォルダは、プロジェクトのユーザおよびコンピュータ固有の設定を保存しています。 • プロジェクトのmakefile • コンフィグレーション別のmakefile • project.xml: IDEが生成したメタデータ ファイル • configurations.xml: メタデータ ファイル
default, MyConfig(2)	プロジェクト コンフィグレーションを保存するフォルダです。ユーザがコンフィグレーションを一切作成しなかった場合、全てのコードは「default」フォルダに保存されます。
production, debug(2)	量産バージョンとデバッグ バージョンのフォルダです。
_ext(2)	外部プロジェクトのファイルを保存するフォルダです。プロジェクト フォルダの外部にあるファイルを参照する場合、そのファイルがこのフォルダに表示されます。

1. これらのフォルダをソース管理に含める必要はありません。これらのフォルダはビルドによって作成されます。5.23「リビジョン管理システムによるソースコードの管理」も参照してください。
2. これらの項目は上図には示していません。

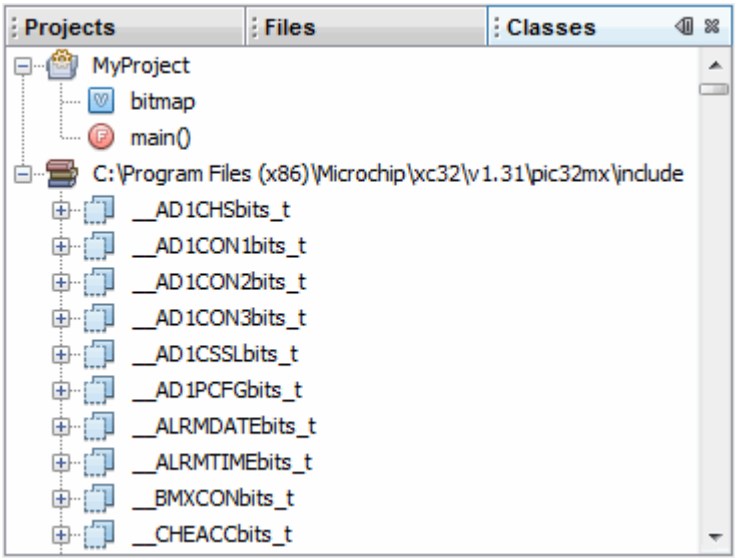
NetBeansヘルプトピックも参照してください。[Help] > [Help Contents]を選択し、[Search]タブに「Files Window C and C++」と入力します。リストで最初に表示される[Files]ウィンドウを見つけます。

8.3 [Classes]ウィンドウの表示

CまたはC++コードの場合、[Classes]ウィンドウ([Window] > [Classes])で全てのクラス(関数)と各クラスのメンバおよびフィールドを確認できます。

NetBeansヘルプトピックも参照してください。[Help] > [Help Contents]を選択し、[Search]タブに「Using the Classes Window」と入力します。

図8-3　[Classes]ウィンドウ – シンプルなCコード プロジェクト



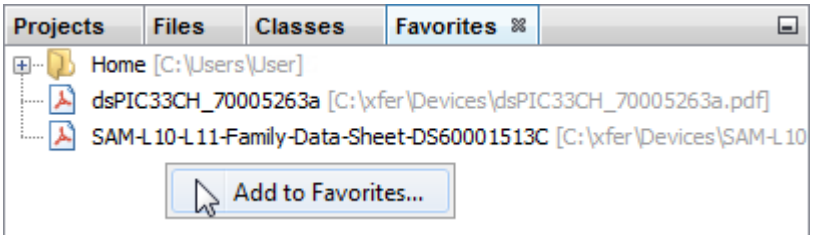
8.4　[Favorites]ウィンドウの表示

[Favorites]ウィンドウ([Window] > [Favorites])を使うと、プロジェクトの内外にかかわらず、コンピュータまたはネットワーク上の任意のファイルまたはフォルダにアクセスできます。

[Favorites]ウィンドウを初めて開いた時には、コンピュータのホームフォルダのみが含まれています。

- ファイルまたはフォルダを追加するには、[Favorites]ウィンドウ内で右クリックして[Add to Favorites]を選択します。
- ファイルを追加するには、[Projects]ウィンドウ内でファイル名を右クリックして[Tools] > [Add to Favorites]を選択します。

図8-4　[Favorites]ウィンドウの表示例



NetBeansヘルプトピックも参照してください。[Help] > [Help Contents]を選択し、[Search]タブに「Favorites Window」と入力します。

8.5　パス/ファイル/フォルダ名の制約

パス/ファイル/フォルダ名には以下のASCII文字のみ使えます。

- A～Z
- a～z
- 0～9
- アンダースコア: _
- ダッシュ: -
- ピリオド: .

Windows

Windowsの最大パス長は260文字です。Microsoft社による説明は以下のURLをご覧ください。

https://msdn.microsoft.com/en-us/library/windows/desktop/aa365247(v=vs.85).aspx

Windowsは、ユーザが長過ぎるパスを作成する事を妨げようとしますが、特定のアプリケーション、カット&ペースト等で、260文字を超えるパスを指定する事は可能です。しかし、プロジェクトのソースファイルが長過ぎるパスを使っている場合、プロジェクトは正常にビルドされません。

また、Windowsはコマンドを8191文字に制限しています。詳細は以下を参照してください。

セクション「Command line too long (Windows)」

クロス プラットフォーム オペレーティング システム

MPLAB X IDEを複数のプラットフォーム(Windows、Mac、Linux)で使う場合、以下の点に注意します。

- 相対パスにはフォワード スラッシュ「/」を使う必要があります。バックスラッシュ「\」は、Windowsプラットフォーム以外では使えません。例:
  `#include headers/myheader.h.`
- Linuxは大文字と小文字を区別します。従って`genericTypeDefs.h`と`GenericTypeDefs.h`は区別されます。

8.6 パス/ファイル/フォルダ プロジェクトに関する推奨事項

プロジェクトに既存のファイルまたはフォルダを追加する場合、8.5「パス/ファイル/フォルダ名の制約」に記載した制約に注意してください。特にパスの長さには注意してください。Windowsには既知の制限(パス260文字、コマンド8191文字)がありますが、パスが非常に長いと他のシステムで問題が発生する可能性があります。ビルドに失敗する場合、パスを短くするか、ファイルまたはフォルダをプロジェクト フォルダに移動してみます。

移植性を最大限に考慮したプロジェクトにするには、ファイルまたはフォルダのインポート時に相対パスを使います。

関連リンク

4.9.3「既存ファイルのプロジェクトへの追加」
6.11「MPLAB X IDEプロジェクトのパッケージ化」
8.5「パス/ファイル/フォルダ名の制約」

8.7 ユーザ コンフィグレーション データの表示

MPLAB X IDEのユーザフォルダ(userdir)は、ユーザ コンフィグレーション データ(例: ウィンドウ構成、エディタ設定、メニューとツールバーのカスタマイズ、各種モジュールの設定(既知のコンパイラー覧等))を保存しています。

プラグインをインストールする場合([Tools] > [Plugins])、プラグイン情報(コード)は、MPLAB Xインストール フォルダではなく、ユーザフォルダに保存されます。この保存場所は、[Tools] > [Plugins] > [Settings] > [Advanced]の下で[Plugin Install Location]ドロップダウン メニューから[Force install into shared directories]を選択して変更できます。

ユーザフォルダの保存場所を調べるには、[Help] > [About]を選択し**Userdir**の次のパスを見つけます。この情報はMPLAB X IDEが作成および管理します。

8.8 MPLAB IDE v8プロジェクトのインポート – 相対パス

MPLAB IDE v8プロジェクトは以下のパスに保存されています。

```
C:\MyProjects\mplab8project
```

MPLAB IDE v8プロジェクトをMPLAB X IDEにインポートした場合、以下のファイルが作成されます。

```
C:\MyProjects\mplab8project\mplab8project.X
```

既定値では、MPLAB X IDEにインポートしたプロジェクトは、両方のプロジェクトの保守性を維持するために、MPLAB IDE v8プロジェクトの下に置かれます。これに対し、MPLAB X IDEのプロジェクト フォルダはどこに置いてもかまいません。また、プロジェクト名はmplab8project.Xに設定されますが、変更してもかまいません。MPLAB X IDEのプロジェクト名は.Xで終わるという規則がありますが、あくまで便宜的なものです。

新規プロジェクトは、ソースファイルをプロジェクト フォルダにコピーしません。その代わりに、v8フォルダ内のファイル保存場所を参照します。独立したMPLAB X IDEプロジェクトを作成するには、新規プロジェクトを作成して、MPLAB IDE v8のソースファイルを、そのプロジェクトにコピーする必要があります。

詳細は5.3「MPLAB IDE v8プロジェクトのインポート」を参照してください。

8.9 プロジェクトの移動、コピー、リネーム

プロジェクトを作成した後で、変更する必要がある事に気付く場合があります。プロジェクトを右クリックすると開くコンテクスト メニューを使うと、MPLAB X IDE内でプロジェクトを移動、コピー、リネームできます。プロジェクトを削除する事もできます。詳細は12.15「[Projects]ウィンドウ」 - [Project]メニューを参照してください。

外部ツールを使う事もできます。プロジェクト ファイルの構造は、MPLAB X IDEを使わなくても問題ないようになっています。

8.10 プロジェクトの削除

MPLAB X IDEでプロジェクトを削除する方法

1. [Projects]ウィンドウでプロジェクト名を右クリックして[Delete]を選択します。
2. [Delete Project]ダイアログでは以下を削除できます。
    2.1. コンピュータに保存されているプロジェクト ファイル - 一部のソースファイルは削除されません。
    2.2. プロジェクト ファイルと全ソースファイル (チェックボックス経由)プロジェクトは削除され、

MPLAB X IDEからは見えなくなります。

Note: MPLAB X IDEは終了時に、プロジェクトに割り当てられた全てのリソースを解放する訳ではありません。MPLAB X IDEはJava上で動作しているため、リソースの解放は即座には行われません。Javaでは、ガーベジ コレクション(GC)を呼び出すタイミングをJREに決定させるのが最善の方法です。しかし、メモリツールバーを有効にしてクリックする事で、GCを強制的に実行する事もできます。

9. トラブルシュート

本章には、MPLAB X IDEの使用中に問題やエラーが発生した場合の対処に役立つ情報を記載しています。ここに記載した方法で対処できなかった場合はMicrochip社までお問い合わせください。連絡先は本書の「サポート」に記載しています。

9.1 USBドライバのインストールに関する問題

正しいUSBドライバのインストール方法は2.3「USBデバイスドライバ (ハードウェア ツール用)のインストール」を参照してください。

問題が生じた場合の対処方法は2.3.3.5「ツールの通信の問題」を参照してください。

9.2 オペレーティング システムに関する問題

オペレーティング システムが異なると、パス、ファイル、フォルダの命名に関する問題も異なります。詳細は以下を参照してください。

8.5「パス/ファイル/フォルダ名の制約」

9.3 NetBeansプラットフォームに関する問題

NetBeansプラットフォームのリリース バージョンによっては、MPLAB X IDEの使用に際して問題が生じる可能性があります。詳細は以下のNetBeansウェブサイトをご覧ください。

http://www.netbeans.org
http://netbeans.org/community/releases/index.html

以下はMPLAB X IDEで既知の問題です。

- 前景色として[Inherited]を選択すると、カテゴリのテキストがエディタ内で見えなくなる
- デバッグ用のインジケータが表示されない
- 識別子を解決できない
- コードを素早くステップ実行すると、[Watches]ウィンドウでSFRが選択解除される
- 9.3.5「サブメニューに移動するとメニューが消える事がある」

9.3.1 前景色として[Inherited]を選択すると、カテゴリのテキストがエディタで見えなくなる

フィールドの前景色([Foreground])が[Inherited]の場合、ポップアップ完了ウィンドウ (ヒント)内のテキストが見えなくなります。ポップアップ完了ウィンドウには、未選択のエントリが白の背景に白のテキストとして表示されます。一覧を下にスクロールすると、エントリごとに青の背景に白のテキストで表示されます。それ以外の機能は正常に働いているように見えます。

この問題はNetbeans Bugzillaの[Fonts & Colors]の問題として登録されています。

https://netbeans.org/bugzilla/show_bug.cgi?id=249766

[Options] > [Fonts & Colors]。[Syntax]タブの任意のカテゴリで前景色として[Inherited]を選択すると、エディタ内でテキストが見えなくなります。プリファレンスxmlファイル「org-netbeans-modules-editor-settings-CustomFontsColors-tokenColorings.xml」を見ると、問題のあるカテゴリ(例: 「Field」、「Identifier」)の前景の属性に有効な色へのリンクが存在しない事が分かります。

回避策:

1. [Tools] > [Options]、**[Fonts & Colors]**ボタン、**[Syntax]**タブを選択します。
2. [Language]に[C]を選択します。
3. [Category]で[Field]をクリックします。
4. [Foreground]に[Inherited]が選択されている事を確認します。リストから任意の色を選択します。
5. **[OK]**をクリックします。

9.3.2 デバッグ用のインジケータが表示されない

デバッグ実行中のようですが、一時停止矢印または線の強調表示がなく、ブレークポイントが破損して表示されます(ガーターのアイコンが壊れている)。これはファイルパスが正しく設定されていない時に発生します。8.5「パス/ファイル/フォルダ名の制約」を参照してください。

9.3.3 識別子を解決できない

ライブパーサが名前のない構造にネストされた構造の未解決識別子を表示します。これは、NetBeansネイティブCNDが匿名フィールド宣言をサポートしていないためです。これはフィールドの識別子を指定する事で回避できます。ただし、コードでは使用するたびにそのフィールド名を指定する必要があります。

例:

```
struct dados {
    unsigned char signature1;
    unsigned char signature2;
};
typedef union {
    unsigned char data [2];
    struct dados;          // unresolved ident.
} EEPROM DATA;

void main(void)
    { EEPROM_DATA
    edata;
    edata.data [0] = 0xAA;    // this is ok
    edata.signature2 = 0xDD; // unresolved ident.
```

以下も参照してください。

https://netbeans.org/bugzilla/show_bug.cgi?id=256329

9.3.4 コードを素早くステップ実行すると、[Watches]ウィンドウでSFRが選択解除される

ステップ実行すると、カーソルがソースで更新され、[Watches]ウィンドウからフォーカスが削除されます。その後、Netbeansは[Watches]ウィンドウ全体を再描画し、ハイライトを削除します。

9.3.5 サブメニューに移動するとメニューが消える

WindowsでMPLAB X IDEメニューを選択してサブメニューを選択すると、メニューが消える事があります。この問題を解決するには、MPLAB X IDEを再起動します。

9.4 MPLAB X IDEに関する問題

下表にMPLAB X IDEの問題を示します。この表で問題の解決策が見つからない場合、プロジェクトのデバッグおよびコンパイルツール関連文書も参照してください。

- MPLAB IDE v8プロジェクトのインポート – 設定
- MPLAB IDE v8プロジェクトのインポート – 変更されたリンカスクリプト
- MPLAB C18 Cコンパイラと構造
- プロジェクトがエンプティである/プロジェクト ファイルが灰色表示されている
- エディタ内のコードが#error文の後に灰色表示されている
- インターネット接続が安定していないため、最新のツールチェーンをダウンロードできない

9.4.1 MPLAB IDE v8プロジェクトのインポート – 設定

MPLAB IDE v8のワークスペースに保存されている設定(ツール設定等)は、新しいMPLAB X IDEプロジェクトに転送されません。ワークスペースに保存されている情報の内容については、MPLAB IDE v8ヘルプ([MPLAB IDE Reference] > [Operational Reference] > [Saved Information])を参照してください。

9.4.2 MPLAB IDE v8プロジェクトのインポート – 変更されたリンカスクリプト

MPLAB IDE v8プロジェクトのリンカスクリプトを変更してオブジェクト ファイルをインクルードするようにしていた場合、MPLAB X IDEにインポートすると、リンカはそのファイルを見つける事ができません。MPLAB IDE v8とMPLAB X IDEではビルドパスが異なるからです。

従って、以下のようなエラーが表示されます。

```
<install path>ld.exe: cannot find file.o
```

MPLAB IDE v8では、ビルド関連のファイルが全て1つのフォルダに保存されるのに対し、MPLAB X IDEではビルドファイルが別々のサブフォルダに保存されるためです。

既存のリンカスクリプトにワイルドカードを使えば、MPLAB X IDEでも動作させる事ができます。例えば、

```
/* Global-namespace object initialization - MPLAB v8*/
.init   :
{
KEEP (crti.o(.init))
:
} >kseg0_program_mem
```

から以下に変更します。

```
/* Global-namespace object initialization - MPLAB X*/
.init   :
{
KEEP (*crti.o(.init))
:
} >kseg0_program_mem
```

これとは別に、Cコードに関数を配置できるアドレス属性を使う方法もあります。

```
int_attribute_((address(0x9D001000))) myfunction (void) {}
```

これによって、既定値のリンカスクリプトを変更する事なく、アドレスに関数を配置できます。

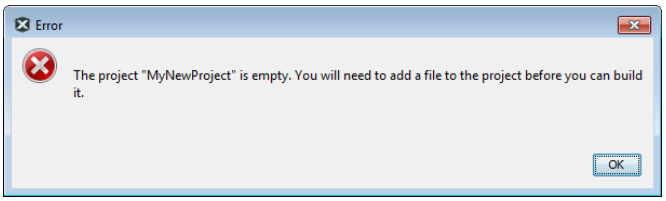
9.4.3 MPLAB C18 Cコンパイラと構造

RAMメモリ内の構造のポインタ値は、[Watches]ウィンドウ内では正しく表示されません。これは、コンパイラが完全なデバッグ情報を保存しないためです。この問題の回避策は、MPLAB XC8 Cコンパイラを代わりに使う事です。

9.4.4 プロジェクトがエンプティである/プロジェクト ファイルが灰色表示されている

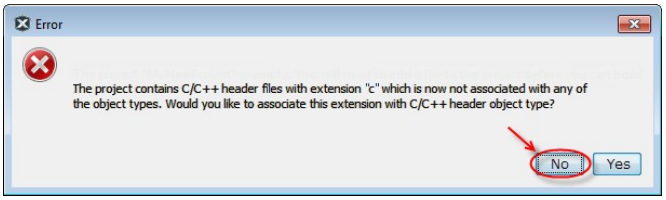
破損したプロジェクトをロード、かつダイアログの質問の答えを間違えた場合、[Projects]ウィンドウが灰色表示され、「ソースファイルがないためプロジェクトをビルドできない」というメッセージが表示されます。

図9-1　プロジェクト エンプティエラー



この問題は、破損したプロジェクトをMPLAB X IDEにロードした場合に発生します。Cソースファイルをヘッダファイルに関連付けられるかどうか尋ねるダイアログが表示されます。この時点で[No](既定値)をクリックする事で、この問題を回避できます。

図9-2　新ヘッダ拡張子エラー



しかし、[Yes]をクリックするとソースファイルがヘッダファイルとして認識されてしまい、プロジェクトは空のように見えます。プロジェクトを修復するため、以下のどちらかの方法が使えます。

- MPLAB X IDEを終了します。

- [Help] > [About]ダイアログで指定されているMPLAB X IDEのユーザフォルダに移動し、MPLAB X IDEの使用中のバージョンのフォルダを削除します。例えば、フォルダ`v3.05`を`C:\Users\UserName\AppData\Roaming\mplab_ide\dev\v3.05`から削除します。
- MPLAB X IDEを再起動します。

または

- [Tools] > [Options]を選択し、[Embedded]ボタンをクリックし、**[Other]**タブを開きます。
- ヘッダファイルの関連付けから「c」を削除します。
- ソースファイルの関連付けに「c」を追加します。
- **[OK]**をクリックします。

9.4.5 エディタ内のコードが#error文の後に灰色表示されている

7.7「C/C++コードのエラー ディレクティブ」を参照してください。

9.4.6 インターネット接続が安定していないため、最新のツールチェーンをダウンロードできない

インターネット接続が不安定であると判断されたため、以下のリンクで説明している機能を利用できません。

4.1.8.2「最新MPLAB XCコンパイラのダウンロード」

9.5 エラー

IDEは様々な形でエラーを示しますが、多くの場合ウィンドウ内のアイコンまたは[Output]ウィンドウ内のメッセージとしてエラーを表示します。アイコンにカーソルを合わせると、問題の内容を示すメッセージが表示されます。テキスト メッセージについては、オンラインヘルプで該当するエラーを検索してください。

IDEとエディタの一般的なエラーと解決方法を以下のセクションに示します。

- Command line too long (Windows)
- Couldn’t reserve space for Cygwin™ heap (Windows 7または8)
- Destination Path Too Long/The filename or extension is too long (Windows)
- The project ProjectName is empty
- For 8-bit code, #asm and #endasm cause red bangs in the Editor window
- For 16-bit code, MPLAB XC16 packed attribute statement causes red bangs in the Editor window
- Hexmate Conflict Report Address Error
- Programming Errors
- Programmer to Go Not Supported
- Mouseover shows “Not Recognized” in Editor
- Error Code in -10000 Range

9.5.1 Command line too long (Windows)

プロジェクトのmakefile実行時、各コマンドはローカル ネイティブシェルに渡されます。Windowsでは8191文字という制限があります。従って、数百Cファイルをリンクするプロジェクトでは、この制限を超えて上のエラーが表示される事があります。

LinuxとmacOSは、通常非常に長いコマンドラインを扱う事ができます。使用中のシステムでその長さを調べるには、コマンドラインでgetconf ARG_MAXと入力します。

応答ファイルを使った回避策

MPLAB XC8/MPLAB XC32 Cコンパイラのv1.01以降、MPLAB XC16 Cコンパイラのv1.22以降の場合、リンカを呼び出す際に応答ファイルを使えば、この問題を回避できる場合があります。

- MPLAB XC8の場合、[Project Properties]*ウィンドウで[Categories]の[XC8 linker]をクリックします。[Option categories]で[Additional Options]を選択します。[Use response file to link]横のチェックボックスを選択します。
- MPLAB XC16の場合、[Project Properties]*ウィンドウで[Categories]の[xc16-ld]をクリックする。[Option categories]で[General]を選択します。[Use response file to link]横のチェックボックスを選択します。
- MPLAB XC32の場合、[Project Properties]*ウィンドウで[Categories]の[xc32-ld]をクリックする。[Option categories]で[General]を選択します。[Use response file to link]横のチェックボックスを選択します。

* [Project Properties]ウィンドウを開くには、4.3「[Project Properties]ウィンドウを開く」を参照してください。

ライブラリによる回避策

その他全てのコンパイラの場合、MPLAB X IDEライブラリ プロジェクトを作成し、メイン プロジェクトからライブラリ プロジェクトにいくつかのソースファイルを移動します。その後、ライブラリ プロジェクトをメイン プロジェクトに追加する事で、この問題を回避できます。

ライブラリ プロジェクトを作成するには、[File] > [New Project]を選択し、[Microchip Embedded]カテゴリをクリックし、プロジェクトとして[Library Project]を選択します。

ライブラリをメイン プロジェクトに追加するには、[Project Properties]ウィンドウを開き([File] > [Project Properties])、[Categories]の[Libraries]を選択し、**[Add Library Project]**ボタンをクリックします。

アーカイブ チェックボックスによる回避策

現在ではMPLAB XC16とMPLAB XC32は、長いファイルセットをライブラリにアーカイブする際にリンク行が文字数の制限を超える可能性がある場合に使えるチェックボックスを備えています。「Break into multiple lines」を使ってアーカイブ行を短い行に分割し、この制限を回避するようにコンパイラを設定できます。

このオプションは、[Project Properties]ウィンドウで[xc16-ar]または[xc32-ar]カテゴリを選択した後、[General]オプション カテゴリの[Break into multiple lines]チェックボックスで設定します。

9.5.2 Couldn't reserve space for MinGW heap

MPLAB X IDEは、make処理にMinGWを使います。Windowsでは、仮想メモリの割り当てエラーが発生する可能性があります。

Windows 7または8

仮想メモリの設定を変更するには、[スタート] > [コントロール パネル]をクリックしてコントロール パネルを開きます。[システム]をクリックし、[システムとセキュリティ] > [システム] > [システムの詳細設定]または[システムの保護]を選択します。**[詳細設定]**タブで、[パフォーマンス]の**[設定]**ボタンをクリックします。このダイアログで、**[詳細設定]**タブをクリックし、**[変更]**ボタンをクリックします。[カスタムサイズ]に以下の値を入力します。

- 初期サイズ(MB) = 「現在の割り当て」の値(下段に表示されている値)
- 最大サイズ(MB) = 「推奨」の値(下段に表示されている値)

[設定]をクリックし、**[OK]**をクリックしてからPCを再起動します。**Windows 10**

以下を参照してください。

https://www.geeksinphoenix.com/blog/post/2016/05/10/how-to-manage-windows-10-virtual-memory.aspx

9.5.3 Destination Path Too Long/The filename or extension is too long (Windows)

Windowsの最大パス長は260文字です。詳細は8.5「パス/ファイル/フォルダ名 の制約」を参照してください。

9.5.4 The project ProjectName is empty

9.4.4「プロジェクトがエンプティである/プロジェクト ファイルが灰色表示されている」を参照してください。

9.5.5 For 8-bit code, #asm and #endasm cause red bangs in the Editor window

`#asm`文と`#endasm`文はMPLAB XC8 Cプログラム内でインライン アセンブリコードを宣言するのに有効な構造です。しかしIDEのエディタでは、Cプリパーサはこれらを有効なディレクティブとは認識せず、これらの文の隣に赤の感嘆符を表示します。しかし、このコードはこのままでも動作します。

エディタ内の赤の感嘆符を除去する場合、`asm()`文を使います。これは、全てのMPLAB XC Cコンパイラでインライン アセンブリを使う際に推奨する方法です。

例:

```
// Inline Code that causes error
    #asm
        movlw 0x20;
        nop;
```

```
        nop;
    #endasm
// Workaround for inline assembly - that will not cause error
// (Multi line asm code)
    asm("\
    movlw
    0x20;\ nop;\
     nop;");
// Workaround for inline assembly - that will not cause Error

// (Single line asm code)
    asm("movlw 0x20; nop; nop");
```

9.5.6 For 16-bit code, MPLAB XC16 packed attribute statement causes red bangs in the Editor window

コンパイラのパーサは以下の文をエラーとしてマーク表示します。しかし、これらの文はビルドされます。

```
unsigned long long Bits8 : 8_attribute ((packed));
```

修飾子に対するパーサのルールでは順序は決まっていません。通常は修飾子は全て識別子の前に指定します。従って、以下のどちらかの文を使えばエラーは発生しません。

```
unsigned long long attribute ((packed)) Bits8 : 8;
```

または

```
attribute ((packed)) unsigned long long Bits8 : 8;
```

9.5.7 Hexmate Conflict Report Address Error

Hexmateが競合レポートを生成した場合、HEXファイル内の競合アドレスとデータが格納されているメモリの[Memory]ウィンドウに表示されるアドレスが異なる場合があります。以下に例を示します。

表9-1　HEXアドレスとメモリアドレスの関係

デバイスファミリ	HEXファイル内のアドレス	メモリ内のアドレス
ベースライン	0x100	0x80*
ミッドレンジ	0x100	0x80*
PIC18 MCU	0x100	0x100
16ビットデバイス	0x100	0x80*
32ビットデバイス	0x100	0x100
* デバイスによっては、競合レポートに表示されたアドレスはメモリ内のデータアドレスの2倍の値となります。		

9.5.8 プログラムエラー

デバイスをコード保護した後にプログラミングしようとすると、アドレス領域に対してプログラミング エラーが出力されます。

[Erase Device Memory Main Project]  機能は、これがプログラミングの問題かどうかを判断するのに役立ちます。この機能は、ツールバーにアイコンとして追加できます([View] > [Toolbars] > [Customize])。またはMPLAB IPEを使います。

1. コード内でコード保護が無効になっている事を確認します。
2. デバイスメモリを消去します。これによりコンフィグレーション ビットの設定が消去されます。
3. デバイスを再プログラミングします。

プログラミングが正常終了した場合、コード保護が原因であった事が分かります。

9.5.9 Programmer to Goがサポートされていない

現在、コンフィグレーション(config)ワードが多過ぎるデバイスはProgrammer to Go (PTG)をサポートできません。PTGはconfigワードが一度に送信される事を期待していますが、MPLAB X IDEはconfigワードが大量にあるデバイスに対して、各configワードを一度に1つずつ送信します。

9.5.10 エディタ内のマウスオーバーで[Not Recognized]と表示される

エディタ内で値を表示するためにマウスオーバーを使う場合、以下を確認します。

1. デバッグモード中([Debug] > [Debug Project])である(デバッグ実行中でないとシンボル値は表示されません。)
2. マクロにマウスを合わせていない(この方法でマクロの値を見る事はできません。マクロを右クリックして[Navigate] > [View Macro Expansion]を選択し、[Macro Expansion]ウィンドウを開きます。)

9.5.11 Error Codes in -10000 Range

「Error codes in the -10000 range」はMicrosoftエラーコードを表します。Microsoftエラーコードを確認するには、数値を正(-10121は10121)にし、10000を減算します(10121-10000 = 121)。Microsoftエラーコードは以下で確認できます。

https://msdn.microsoft.com/en-us/library/windows/desktop/ms681382(v=vs.85).aspx

9.6 フォーラム

ここまでの説明で問題が解決しない場合、Microchip社のフォーラムを参照してください。

http://www.microchip.com/forums/f238.aspx

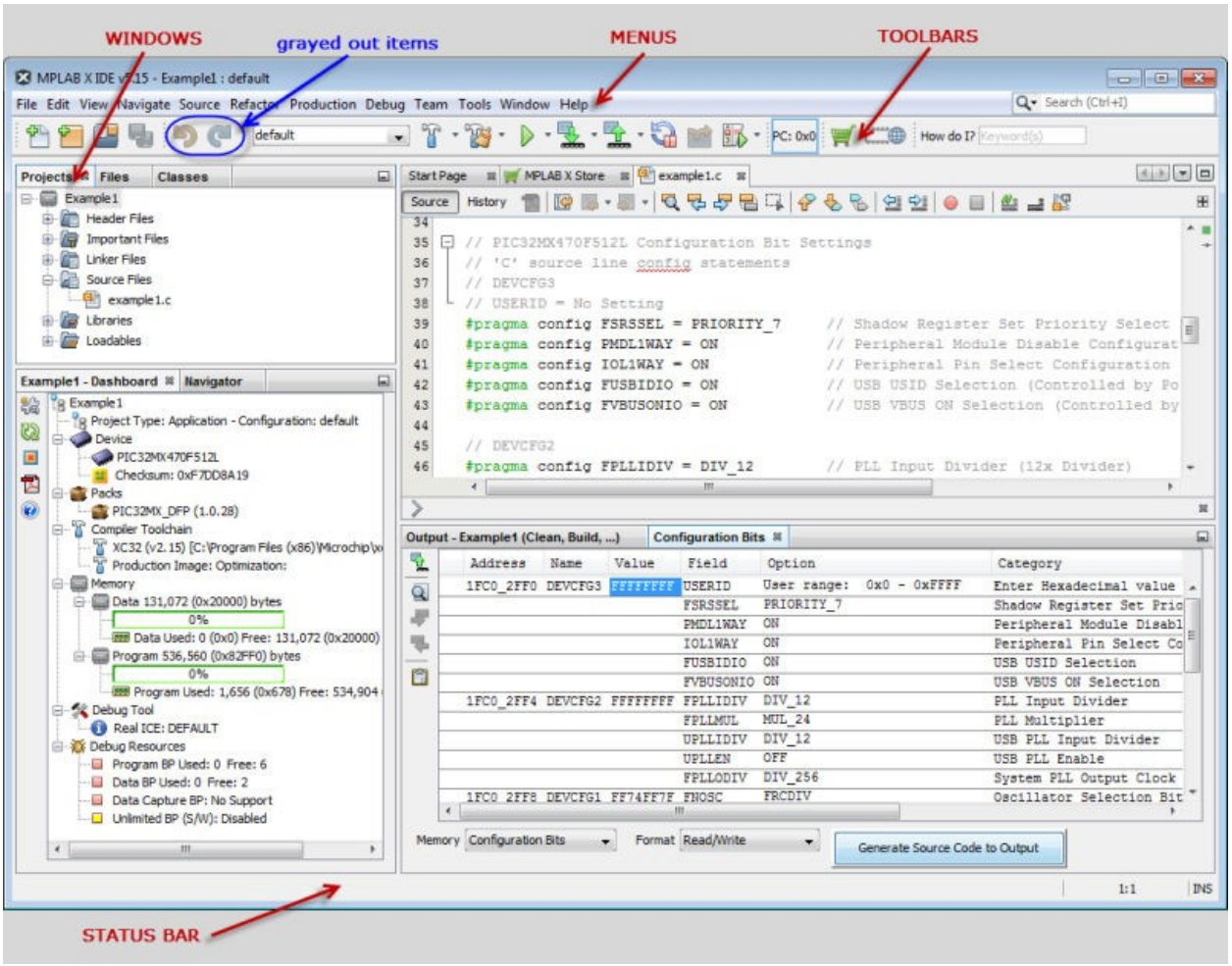
このフォーラムは、問題に関するディスカッションや投稿された解決策を掲載しています。

10. デスクトップの詳細

MPLAB® X IDEデスクトップは、他のデスクトップ アイテムとは独立して動作するサイズ変更可能なウィンドウです。デスクトップはメニュー、ツールバー、ステータスバー、タブ形式のウィンドウで構成されています。本章ではメニュー、ツールバー、ステータスバーについて説明します。ウィンドウとダイアログについては次章で説明します。

Note: NetBeansヘルプトピックで「workspace」と書かれている場合、それは「デスクトップ」の事を意味します。この「workspace」は、MPLAB IDE v8以前のバージョンの「ワークスペース」の事ではありません。

図10-1 MPLAB® X IDEのデスクトップ



10.1 メニュー

MPLAB® X IDEの多くの機能は、デスクトップ最上段にあるメニューバーからメニュー項目として選択できます。メニュー項目の右側に「(...)」が付く場合、その項目を選択するとダイアログが開きます。ダイアログの詳細は12.「MPLAB X IDEのウィンドウとダイアログ」を参照してください。

メニュー項目の横にはショートカット キーが表示されます

例: [New File]のショートカット キーは<Ctrl>+<N>です。ショートカット キーの詳細は[Help] > [Keyboard Shortcuts Card]を参照してください。

メニュー項目は状況に応じて無効化(灰色表示)される事があります。10.4「メニュー項目とボタンの灰色表示または非表示」を参照してください。

ウィンドウ内で右クリックすると、コンテクスト メニューが表示されます。これらのメニューの詳細は12.15「[Projects]ウィンドウ」を参照してください。

メニュー一覧を以下に記載します。

Note: [Start Page]で[My MPLAB IDE] > [Extend MPLAB] > [Selecting Simple or Full-Featured Menus]を選択すると、全てのメニューとメニュー項目を表示できます。それらのメニュー項目には、組み込みシステム開発では使わない項目も含まれています。

10.1.1 [File]メニュー

本セクションでは[File]メニューの項目を示します。

これらのメニュー項目の一部に割り当てられているキーボード ショートカットについては[Help] > [Keyboard Shortcuts Card]を参照してください。

表10-1 [File]メニューの項目

コマンド	動作
New Project	[New Project]ウィザードを使ってプロジェクトを新規作成します。
New File	[New File]ウィザードを使ってファイルを新規作成します。
Open Project	既存のプロジェクトを開きます。
Open Recent Project	最近開いた全てのプロジェクトの一覧を表示します。
Import	以下のいずれかをインポートします。 • Hex/ELF...(Prebuilt) File – 別のツールを使ってビルドしたファイル • MPLAB IDE v8 Project – [Import Legacy Project]ウィザードを起動 • その他の組み込み – その他の組み込みプラットフォームからのインポート • ZIPから – アーカイブしたプロジェクトからのインポート (ファイルを解凍してインポート)
Close Project	現在のプロジェクトを閉じます。
Close Other Projects	メイン プロジェクト以外の全プロジェクトを閉じます。
Close All Projects	開いている全てのプロジェクトを閉じます。
Open File	既存のファイルを開きます。
Open Recent File	ファイルを選択するために、最近開いた全てのファイルの一覧を表示します。
Project Group	現在のプロジェクトをグループに割り当てます。
Project Properties	[Project Properties]ウィンドウを開きます。
Save	現在のファイルを保存します。
Save As	現在のファイルを、パス/名前を変更して保存します。
Save All	開いている全てのファイルを保存します。 [Compile on Save]を設定している場合、保存時にプロジェクト ファイルをコンパイル/ビルドします。
Page Setup	印刷用にページを設定します。
Print	現在のファイルを印刷します(プレビュー付き)。
Print to HTML	現在のファイルをHTML形式で書き出します。
Exit	MPLAB® X IDEを終了します。

10.1.2 [Edit]メニュー

本セクションでは[Edit]メニューの項目を示します。

これらのメニュー項目の一部に割り当てられているキーボード ショートカットについては[Help] > [Keyboard Shortcuts Card]を参照してください。

表10-2　[Edit]メニューの項目

コマンド	動作
Undo	以前のエディタ操作を1つずつ遡って取り消します(保存操作は取り消せません)。
Redo	以前の[Undo]操作を1つずつ元に戻します。
Cut	現在選択している内容を削除し、クリップボードに移動します。
Copy	現在選択している内容をクリップボードにコピーします。
Paste	クリップボードの内容を現在の選択位置に挿入します。
Paste Formatted	クリップボードの内容を、書式設定付きで現在の選択位置に挿入します。
Paste from History	コピー履歴の内容を現在の選択位置に挿入します。
Delete	現在選択している内容を削除します。
Select All	現在の文書またはウィンドウ内の全ての内容を選択します。
Select Identifier	カーソルに最も近い単語を選択します。
Find Selection	現在選択している内容に一致するインスタンスを検索します。
Find Next	一致するテキストのインスタンスを順方向に検索します。
Find Previous	一致するテキストのインスタンスを逆方向に検索します。
Find	文字列を検索します。
Replace	文字列を検索し、一致した文字列を指定した別の文字列に置換します。
Find Usages	選択したコードが使われている箇所とサブタイプを検索します。
Find in Projects	指定したテキスト、オブジェクト名、オブジェクト タイプをプロジェクト内で検索します。
Replace in Projects	テキスト、オブジェクト名、オブジェクト タイプをプロジェクト内で置換します。
Start Macro Recording	マクロ用のキー操作の記録を始めます。
Stop Macro Recording	キー操作の記録を終了します。 マクロを管理するには、[Tools] > [Options]を選択し、**[Editor]**ボタンをクリックし、**[Macros]**タブを開きます。

10.1.3　[View]メニュー

本セクションでは[View]メニューの項目を示します。

これらのメニュー項目の一部に割り当てられているキーボード ショートカットについては[Help] > [Keyboard Shortcuts Card]を参照してください。

表10-3　[View]メニューの項目

コマンド	動作
Editors	ファイルを表示するエディタを選択します。 • History – このファイルの編集履歴を表示します。 • Source – このファイルのソースコードを表示します。
Split	現在のエディタ ウィンドウを、垂直または水平方向に2分割します。
Code Folds>Collapse Fold	カーソル位置が折り畳み可能なテキスト内である場合、そのセクションを折り畳みます。
Code Folds>Expand Fold	カーソル位置が折り畳まれたセクションである場合、そのセクションを展開して表示します。

(続き)

コマンド	動作
Code Folds>Collapse All	折り畳み可能なテキスト セクションを全て折り畳みます。
Code Folds>Expand All	折り畳まれているテキスト セクションを全て展開します。
Code Folds>Collapse Fold Tree	ネスティングされた複数の折り畳みを全て折り畳みます。
Code Folds>Expand Fold Tree	ネスティングされた複数の折り畳みを全て展開します。
Web Browser	既定値のウェブブラウザを開いて、NetBeansホームページを表示します。
IDE Log	[Output]ウィンドウのタブにログファイルを開きます。
Toolbars>File, etc.	関連ツールバーを表示します。
Toolbars>Small Toolbar Icons	Uses small icons on the toolbars.
Toolbars>Reset Toolbars	ツールバーを既定値の状態に戻します。
Toolbars>Customize	既存ツールバーのカスタマイズまたはツールバーを新規作成できます。
Show Editor Toolbar	[File]タブにエディタのツールバーを表示します。
Show Line Numbers	左の余白に行番号を表示します。
Show Non-printable Characters	印刷できない文字(非ASCI)であっても表示します。
Show Breadcrumbs	エディタ ウィンドウ下部にブレッドクラム (プロジェクト ファイルからそのファイルまでのパス)を表示します。
Show Indent Guide Lines	インデントを表す薄い線を表示します。
Show Diff Sidebar	DIFFサイドバーを表示します。
Show Versioning Labels	バージョンラベルを表示します。
Synchronize Editor with Views	開いている画面にエディタを同期させます。
Show Only Editor	IDEウィンドウ全体にエディタ ウィンドウを表示します。
Full Screen	ウィンドウを全画面表示にします。

10.1.4 [Navigate]メニュー

本セクションでは[Navigate]メニューの項目を示します。

これらのメニュー項目の一部に割り当てられているキーボード ショートカットについては[Help] > [Keyboard Shortcuts Card]を参照してください。

表10-4 [Navigate]メニューの項目

コマンド	動作
Go to File	指定したファイルを検索して開きます。
Go to Type	指定したクラスまたはインターフェイスを検索して開きます。
Go to Symbol	指定したシンボル名を検索します。
Go to Previous Window	1つ前に選択したウィンドウにフォーカスします。
Go to Source	選択したクラスの定義を含むソースファイルを表示します。
Go to Declaration	カーソル位置のアイテムの宣言位置へ移動します。
Go to Super Implementation	カーソル位置のアイテムのスーパー 実装位置へ移動します。
Last Edit Location	直前に編集した位置にジャンプします。
Back	逆方向に移動します。

(続き)

コマンド	動作
Forward	順方向に移動します。
Go to Line	指定した行に移動します。
Toggle Bookmark	コード行にブックマークを設定します。
Bookmark History Popup Next	[Bookmark]ウィンドウの次のブックマークに移動します([Window] > [IDE Tools] > [Bookmarks])。
Bookmark History Popup Previous	[Bookmark]ウィンドウの1つ前のブックマークに移動します([Window] > [IDE Tools] > [Bookmarks])。
Next Error	次のビルドエラーを含む行にジャンプします。
Previous Error	1つ前のビルドエラーを含む行にジャンプします。
Select in Projects	[Projects]ウィンドウを開いて、その中で現在の文書を選択します。
Select in Files	[Files]ウィンドウを開いて、その中で現在の文書を選択します。
Select in Classes	[Classes]ウィンドウを開いて、その中で現在の文書を選択します。
Select in Favorites	[Favorites]ウィンドウを開いて、その中で現在の文書を選択します。

10.1.5 [Source]メニュー

本セクションでは[Source]メニューの項目を示します。

これらのメニュー項目の一部に割り当てられているキーボード ショートカットについては[Help] > [Keyboard Shortcuts Card]を参照してください。

表10-5 [Source]メニューの項目

コマンド	動作
Format	選択したコード(何も選択していない場合はファイル全体)をフォーマットします。
Remove Trailing Spaces	行末のスペースを削除します。
Shift Left	選択した行をタブ1つ左へシフトします。
Shift Right	選択した行をタブ1つ右へシフトします。
Move Up	選択した行を1行上へ移動します。
Move Down	選択した行を1行下へ移動します。
Move Code Element Up	選択したコードエレメントを1行上へ移動します。
Move Code Element Down	選択したコードエレメントを1行下へ移動します。
Duplicate Up	選択した行を1行上に複製します。
Duplicate Down	選択した行を1行下に複製します。
Toggle Comment	選択した行をコメント化(または解除)します。
Complete Code	コード補完ボックスを表示します。
Insert Code	コンストラクタ、ゲッター、セッター等、一般的な構造体の生成に使えるコンテクスト メニューを開きます。
Fix Code	エディタが提示するヒントを表示します。 ユーザに提示可能なヒントが存在する場合、IDEは電球アイコンを表示します。
Show Method Parameters	次のパラメータを選択します。 このショートカットを使う前に、1つのパラメータを選択(ハイライト表示)しておく必要があります。

(続き)

コマンド	動作
Show Documentation	カーソル位置のアイテムの説明を表示します。
Insert Next Matching Word	入力した文字列から始まる(不完全な)単語を、コード内で順方向に検索し、最初に見つかった単語に基づいて、入力した文字列を補完します。
Insert Previous Matching Word	入力した文字列から始まる単語を、コード内で逆方向に検索し、最初に見つかった単語に基づいて、入力した文字列を補完します。
Inspect	選択中のファイル、パッケージ、プロジェクトのいずれかに関する指定した検査を実行します。
Scan for External Changes	ファイルをスキャンして、MPLAB® X IDEの外部で加えられた変更を見つけます。

10.1.6 [Refactor]メニュー

本セクションでは[Refactor]メニューの項目を示します。メニューに表示される項目は、リファクタリングするオブジェクトのタイプ(変数、関数等)によって異なります。詳細は7.6「Cコードのリファクタリング」を参照してください。

これらのメニュー項目の一部に割り当てられているキーボード ショートカットについては[Help] > [Keyboard Shortcuts Card]を参照してください。

表10-6 [Refactor]メニューの項目

コマンド	動作
Rename	変数または関数名を変更し、プロジェクト内の全ソースコードを更新します。
Move	クラスを別のパッケージまたはクラスに移動し、プロジェクト内の全てのコードを更新します。
Copy	クラスを同じパッケージまたは別のパッケージにコピーします。
Safely Delete	コードエレメントへの参照を確認し、他のコードから参照されていなければそのエレメントを削除します。
Change Function Parameter	関数のパラメータの数と名前を変更します。
Encapsulate Fields	**C++専用:** このオプションを表示するにはC++ソースファイル内でクリックします。フィールド用のゲッターメソッドとセッターメソッドを自動的に生成し、オプションで全ての参照コードを更新して、ゲッターメソッドとセッターメソッドでフィールドにアクセスします。 以下も参照してください。 http://wiki.netbeans.org/Refactoring

10.1.7 [Production]メニュー

本セクションでは[Production]メニュー(以前の[Run]メニュー)の項目を示します。

Note: [Run]操作はこのメニューから削除されましたが、ツールバーのアイコン  として引き続き使用できます。これらのメニュー項目の一部に割り当てられているキーボード ショートカットについては[Help] > [Keyboard Shortcuts Card]を参照してください。

表10-7 [Production]メニューの項目

コマンド	動作
Build Project	プロジェクト内の全ファイルをビルドします。
Clean and Build Project	以前に生成済みのプロジェクト ファイルを全てクリーニング(削除)してから、プロジェクト内のファイルをリビルドします。

(続き)

コマンド	動作
Batch Build Project	プロジェクトの複数のコンフィグレーションをビルドします(組み込みのみ利用可)。
Set Project Configuration	プロジェクト コンフィグレーションを選択します(「既定値」を選択)。
Set Main Project	開かれているプロジェクトの一覧の中から、メイン プロジェクトを選択します。
Configuration Bits	[Configuration Bits]ウィンドウを開きます。
Check File	ファイルが規格に沿っているかチェックします(XML関連)。
Validate File	ファイルが規格に沿っているか検証します(XML関連)。
Repeat Build/Run	一時停止から再度実行します。
Stop Build/Run	実行を終了します。

10.1.8 [Debug]メニュー

下表に[Debug]メニューの項目を示します。

表10-8 [Debug]メニューの項目

コマンド	動作
Debug Project	メイン プロジェクトまたは選択したプロジェクトをデバッグします。
Discrete Debugger Operation	以下のデバッグ作業を1ステップずつ別々に実行します。 • Build for Debugging(デバッグ実行プログラムによるビルド) • Program Device for Debugging(デバッグビルドによるプログラミング) • Launch Debugger • Live Connect Debug これは、スタータキットを使ってデバッグ中に[Memory]ウィンドウの設定を変更する場合に便利です。
Finish Debugger Session	デバッグ セッションを終了します。
Pause	デバッグを一時停止します。[Continue]で再開します。
Continue	「一時停止」したデバッグを再開します。デバッグは次のブレークポイントまたはプログラムの末尾で再び停止します。
Step Over	プログラムのソースを1行実行します。 その行が関数コールである場合、呼び出した関数全体を実行後に停止します。
Step Into	プログラムのソースを1行実行します。 その行が関数コールである場合、呼び出した関数の先頭の命令文を実行した後に停止します。
Step Out	プログラムのソースを1行実行します。 現在の関数の実行を完了し、その関数の呼び出しの直後のソース行で停止します。
Step Instruction	マシン命令を1つ実行します。 その命令が関数コールの場合、その関数を実行してから呼び出し元の制御に戻ります。
Run to Cursor	プロジェクトをカーソル位置まで実行して、停止します。
Reset	デバイスをリセットします。
Set PC at cursor	プログラム カウンタ(PC)の値を、カーソル位置の行アドレスに設定します。

(続き)

コマンド	動作
Focus Cursor at PC	カーソルを現在のPCアドレス位置へ移動し、このアドレスをウィンドウの中心に表示します。
Stack>Make Callee Current	呼び出される側のメソッドを現在の呼び出しとします。この機能は、[Call Stack]ウィンドウで呼び出しを選択している場合にのみ利用できます。
Stack>Make Caller Current	呼び出す側のメソッドを現在の呼び出しとします。この機能は、[Call Stack]ウィンドウで呼び出しを選択している場合にのみ利用できます。
Stack>Pop Topmost Call	Top Of Stackの呼び出しをポップします。
Stack>Pop To Current Stack Frame	Top Of Stackから 1 フレームをポップして捨てます。
Stack>Pop Last Debugger Call	デバッグツールからの最新のコールをTop Of Stackにポップします。
Toggle Line Breakpoint	プログラム内のカーソル行のブレークポイントを設定/解除します。
New Breakpoint	指定した行、例外、メソッドに新しいブレークポイントを設定します。
New Watch	指定したシンボルをウォッチに追加します。
New Run Time Watch	指定したシンボルを、プログラム実行中に値の変化を観察するウォッチに追加します。
Disconnect from Debug Tool	MPLAB® X IDEとデバッグツール間の接続を解除します。 [Run]または[Debug]を選択すると再接続します。
Run Debugger/ Programmer Self Test	デバッグツールのセルフテストを実行します。 ツールがセルフテスト機能をサポートしている場合、このテストを実行して動作を確認できます。このためのハードウェアの設定については、そのツールの文書を参照してください。
Hardware Tool Emergency Boot Firmware Recovery	このユーティリティは、ハードウェア ツール ブート ファームウェアを工場出荷時状態に戻す場合に使います。MPLAB ICD 4またはPICkit 4のヘルプを参照してください。

10.1.9 [Team]メニュー

本セクションでは[Team]メニューの項目を示します。

これらのメニュー項目の一部に割り当てられているキーボード ショートカットについては[Help] > [Keyboard Shortcuts Card]を参照してください。

表10-9 [Team]メニューの項目

コマンド	動作
Shelve Changes	まだコミットしていない変更をパッチファイルとして作業フォルダに一時的に保存します。
Git, Mercurial, Subversion	バージョン管理システムによって表示されるサブメニューが異なります。 サブメニューの詳細は各製品のマニュアルを参照してください。
History	ファイルの履歴を表示します。ファイルを以前のバージョンに復元する事もできます。
Find Task	バージョン管理システムで問題を見つけます。
Report Task	バージョン管理システムに問題を報告します。
Create Build Job	バージョン管理システムを使ってビルドを作成します。

10.1.10 [Tools]メニュー

本セクションでは[Tools]メニューの項目を示します。

これらのメニュー項目の一部に割り当てられているキーボード ショートカットについては[Help] > [Keyboard Shortcuts Card]を参照してください。

表10-10 [Tools]メニューの項目

コマンド	動作
Embedded	追加されているプラグインを表示します。サブメニューからプラグインを選択します。
Licenses	コンパイラ ライセンスを管理します。詳細は以下の[Documentation]タブを参照してください。https://www.microchip.com/mplab/compilers • ネットワーク ライセンスのローミング • ワークステーション ライセンスのアクティベート • ライセンス ステータス • ライセンスタイプの変更
Packs	MPLAB Pack Managerを開き、プロジェクトで使っているデバイスに対応しりバージョン管理されたデバイスパックを選択します。
Templates	[Template Manager]ウィンドウを開きます。
DTDs and XML Schemas	[DTDs and XML Schemas Manager]を開きます。
Plugins	[Plugins Manager]を開きます。 詳細は以下のNetBeansヘルプトピックを参照してください。 http://wiki.netbeans.org/InstallingAPlugin
Plugins Download	選択ダイアログが表示されます。以下が可能です。 • Visit “Embedded Code Source” Website - プラグインをダウンロードし、後で[Plugin Manager]の[Downloaded]タブを使ってインストールする • Go to MPLAB X Plugin Manager - [Plugin Manager]で[Available Plugins]のリストからプラグインを直ちにインストールする
Options	[Options]ダイアログを開きます。 macOSの場合: [MPLAB X IDE] > [Preferences]を使います。 12.16「[Tools] > [Options]ウィンドウ、[Embedded]」を参照してください。

10.1.11 [Window]メニュー

本セクションでは[Window]メニューの項目を示します。

これらのメニュー項目の一部に割り当てられているキーボード ショートカットについては[Help] > [Keyboard Shortcuts Card]を参照してください。

表10-11 [Window]メニューの項目

コマンド	動作
Projects	[Projects]ウィンドウを開きます。8.1「[Projects]ウィンドウの表示」を参照してください。
Files	[Files]ウィンドウを開きます。8.2「[Files」ウィンドウの表示」を参照してください。
Classes	[Classes]ウィンドウを開きます。8.3「[Classes」ウィンドウの表示」を参照してください。
Favorites	[Favorites]ウィンドウを開きます。8.4「[Favorites]ウィンドウの表示」を参照してください。
Services	[Services]ウィンドウを開きます。 このウィンドウの詳細はNetBeansヘルプを参照してください。
Dashboard	[Dashboard]ウィンドウを開きます。 5.19「ダッシュボードの表示」を参照してください。

(続き)

コマンド	動作
Navigator	[Navigator]ウィンドウを開きます。 このウィンドウの詳細はNetBeansヘルプを参照してください。
Acton Items	[Action Items]ウィンドウを開きます。12.1「[Action Items]ウィンドウ」を参照してください。
Tasks	[Task List]ウィンドウを開きます。
Output	[Output]ウィンドウを開きます。12.13「[Output]ウィンドウ」を参照してください。
Editor	空のエディタ ウィンドウを開きます。7.1「エディタの使い方」を参照してください。
Debugging>Output	デバッグ用の[Output]ウィンドウ(Disassembly Listing File)を開きます。
Debugging>Variables	[Local Variables]デバッガ ウィンドウを開きます。4.17「ローカル変数値の変化の観察」を参照してください。
Debugging>Watches	[Watches]デバッガ ウィンドウを開きます。4.16「シンボル値の変化の観察」を参照してください。
Debugging>Call Stack	[Call Stack]デバッガ ウィンドウを開きます。5.17「コールスタックの表示」を参照してください。
Debugging>Breakpoints	[Breakpoints]ウィンドウを開きます。 4.14「ブレークポイントによるプログラム実行の制御」を参照してください。
Debugging>Sessions	[Sessions]ウィンドウを開きます。
Debugging>Sources	[Sources]ウィンドウを開きます。
Debugging>Disassembly	[Disassembly]ウィンドウを開きます。12.7「[Disassembly]ウィンドウ」を参照してください。
Debugging>PIC AppIO	[Application In/Out]ウィンドウを開きます。 MPLAB® REAL ICE™インサーキット エミュレータで使います。
Debugging>Trace	[Trace]ウィンドウを開きます。 シミュレータまたはMPLAB REAL ICEインサーキット エミュレータで使います。
Debugging>Stopwatch	[Stopwatch]ウィンドウを開きます。5.15「ストップウォッチの使用」を参照してください。
Debugging>PC Profiling	[PC Profiling]ウィンドウを開きます。 MPLAB® REAL ICE™インサーキット エミュレータで使います。
Debugging>Triggers	[Trigger In]ウィンドウを開きます。 MPLAB® REAL ICE™インサーキット エミュレータで使います。
Debugging>Debugger Console	コマンドを入力するためのコンソール ウィンドウを開きます。
Web>Web Browser	既定値のウェブブラウザを開きます。
Web>CSS	[CSS Styles]ウィンドウを開きます。このウィンドウを使うと、CSSファイル内のHTML要素とセレクタのルールの宣言を編集できます。
IDE Tools>Bookmarks	[Bookmarks]ウィンドウを開きます。開いたプロジェクトのファイルのブックマークの一覧を表示します。

(続き)

コマンド	動作
IDE Tools>Notifications	[Notifications]ウィンドウを開きます。現在のIDEセッションで発生したIDEのエラーと警告のリストを表示します。
IDE Tools>Terminal	ローカルまたはリモート用の[Terminal]ウィンドウを開きます。
IDE Tools>Processes	[Processes]ウィンドウを開きます。実行中のプロセスにデバッグツールを接続できます。
PIC Memory Views>Memory	指定した[Memory]ウィンドウを開きます。プロジェクト デバイスによって表示されるメモリが異なります。以下を参照してください。 12.9 [Memory]ウィンドウ - 8/16ビットデバイス 12.10 [Memory]ウィンドウ - 32ビットデバイス
Simulator>Stimulus	[Simulator Stimulus]ウィンドウを開きます。 詳細はMPLAB Xシミュレータのマニュアルを参照してください。
Simulator>Analyzer	[Simulator Analyzer]ウィンドウを開きます。 詳細はMPLAB Xシミュレータのマニュアルを参照してください。
Simulator>IOPin	[I/O Pins]ウィンドウを開きます。 詳細はMPLAB Xシミュレータのマニュアルを参照してください。
Simulator>Register Trace	[Simulator Register Trace]ウィンドウを開きます。 詳細はMPLAB Xシミュレータのマニュアルを参照してください。
Configure Window> Maximize	アクティブなウィンドウをIDEウィンドウの最大サイズに拡大します。
Configure Window>Float	フローティングのタブ付きIDEウィンドウを開きます。
Configure Window> Float Group	フローティングのタブ付きIDEウィンドウ グループを開きます。
Configure Window>Minimize	アクティブなウィンドウを最小化します。
Configure Window> Minimize Group	アクティブなウィンドウ グループを最小化します。
Configure Window>Dock	フローティングのタブをメインウィンドウに復元します。
Configure Window> Dock Group	フローティングのタブグループをメインウィンドウに復元します。
Configure Window> Clone Document	同じ文書に対して新規タブを開きます。
Configure Window> Split Window	アクティブな文書を垂直または水平に分割します。
Configure Window< New Document Tab Group	エディタ ウィンドウを2つのタブグループに分割し、選択したタブを新規グループに分けます。
Configure Window>Collapse Document Tab Group	独立したタブグループを1つのグループに結合します。
Reset Window	ウィンドウを既定値状態にリセットします。
Close Window	ウィンドウ内で現在表示しているタブを閉じます。ウィンドウにタブがない場合、そのウィンドウを閉じます。
Close All Documents	ソースエディタで開いている全ての文書を閉じます。
Close Other Documents	現在アクティブな文書以外の全ての文書を閉じます。

(続き)

コマンド	動作
Document Groups	名前が付いた文書グループを作成します。
Documents	[Documents]ダイアログ ボックスを開きます。このダイアログでは、現在開いている文書を保存、閉じる事ができます。

10.1.12 [Help]メニュー

本セクションでは[Help]メニューの項目を示します。

これらのメニュー項目の一部に割り当てられているキーボード ショートカットについては[Help] > [Keyboard Shortcuts Card]を参照してください。

表10-12[Help]メニューの項目

コマンド	動作
Help Contents	JavaHelpビューアを開いて、インストールされている全てのヘルプファイルを表示します。
Tool Help Contents	1つのJavaHelpビューアを開いて、1つのツール ヘルプファイルを表示します。
Release Notes	MPLABがサポートするツールのリリースノートへのリンクのリストを表示します。その他MPLAB X IDEサポート文書へのリンクも表示します。
Online Docs and Support	NetBeansのサポート ウェブページを開きます。
Keyboard Shortcuts Card	キーボード ショートカット マニュアルを表示します。
MPLAB X Store	[MPLAB X Store]タブを開きます。
Start Page	[Start Page]タブを開きます。
About	[about MPLAB® X IDE]ウィンドウを開きます。

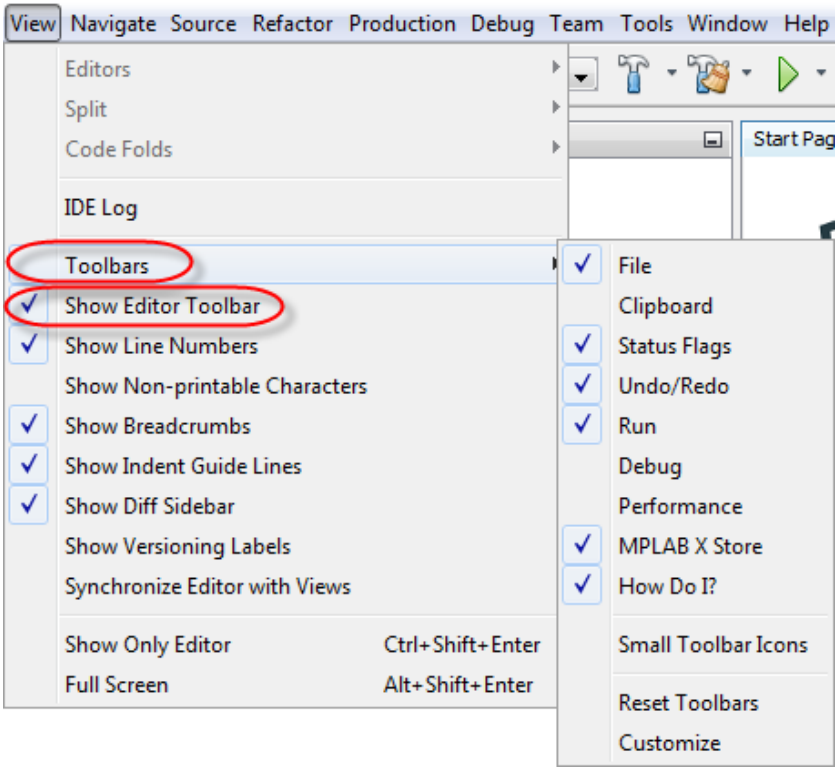
10.2 ツールバー

MPLAB X IDEは、現在使用中の機能またはツールに応じて、各種ツールバーを表示します。これらのツールバーを使うと、頻繁に行う操作に素早くアクセスできます。

以下はツールバーを操作する時に役立ちます。

- ツールバーの特長
- ツールバーのカスタマイズ
- メニュー項目とボタンの灰色表示または非表示

図10-2　View>Toolbars



10.2.1　[File]ツールバー

[File]ツールバーは以下の機能のアイコンを収めています。これらの機能は[File]メニューからも利用できます。

アイコン	アイコンテキスト	機能
	New File	[New File]ウィザードを使ってファイルを新規作成します。
	New Project	[New Project]ウィザードを使ってプロジェクトを新規作成します。
	Open Project	既存のプロジェクトを開きます。
	Save All Files	開いている全てのファイルを保存します。

関連リンク

10.2.11　ツールバーの特長
10.2.12　ツールバーのカスタマイズ

10.2.2　[Clipboard]ツールバー

[Clipboard]ツールバーは以下の機能のアイコンを収めています。これらの機能は[Edit]メニューからも利用できます。

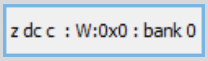
アイコン	アイコンテキスト	機能
	Cut	現在選択している内容を削除し、クリップボードに移動します。
	Copy	現在選択している内容をクリップボードにコピーします。
	Paste	クリップボードの内容を現在の位置に挿入します。

関連リンク

10.2.11 ツールバーの特長
10.2.12 ツールバーのカスタマイズ

10.2.3 [Status Flags]ツールバー

[Status Flags]ツールバーは以下の機能を収めています。

イメージ	イメージテキスト	説明
	Program Counter	プログラム カウンタ(PC)の現在値
	Status Flags	デバイスのステータスフラグ ステータスビットまたはレジスタの詳細は、デバイスのデータシートを参照してください。

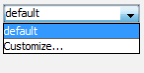
10.2.4 [Undo/Redo]ツールバー

[Undo/Redo]ツールバーは以下の機能のアイコンを収めています。これらの機能は[Edit]メニューからも利用できます。

アイコン	アイコン名	機能
	Undo	エディタ操作を1つずつ取り消します(保存操作は取り消せません)。
	Redo	[Undo]操作を1つずつ元に戻します。

10.2.5 [Run]ツールバー

[Run]ツールバーは以下の機能のアイコンを収めています。これらの機能は[Run]、[Debug]、[Project]コンテクストメニューからも利用できます。

アイコン	アイコンテキスト	機能
	Set Project Configuration	プロジェクト コンフィグレーションを選択します。[default]または[Customize]を選択します。詳細は[Project Properties]ウィンドウを参照してください。
	Build Project	全てのプロジェクト ファイルをビルドします。
	>Build for Debugging	デバッグ用に全てのプロジェクト ファイル (デバッグ実行プログラム含む)をビルドします。[Discrete Debugger Operation]のStep 1
	Clean and Build Project	前回のビルドで作成されたファイルを削除後、全てのプロジェクト ファイルをビルドします。
	>Clean and Build for Debugging	前回のビルドで作成されたファイルを削除後、デバッグ用に全てのプロジェクト ファイル (デバッグ実行プログラム含む)をビルドします。[Discrete Debugger Operation]のStep 1

(続き)

アイコン	アイコンテキスト	機能
	Run Project	選択したプロジェクトをビルドし、ターゲットをプログラミング後、プロジェクトを実行します。
	Make and Program Device Project	選択したプロジェクトをビルドし、ターゲットをプログラミング後、リセット状態に保持します。
	>Program Device for Debugging	デバッグビルドでターゲットをプログラミングします。[Discrete Debugger Operation]のStep 2
	>Program Device for Production	量産ビルドでターゲットをプログラミングします。
	>Erase Device Memory	デバイス プログラムメモリを消去します。
	>Programmer To Go PICkit3	プロジェクト デバッグツールとしてPICkit 3を使ってProgrammer To Go (PTG)でターゲットをプログラミングします。
	Read Device Memory	ターゲットデバイスのメモリの内容を読み出し、MPLAB X IDEにロードします。
	>Discrete Multi-Partition Read Operation	マルチ パーティション (デュアル パーティション) デバイスで指定されたフラッシュメモリ パーティションを読み出します。
	>Read Device Memory to a File	ターゲット デバイスメモリを読み出し、ファイルに保存します。
	>Read EE/Flash Data Memory to a File	ターゲット デバイスのデータメモリを読み出し、ファイルに保存します。
	Hold in Reset/ Release from Reset	選択したプロジェクトをリセット状態でホールドするか、リセットから解放します。
	Debug Project	選択したプロジェクトをビルドし、ターゲットをプログラミング後、デバッグします。
	>Launch Debugger	デバッガを起動し、ターゲットコードを実行します。[Discrete Debugger Operation]のStep 3
	>Live Connect Debug	MECデバイスの場合、実行中のターゲットに接続し、メモリの内容を検査します。量産デバイスでデバッグが可能です。
>: 下向き矢印アイコンから利用可能		

関連リンク

10.2.11 ツールバーの特長
10.2.12 ツールバーのカスタマイズ

10.2.6 [Debug]ツールバー

[Debug]ツールバーは以下の機能のアイコンを収めています。これらの機能は[Debug]メニューからも利用できます。

アイコン	アイコンテキスト	機能
	Finish Debugger Session	デバッグ セッションを終了します。
	Pause	デバッグを一時停止します。[Continue]で再開します。
	Reset	プロジェクトをカーソル位置まで実行して、停止します。

(続き)

アイコン	アイコンテキスト	機能
	Continue	デバッグを再開します。デバッグは次のブレークポイントまたはプログラムの末尾で再び停止します。
	Step Over	プログラムのソースを1行実行します。その行が関数コールである場合、呼び出した関数全体を実行した後に停止します。
	Step Into	プログラムのソースを1行実行します。その行が関数コールである場合、呼び出した関数の先頭の命令文を実行した後に停止します。
	Step Out	プログラムのソースを1行実行します。この行が関数コールの場合、実行後に呼び出し元に戻ります。
	Run to Cursor	実行中のプロジェクトをファイル内のカーソル位置まで実行した後にプログラムの実行を停止します。
	Set PC at cursor	プログラム カウンタ(PC)の値を、カーソル位置の行アドレスに設定します。
	Focus Cursor at PC	カーソルを現在のPCアドレス位置へ移動し、このアドレスをウィンドウの中心に表示します。

関連リンク

10.2.11「ツールバーの特長」
10.2.2「[Clipboard]ツールバー」

10.2.7 [Performance]ツールバー

[Performance]ツールバーは以下を収めています。

項目	名称	機能
	Memory Usage Display	クリックするとガーベジ コレクションを実行できます。これによりメモリを解放できます。
	Profile the Application	クリックするとアプリケーションのプロファイリングを開始できます。もう一度クリックするとプロファイリングが停止し、ファイル(.npss)に保存されます。 PCサンプリングとプロファイリングの詳細はMPLAB® REAL ICE™ インサーキット エミュレータのマニュアルを参照してください。

10.2.8 [MPLAB X Store]ツールバー

[MPLAB X Store]ツールバーは以下の機能のアイコンを収めています。MPLAB X Storeの機能はヘルプメニューにも表示されます。

アイコン	アイコンテキスト	機能
	MPLAB X Store	クリックすると、IDEに**[MPLAB X Store]**タブが開きます。製品、サービス、デバイスのご注文について確認できます。
	Programming Center	クリックすると、ブラウザタブでMicrochip社プログラミング センターの情報を表示できます。Microchip社は迅速、低コスト、セキュアで実証済みのデバイス プログラミングを提供しています。

10.2.9 [How Do I]ツールバー

この機能はインターネット接続を必要とします。

開発者ヘルプで情報を検索できます。テキストボックスに質問を入力できます。以下も参照してください。

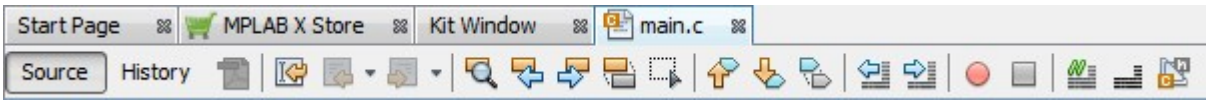
https://microchipdeveloper.com/

10.1.12[Help]メニュー

10.2.10 エディタ ツールバー

エディタのツールバーは以下の機能のアイコンを収めています。これらの機能は[Edit]および[Source]メニューからも利用できます。このツールバーは、現在のソースコード ファイルを表示しているタブの上に表示されます。

図10-3 エディタ ツールバー



アイコン	アイコンテキスト	機能
Source	Source	ソースコードを表示します。
History	History	ソースコードの編集履歴を表示します。
	Online Data Sheet	プロジェクト デバイスのデータシートを開き、ソースコードで選択(強調表示)されている周辺モジュールとSFRを検索します。 現在この機能はAVRデバイス専用です。
	Last Edited	直前に編集した行へジャンプします。
	Back	逆方向に移動します。
	Forward	順方向に移動します。
	Find Selection	選択したテキストと最初に一致するテキストを検索します。
	Find Previous Occurrence	選択したテキストを逆方向に検索します。
	Find Next Occurrence	選択したテキストを順方向に検索します。
	Toggle Highlight Search	検索に一致したテキストの強調表示をON/OFFします。
	Toggle Rectangular Selection	検索に一致したテキストの行の強調表示をON/OFFします。
	Previous Bookmark	1つ前のブックマークに移動します(逆方向に循環)。
	Next Bookmark	次のブックマークに移動します(順方向に循環)。
	Toggle Bookmark	コード行にブックマークを設定します。
	Shift Left	選択した行をタブ1つ左へシフトします。
	Shift Right	選択した行をタブ1つ右へシフトします。
	Start Macro Recording	マクロ用のキー操作の記録を始めます。

(続き)		
アイコン	アイコンテキスト	機能
	Stop Macro Recording	キー操作の記録を終了します。
	Comment	選択したコード行に「//」を追加してコメント行にします。
	Uncomment	選択したコメント行の「//」を削除してコード行にします。
	Go to Header/Source	対応するヘッダとソースコードの間を移動します。

関連リンク

10.2.11 ツールバーの特長
10.2.12 ツールバーのカスタマイズ

10.2.11 ツールバーの特長

ツールバーは以下の特長を備えています。

- カーソルをアイコンに**合わせると**機能を表示します。
- **ドラッグ操作**により、ツールバーを移動できます。
- ツールバー領域内で**右クリックして**、ツールバーの表示/非表示を切り換えたり、一部のツールバーの内容を変更したりする事ができます。
- **[View] > [Toolbars]**を選択すると、ツールバーの表示/非表示の切り換え、一部ツールバーの内容の変更、カスタム ツールバーの作成を実行できます。

10.2.12 ツールバーのカスタマイズ

[Customize Toolbars]ウィンドウを使うとMPLAB X IDEのツールバーをカスタマイズできます。このウィンドウは[View] > [Toolbars] > [Customize]で開きます。
to open the window.

[Clean Only]、[Run]、[Set PC to Cursor]等のアイコンを使えます。

ツールバーに機能を追加する方法

- [Customize Toolbars]ウィンドウからツールバーにアイコンをドラッグします。

ツールバーから機能を削除する方法

- ツールバーから[Customize Toolbars]ウィンドウにアイコンをドラッグします。

独自のツールバーを追加する方法

- [New Toolbar]をクリックして、新規ツールバーに名前を付けます。

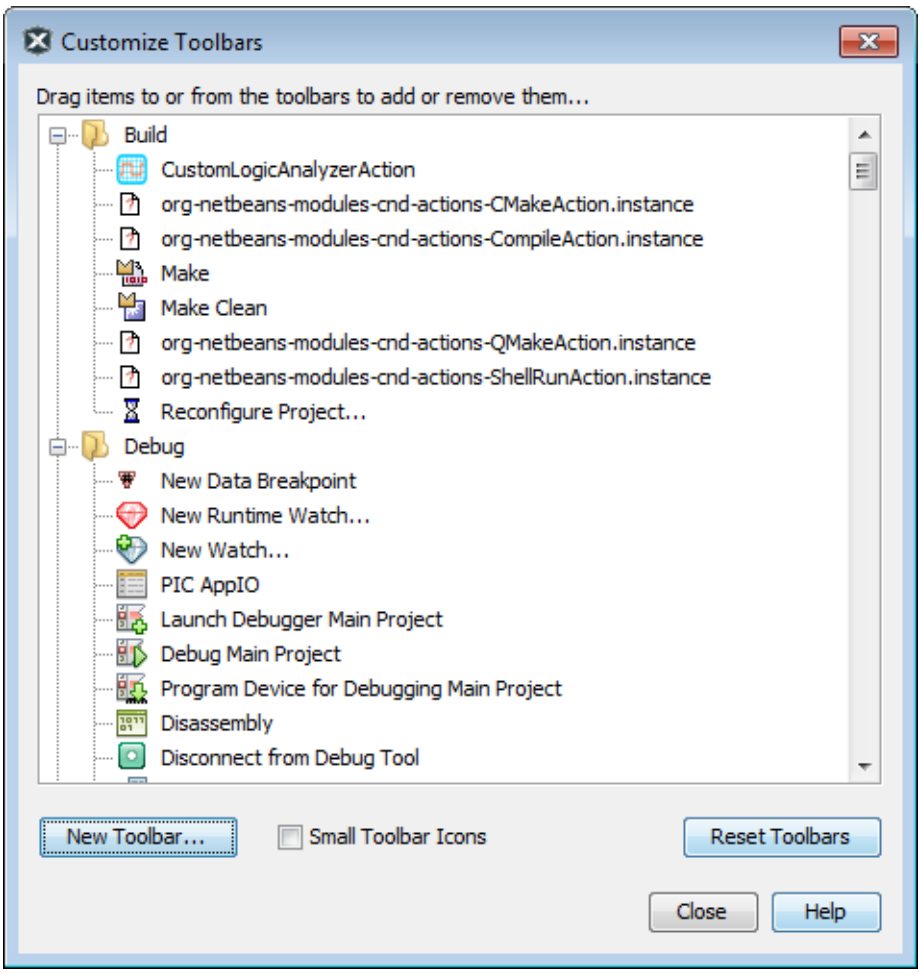
ツールバー アイコンのサイズを変更する方法

- アイコンを小さくするには、[Small Toolbar Icons]にチェックを入れます。
- アイコンを大きくするには、チェックを外します。

既定値のツールバーに戻す方法

- [Reset Toolbars]をクリックします。

図10-4　[Customize Toolbars]ウィンドウ



10.3　ステータスバー

ステータスバーは、MPLAB IDEセッションの最新のステータス情報を表示します。エディタの情報も表示します。

図10-5　デバッグ中のステータスバー

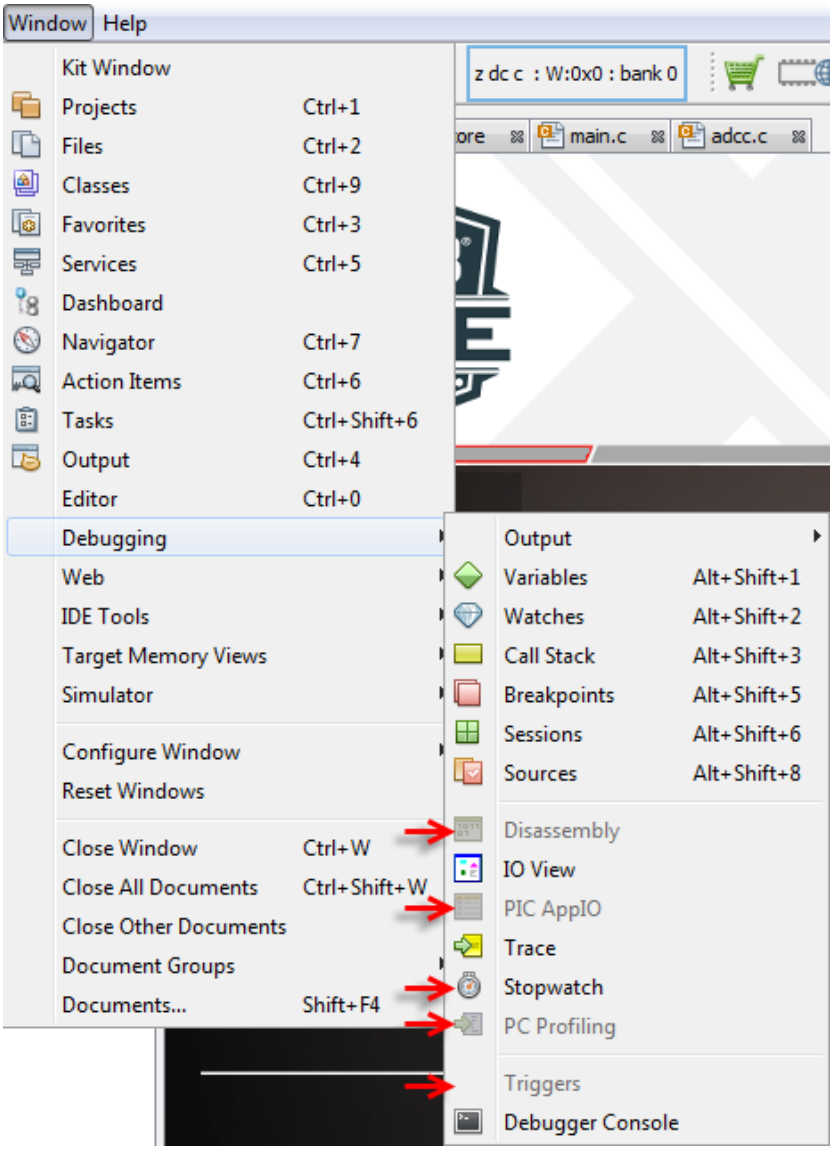
10.4　メニュー項目とボタンの灰色表示または非表示

以下のような条件では、メニュー項目、ツールバーボタン、ステータスバー アイテムは灰色で表示(無効化)されるか、表示されません。

- その項目/ボタンの機能が、選択したデバイスでは利用できないデバイス機能である場合(例: PIC16F877Aは外部メモリをサポートしない)
- その項目/ボタンの機能が、選択したツールでは利用できないツール機能である場合(例: MPLAB ICD 3では[Step Out]は使えない)
- その項目/ボタンの機能がプロジェクトに関連し、プロジェクトが何も選択されていない場合(例: アクティブなプロジェクトが存在しないと、プロジェクトのビルド機能は使えない)
- その項目/ボタンの機能が、選択したデバイスまたはツール向けにサポートされていない場合

- その項目/ボタンの機能が既に実行中である場合(例: プログラムの実行中、[Run Project]は灰色表示になる)
- その項目/ボタンの機能が他の項目/ボタンと排他的な関係にある場合(例: プログラム実行中は[Pause]が使えて[Continue]は灰色表示、プログラム停止中は[Continue]が使えて[Pause]は灰色表示になる)

図10-6　灰色表示されたメニュー項目



11. MPLAB X IDEのウィンドウの挙動

MPLAB X IDEの各ウィンドウは、アプリケーション コードの作成とデバッグを支援する機能を備えています。ただし、各ウィンドウを活用するにはこれらの機能を理解する必要があります。

11.1 MPLAB X IDEのウィンドウの管理

IDEのウィンドウの管理に関する情報は以下のNetBeansヘルプトピックを参照してください。

Managing IDE Windows

その他のウィンドウの情報は以下を参照してください。

11.1.1 ウィンドウのデータの更新

開いたウィンドウ(読み出し専用のフラッシュメモリのウィンドウを除く)は、プログラムの停止時に更新されます。プログラムの停止には、実行後の停止とステップ実行があります。停止による更新は以下の事項に影響します。

- 速度 – 更新には時間がかかります。更新時間を短縮するには、使っていないウィンドウを閉じます。
- データの上書き – 開いたウィンドウに表示されたファイルレジスタ値は、停止時に読み出されます。読み出し時のレジスタ動作は、使うデバイスのデータシートを参照してください。

11.1.2 ウィンドウのデータの変更

MPLAB X IDEのウィンドウのデータは、以下に示すように編集できる場合があります。データを編集できない場合、この情報を利用して変更する事はできません。

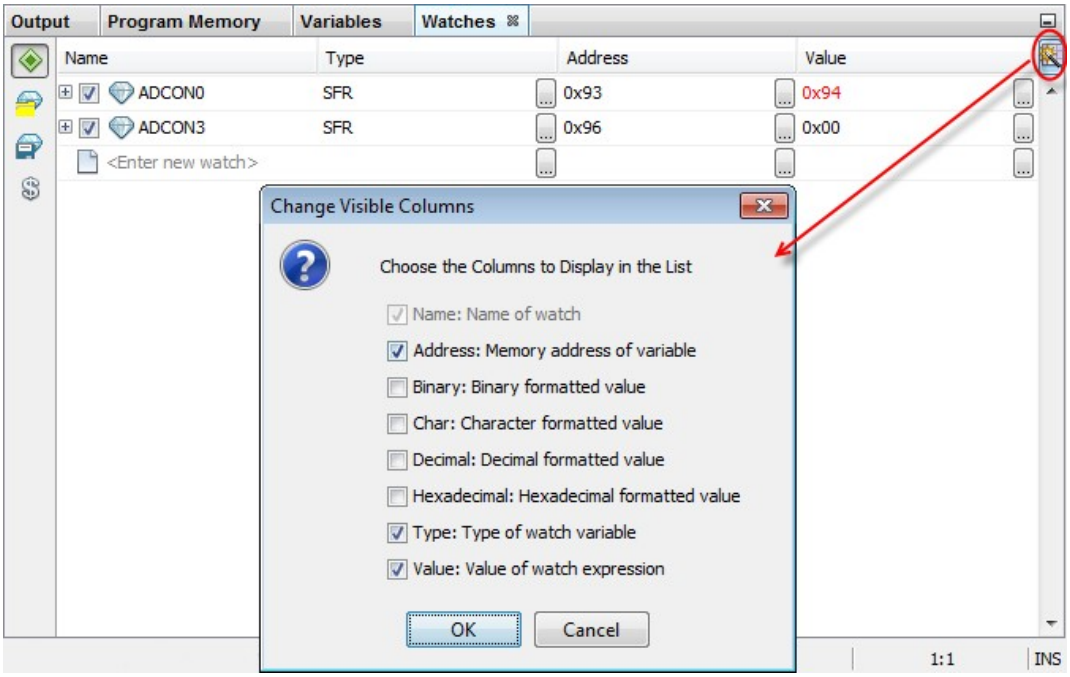
- データを直接編集する(アイテムをダブルクリックして選択し新規の値を入力します。または、アイテムの隣の省略記号[...]をクリックしポップアップ ウィンドウに新規の値を入力します。)
- データをドロップダウン リストから選択する(特定の選択肢のみが利用できる場合)

11.1.3 ウィンドウの列表示

一部のウィンドウでは、列の表示を変更できます。ウィンドウの右上隅にあるアイコンをクリックします。

	列の表示を変更します。
	表示する列を変更します。

図11-1　表示する列の変更例



11.1.4　ウィンドウのフォーカス

ウィンドウに確実にフォーカスするには、ウィンドウ フレームだけでなくボタン、表のセル、ドロップダウン、コンボボックスをクリックします。

11.2　ウィンドウに表示されるバンク化されたデータ メモリと値

この機能は、バンク化されたデータメモリ(RAM)を使う8ビットPIC MCUデバイスでのみ使えます。

MPLAB X IDE v5.20以前では、レジスタアドレスの表示値は常にバンクオフセットを含んでいます。しかし、参照している命令がプログラム カウンタ(PC)位置にある場合にのみ実際の値を正確に計算できるため、常に正しい値を得る事はできません。

MPLAB X IDE v5.25以降では、PCとレジスタアドレスが同一行にある場合のみ、レジスタアドレスはバンクオフセットを含みます。PCがその行にない場合、値はコード内の既定値に戻ります(バンクオフセットなしまたはバンク0)。

例11-1　v5.20からv5.25への挙動の変化

下図はサンプルコードのスニペットです。下表に、プログラム メモリウィンドウ(DisAssy列)または逆アセンブリ ウィンドウ内の逆アセンブリコードの表示挙動を示します。

図11-2　サンプルコードのスニペット

```
14        asm("MOVLB 0x3A"); // change to bank 58
15        asm("CLRF 0x74"); // clear common RAM register
```

表11-1　v5.20からv5.25への挙動の変化

MPLAB X IDE	15行目の逆アセンブル - PC上でない	15行目の逆アセンブル - PC上
v5.20以前	CLRF 0x1D74	CLRF 0x1D74
v5.25以降	CLRF 0x74	CLRF 0x1D74

12. MPLAB X IDEのウィンドウとダイアログ

MPLAB X IDEのデスクトップはタブ形式のウィンドウを収めた4つのセクション (ペイン)に分割されます。一部のウィンドウは特定の機能を選択するまで表示されません。一部のウィンドウには、特定の情報を入力するためのダイアログが関連付けられています。

例えば、MPLAB X IDEの初回起動時には**[Start Page]**と**[MPLAB X Store]**のタブ形式ウィンドウだけが開きます。プロジェクトを開くと、基本的なウィンドウ(左上ペインに[Projects]と[Files]、左下ペインに[Dashboard]と[Navigator]、右下ペインに[Output])が開きます。**[Start Page]**と**[MPLAB X Store]**のタブ形式ウィンドウは右上のペインに移動します。

MPLAB X IDEでは、NetBeansプラットフォームのウィンドウとMPLAB X IDE専用のウィンドウの両方を使います。各種ダイアログは、メニューから対応する項目を選択すると開きます。これらのダイアログも、ウィンドウと同様にNetBeansプラットフォームのダイアログとMPLAB X IDEのダイアログ両方です。NetBeansのウィンドウとダイアログの詳細は13.1「NetBeansウィンドウとウィンドウ メニュー」を参照してください。

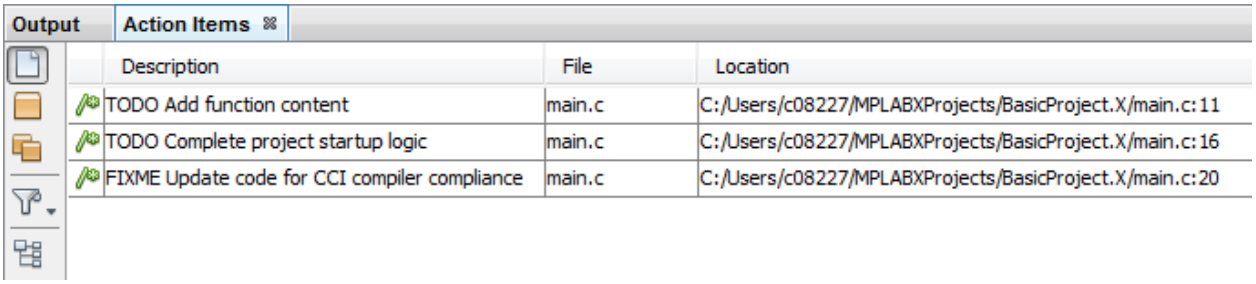
12.1 [Action Items]ウィンドウ

[Action Items]ウィンドウは[Window] > [Action Items]から開きます。

[Action Items]ウィンドウには、コンパイラの警告やエラー等各種ファイルおよびプロジェクトで解決する必要のあるアクション アイテムの一覧が表示されます。コードにコメントとしてアクション アイテムを書きとめておきます。例えば、下図では以下のコード行が表示されています。

```
//TODO Add function content
       :
//TODO Complete project startup logic
       :
//FIXME Update code for CCI compiler compliance
```

図12-1 [Action Items]ウィンドウの例



フィルタを使うと表示するアクション アイテムを絞り込め、列の見出しをクリックするとソートできます。[Action Items]ウィンドウのアイコンから表示されたアクション アイテムの変更と操作が可能です。

詳細は以下を参照してください。
http://microchipdeveloper.com/mplabx:tasks-list

12.1.1 [Action Items]ウィンドウの表示

この画面は以下の列にデータを表示します。

- [Description] – アクション アイテムの説明
- [File] – ファイル名
- [Location] – ファイルの場所

[Action Items]ウィンドウのアイコンは、ウィンドウの左側に表示されます。

表12-1　[Action Items]ウィンドウのアイコン

アイコンテキスト	アイコンテキスト	説明
	Show action items for currently edited file only	IDEは現在エディタで選択されているファイルをスキャンし、見つかったアクション アイテムを表示します。エディタで別のファイルを開くと、IDEはそのファイルのアクション アイテムのみ表示します。
	Show action items for the selected project	IDEは現在選択されているプロジェクトの全ファイルをスキャンし、プロジェクト ウィンドウに全アクション アイテムを表示します。
	Show action items for all opened projects	IDEは開いている全プロジェクトの全ファイルをスキャンし、ウィンドウに全アクション アイテムを表示します。
	Click here to select a filter	フィルタを変更または新規に設定できます。このボタンのドロップダウン リストをクリックすると、アクション アイテムのリストに適用するフィルタを選択できます。
	Group action items by category	アクション アイテムをカテゴリ別にグループ化します。

12.1.2　[Action Items]ウィンドウのメニュー

ウィンドウ内の行を右クリックすると、このメニューが表示されます。

表12-2　[Action Items]ウィンドウのコンテクスト メニュー

項目	説明*
Show	コード内のエラー/警告の位置を示します。
Scope	アイテムの範囲(現在のファイル、現在のプロジェクト、全てのプロジェクト)を示します。
Filter	フィルタを表示します。フィルタの変更もできます。
Refresh	ウィンドウの内容を更新します。
Sort By	ウィンドウの内容をソートします。

12.2　[Breakpoints]ウィンドウ

このウィンドウは[Window] > [Debugging] > [Breakpoints]を選択して開きます。

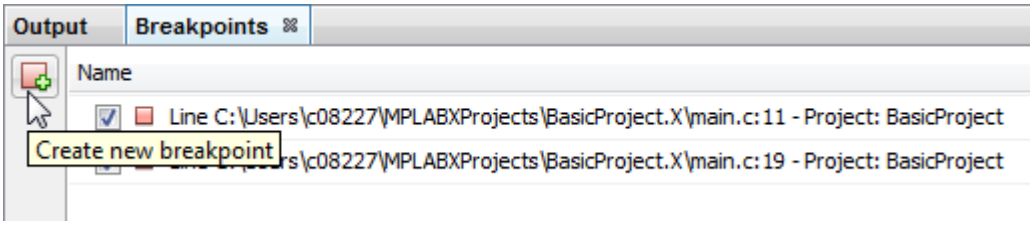
［Breakpoints］ウィンドウは、コード内のブレークポイントの管理に使います。[Breakpoints]ウィンドウの使い方の詳細は以下を参照してください。

4.14「ブレークポイントによるプログラム実行の制御」.

[New Breakpoint]ダイアログ

［Breakpoints］ウィンドウの**［Create New Breakpoint］**ボタンをクリックすると、ダイアログが開きブレークポイントを新規作成できます。4.14.2「［Breakpoint］ダイアログによるブレークポイントの設定」を参照してください。プロジェクト デバイスで使えるブレークポイント タイプの一覧が表示されます。

図12-2　［New Breakpoint］アイコン



[Breakpoints]ウィンドウのメニュー

ウィンドウの空いた場所で右クリックすると以下のメニューが表示されます。

表12-3　［Breakpoints］ウィンドウのメニュー - ブレークポイント行が選択されていない場合

項目	説明
New Breakpoint	［New Breakpoint］ダイアログを開きます。
Enable All Disable All	全てのブレークポイントを有効または無効にします。
Delete All	全てのブレークポイントを削除します。

ウィンドウ内で任意のブレークポイント行を右クリックするとメニューが表示されます。

表12-4　［Breakpoints］ウィンドウのメニュー - ブレークポイント行が選択されている場合

項目	説明
Go to Source	コード内のブレークポイントの場所を表示します。
Complex Breakpoint	複合ブレークポイントの場合、このブレークポイントを新しいシーケンスまたはタプルに追加します。シーケンスまたはタプル内の場所も指定します。
Disable Enable	このブレークポイントの動作を無効または有効にします。
New Breakpoint	［New Breakpoint］ダイアログを開きます。
Enable All Disable All	全てのブレークポイントを有効または無効にします。
Delete Delete All	このブレークポイントまたは全てのブレークポイントを削除します。
Customize	ブレークポイントの動作をカスタマイズします。

12.3　[New Breakpoint]ダイアログ

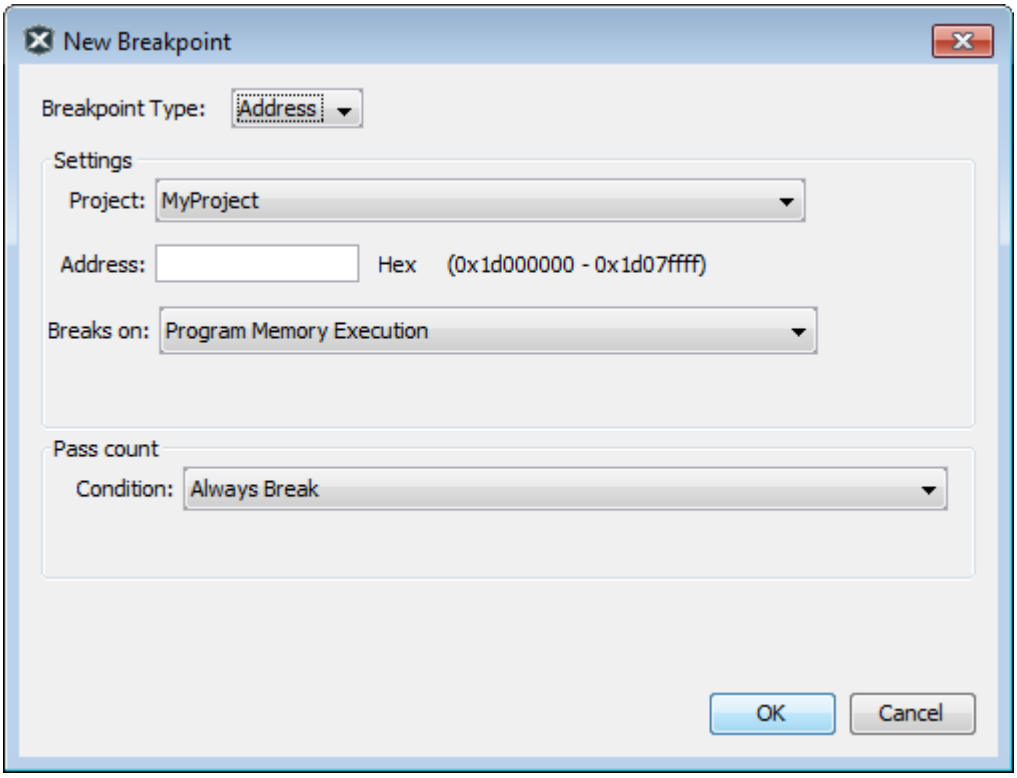
このダイアログは［Breakpoints］ウィンドウから開きます([Window] > [Debugging] > [Breakpoints])。[New Breakpoint]アイコン  をクリックするか、コンテクスト メニューから[New Breakpoint]を選択します。

ブレークポイントで使えるオプションは、選択したブレークポイントのタイプとデバイスで決まります。デバイスによって使えないオプションがあります。

ブレークポイントには以下のタイプがあります。

- 12.3.1 「行ブレークポイント」
- 12.3.2 「データ ブレークポイント」
- 12.3.3 「アドレス ブレークポイント」
- 12.3.4 「イベント ブレークポイント」
- 12.3.5 「関数ブレークポイント」
- 12.3.6 「パスカウント動作」

図12-3　［New Breakpoint］ダイアログ - ［Address］



12.3.1　行ブレークポイント

コード行で指定するブレークポイントには、以下のオプションを利用できます。

表12-5　[Breakpoint Type]: [Line] – [Settings]

項目	説明
Settings	ファイル行の隣の[ガーター]をクリックして、新規の行ブレークポイントを作成します。または、エディタのコンテクスト メニューから[Toggle Line Breakpoint]を選択します。

12.3.2　データ ブレークポイント

データメモリのブレークポイントには、以下のオプションが利用できます。

表12-6　[Breakpoint Type]: [Data] – [Settings]

項目	説明
Project	開いているプロジェクトをドロップダウン リストから選択します。 このプロジェクトのコードにブレークポイントが設定されます。
Symbols	グローバル シンボルまたはSFRの名前を入力します。または、**[Symbols]**をクリックして参照します。[Breaks on]に従ってこのシンボルをアクセスすると、コード実行の一時停止がトリガされます。
Enable Range Address	チェックを入れるとレンジ ブレークポイントが設定されます。チェックを外すとシングル ブレークポイントが設定されます。
Address **Address (Start)**	[Enable Range Address]の選択に応じて、この項目は[Address]と[Address (Start)]のどちらかになります。 データメモリの16進アドレスを入力します。 [Breaks on]に従ってこのアドレスをアクセスすると、コード実行の一時停止がトリガされます。

(続き)	
項目	**説明**
Address (End)	[Enable Range Address]にチェックを入れると、この欄は有効になります。データメモリの16進アドレスを入力します。
Breaks on	**［Read］, ［Write］, ［Read or Write］**: 上で宣言したシンボルまたはアドレスに読み込み、書き込み、読み込みまたは書き込みのいずれかを実行した場合、ブレークコードが実行されます。 **［Read Specific Value], [Write Specific Value], [Read or Write Specific Value]**: 上で宣言したシンボルまたはアドレスに、以下に示す特定の値の読み込み、書き込み、読み込みまたは書き込みのいずれかを実行した場合、ブレークコードが実行されます。 **Note:** dsPIC DSCでは、XおよびYバスに対する読み出しおよび書き込みがあります。
Value	[Breaks on]に[Read Specific Value], [Write Specific Value], [Read or Write Specific Value]のいずれかを選択する場合、ここに16進値を入力します。
Value Comparison	PIC16F1xxxデバイスの場合のみ以下に示すように「Value」と比較します。 **= Value**: 「Value」と等しい **!= Value**: 「Value」と等しくない **> Value**: 「Value」より大きい **< Value**: 「Value」より小さい
Data Value Mask	PIC16F1xxxデバイスの場合のみ 「Value」と比較する際にマスクを使います。 0x00～0xhhのレンジの値を入力します。0x00: どのビットも比較しません。 0xhh: 全てのビットを比較します。

表12-7　[Breakpoint Type]: [Data] – [Pass Count]

項目	説明
Condition	[Breaks on]で指定したブレークが発生する条件を設定します。 **Always Break**: [Breaks on]条件が成立すると常にブレークします。 **Break occurs Count instructions after Event**: [Breaks on]条件発生後、ブレーク前にCount命令を実行します。 **Event must occur Count times**: ブレーク前に、[Breaks on]条件がCountで指定した回数発生する必要があります。
Count	[Condition]で指定した条件に従って、イベント後の命令数かイベント数を入力します。12.3.4 「イベント ブレークポイント」を参照してください。
Trigger Options	PIC16F1xxxデバイスの場合のみトリガする条件を選択します。 • Do not trigger out when breakpoint is reached • trigger out when breakpoint is reached
Interrupt Context	PIC16F1xxxデバイスの場合のみ アドレス/データ ブレークポイント用の割り込みコンテクスト修飾子です。以下のいずれかを選択します。 • Always break - ISRとメインコードの両方でブレークします。 • Break in main line (non-interrupt) context only - メインコード内のみでブレークします。 • Break in interrupt context only - ISRコード内でのみブレークします。

12.3.3 アドレス ブレークポイント

プログラム/実行メモリのブレークポイントには以下のオプションを利用できます。

表12-8 [Breakpoint Type]: [Address] – [Settings]

項目	説明
Project	開いているプロジェクトをドロップダウン リストから選択します。 このプロジェクトのコードにブレークポイントが設定されます。
Enable Range Address	チェックを入れるとレンジ ブレークポイントが設定されます。チェックを外すとシングル ブレークポイントが設定されます。
Address **Address (Start)**	[Enable Range Address]の選択に応じて、この項目は[Address]と[Address (Start)]のどちらかになります。 データメモリの16進アドレスを入力します。 [Breaks on]に従ってこのアドレスをアクセスすると、コード実行の一時停止がトリガされます。
Address (End)	[Enable Range Address]にチェックを入れると、この欄は有効になります。データメモリの16進アドレスを入力します。
Breaks on	**Program Memory Execution**: 上で指定したアドレスに達した場合、ブレークコードが実行されます。 **TBLRD Program Memory**: 上で指定したアドレスへのテーブル読み出しが発生した場合、ブレークコードが実行されます。 **TBLWT Program Memory**: 上で指定したアドレスへのテーブル書き込みが発生した場合、ブレークコードが実行されます。

表12-9 [Breakpoint Type]: [Data] – [Pass Count]

項目	説明
Condition	[Breaks on]で指定したブレークが発生する条件を設定します。 **Always Break**: [Breaks on]条件が成立すると常にブレークします。 **Break occurs Count Instructions after Event**: [Breaks on]条件発生後、ブレークの前にCount命令を実行します。 **Event must occur Count times**: ブレークの前に、[Breaks on]条件がCountで指定した回数発生する必要があります。
Count	[Condition]で指定した条件に従って、イベント後の命令数かイベント数を入力します。12.3.4 「イベント ブレークポイント」を参照してください。
Trigger Options	PIC16F1xxxデバイスの場合のみトリガする条件を選択します。 • Do not trigger out when breakpoint is reached • trigger out when breakpoint is reached
Interrupt Context	PIC16F1xxxデバイスの場合のみ アドレス/データ ブレークポイント用の割り込みコンテクスト修飾子です。以下のいずれかを選択します。 • Always break - ISRとメインコードの両方でブレークします。 • Break in main line (non-interrupt) context only - メインコード内のみでブレークします。 • Break in interrupt context only - ISRコード内でのみブレークします。
「一部のデバイス(PIC16F1xxx MCU)では、拡張イベント ブレークポイントが使えます。」のセクションも参照してください。	

12.3.4 イベント ブレークポイント

イベントのブレークポイントには以下のオプションを利用できます。

表12-10 [Breakpoint Type]: [Event]

項目*	説明
Project	開くプロジェクトをドロップダウン リストから選択します。 このプロジェクトのコードにブレークポイントが設定されます。
Break on clock mode switch	クロックモードが切り換わった時にブレークします。
Break on Reset instruction	デバイスリセット命令が発生した時にブレークします。
Break on Sleep	スリープに移行した時にブレークします。
Break on stack over/underflow	スタック オーバーフロー/アンダーフロー時にブレークします。
Break on wake up	デバイスがスリープから復帰した時にブレークします。
Break when watchdog timer has expired	ウォッチドッグ タイマ時間が終了した時にブレークします。
Break on execution out of bounds	プログラムが通常のプログラムメモリ/空間を逸脱しようとした時にブレークします。
Break on MCLR Reset	マスタクリアリセット(MCLR)でブレークします。
Break on trigger in signal	トリガインパルスを検出した時にブレークします。
* 一時停止(ブレーク)するコード実行を引き起こすイベントをリストから選択します。デバイスによって利用できないイベントがあります。	

一部のデバイス(PIC16F1xxx MCU)では、拡張イベント ブレークポイントが使えます。

操作	説明
Break	指定した設定に従って実行をブレーク(停止)します。
Trigger out	指定した設定に従ってトリガアウト パルスを送出します。
Break and trigger out	指定した設定に従って実行をブレーク(停止)し、トリガアウト パルスを送出します。

12.3.5 関数ブレークポイント

関数のブレークポイントには以下のオプションを利用できます。

表12-11 [Breakpoint Type]: [Function] – [Settings]

項目	説明
Symbol/Hex Address	関数のシンボルまたはアドレスを入力または選択します。

表12-12 [Breakpoint Type]: [Function] – [Conditions]

項目	説明
Condition	チェックを入れて有効にします。その後、ブレーク条件を入力します。
Break when hit count exceeds	チェックを入れて有効にします。その後、関数の入力回数を整数で入力します。この回数後にブレークが発生します。

表12-13 [Breakpoint Type]: ［Function］ – ［Actions］

項目	説明
Suspend	ブレーク時は以下の条件でサスペンドします。 **No thread (continue)**: プログラムはサスペンドしません。 **Breakpoint thread**: 関数スレッドのみサスペンドします。 **All threads**: メインプログラムではない全てのスレッドをサスペンドします。

(続き)	
項目	**説明**
Thread ID	ブレークポイント スレッドのIDを設定します。
Print Text	ブレーク発生時に[Output]ウィンドウに表示するテキストを入力します。

12.3.6 パスカウント動作

パスカウントを使う事で、指定したカウントの後までブレークの実行を遅延させる事ができます。

Break occurs Count Instructions after Event

Countは、ブレークポイントの後、停止までに実行する命令数です。例えば、

- 0: 即座に実行を停止する
- 1: 追加の1命令後に実行を停止する
- 10: 追加の10命令後に実行を停止する

Event must occur Count times

Countは、停止までに発生するイベント回数です。例えば、

- 0: 即座に実行を停止する
- 1: イベントが1回発生したら、次回のイベント(2回目のイベント発生)で実行を停止する
- 10: イベントが10回発生したら、次回のイベント(11回目のイベント発生)で実行を停止する

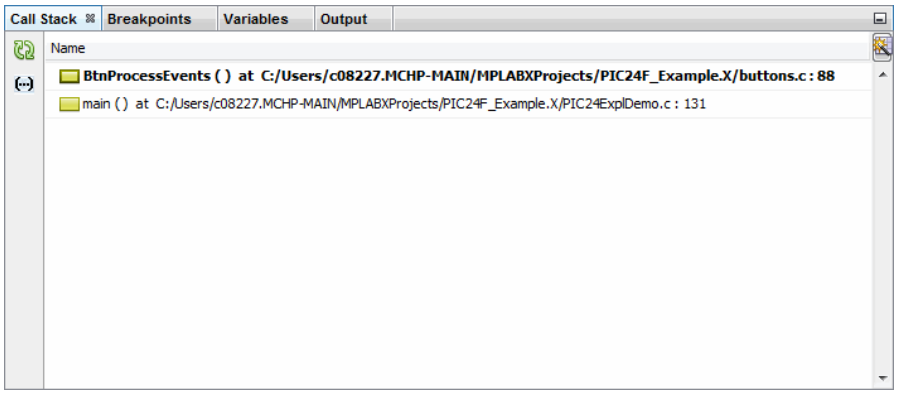
12.4 [Call Stack]ウィンドウ

[Window] > [Debugging] > [Call Stack]を選択すると[Call Stack]ウィンドウが開きます。内容はデバッグモード中にのみ表示できます。

[Call Stack]ウィンドウは、C関数とその引数を、実行中のプログラムで呼び出された順番に一覧表示します。最適化していないCコードが最も適しています。

コールスタックは16ビットおよび32ビットデバイスで使えます。このウィンドウは、デバッグの一時停止または停止時に更新されます。

図12-4 [Call Stack]ウィンドウ



一時停止または停止時にこのウィンドウで使えるアイコンは以下の通りです。

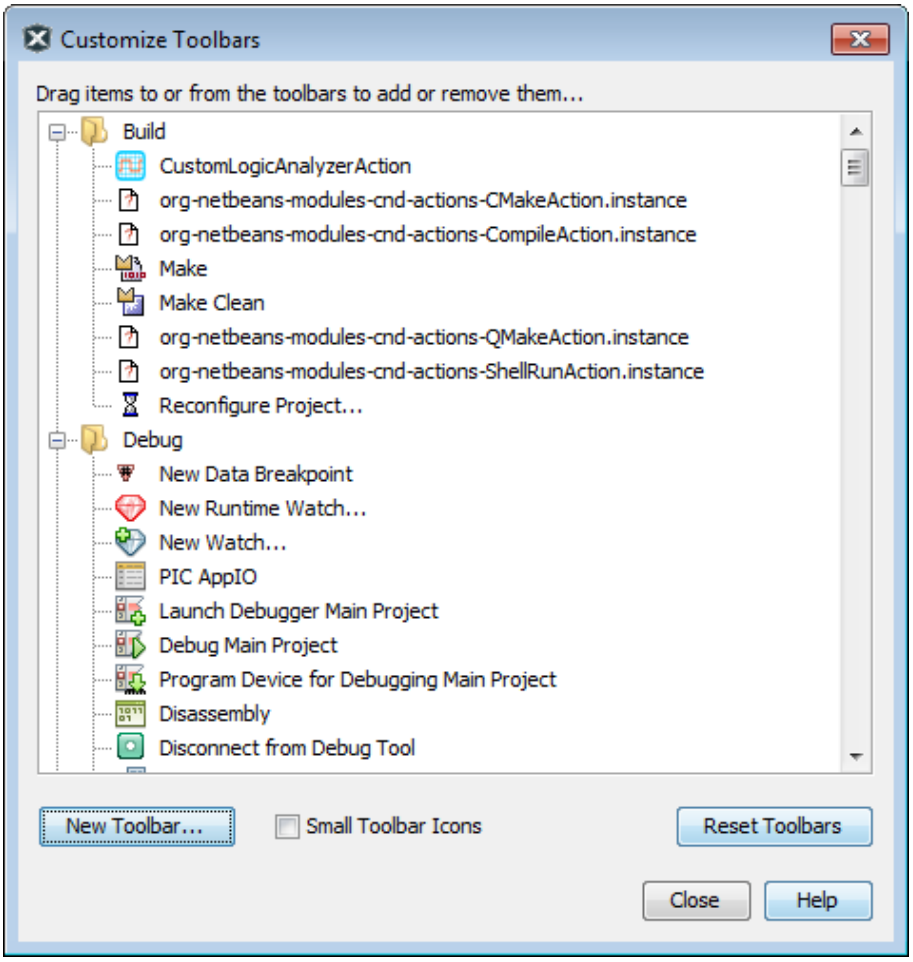
表12-14 [Call Stack]アイコン

アイコン	説明
	ウィンドウの内容を自動または手動で更新します。[Tools] > [Options] > [Embedded] > [Generic Settings]の[Disable auto refresh for call stack view during debug sessions]の値(チェックを外す = 偽(既定値)、チェックを入れる = 真)の影響を受けます。12.16.1「[Generic Settings]タブ」を参照してください。 偽 = 自動更新(緑のアイコン): ウィンドウの内容は、一時停止または停止時に自動更新されます。ウィンドウが閉じられている場合、またはフォーカスされていない場合、ウィンドウを選択しこのボタンをクリックすると更新が表示されます。再度実行および一時停止/停止する必要はありません。 真 = 手動更新(オレンジのアイコン): ウィンドウの内容は、一時停止または停止しても自動では更新されません。一時停止/停止時にウィンドウの内容を更新するには、このボタンをクリックする必要があります。
	関数のパラメータ変数の評価の無効化/有効化を選択します。
	表示する列を変更します。このウィンドウでは、コールスタック フレーム位置の表示/非表示を切り換える事ができます。

12.5 [Customize Toolbars]ウィンドウ

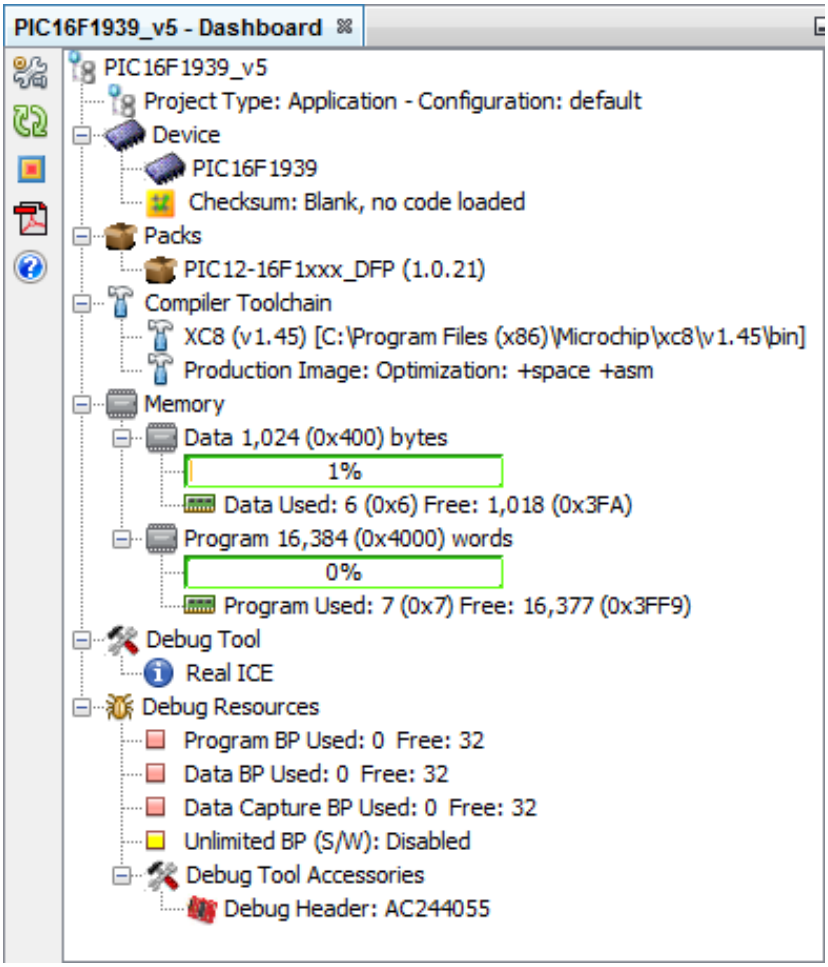
[Customize Toolbars]ウィンドウを使うとMPLAB X IDEのツールバーをカスタマイズできます。このウィンドウは[View] > [Toolbars] > [Customize]と選択して開きます。
to open the window.

カスタマイズ手順は10.2.12「ツールバーのカスタマイズ」を参照してください。



12.6 [Dashboard]ウィンドウ

[Dashboard]ウィンドウはチェックサム、コンパイラ バージョン、メモリ使用率、ブレークポイント リソース等の一般的なプロジェクト情報を表示します。詳細は以下を参照してください。5.19「ダッシュボードの表示」



12.7 [Disassembly]ウィンドウ

[Disassembly]ウィンドウは逆アセンブルCコードを表示します。このウィンドウは[Window] > [Debugging] > [Disassembly]と選択して開きます。デバッグモード中である事が必要です。

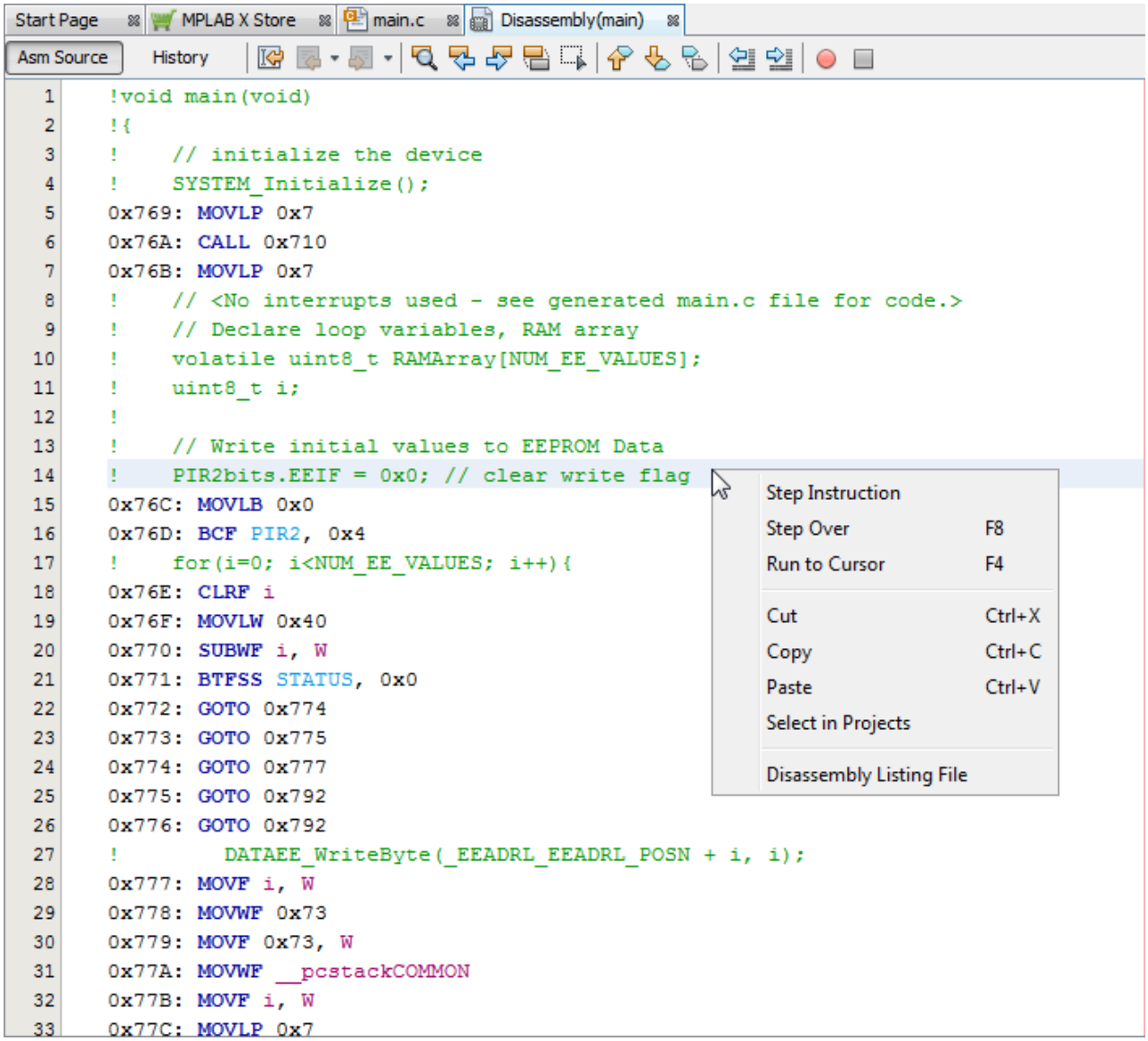
デバッグ中は「Step Over」関数がソースコード コンテクストで動作します。逆アセンブル デバッグには「Step In」関数を使います。逆アセンブル デバッグ中に「Step In」が不要な場合(例えばCALL命令をステップインせずにステップオーバする必要がある場合)、「Run to Cursor」機能を使います。PCを進ませたい行にカーソルを置き、「Run to Cursor」/F4ボタンを押します。

ウィンドウ内で右クリックすると、コンテクスト メニューが表示されます。

逆アセンブル ファイル全体を素早く表示するには、[Disassembly Listing File]を選択します。プロジェクト デバイスがRAMバンクを使う場合、以下も参照してください。

11.2「ウィンドウに表示されるバンク化されたデータメモリと値」

図12-5　[Disassembly]ウィンドウ



12.8　[I/O View]ウィンドウ

[I/O View]ウィンドウ([Window] > [Debugging] > [IO View])を使うと、現在のプロジェクトのターゲット デバイスのレジスタ概要を確認できます。操作の概要は以下を参照してください。

5.20「プロジェクト レジスタの表示([I/O View])」

[I/O View]ウィンドウのセクション

このウィンドウは既定値で上下に分割されており、上のセクションに周辺モジュール グループ、下のセクションにレジスタを表示します。通常、各周辺モジュールは定義済みの設定と値のエニュメレーションを格納し、周辺モジュール画面(上部セクション)でレジスタを展開するとこれらを表示できます。レジスタ画面(下側)は、選択した周辺モジュール グループに属する全てのレジスタを表示します。周辺モジュール未選択時、画面には何も表示されません。レジスタを展開して、そのレジスタに属する定義済み値グループを表示する事もできます。

図12-6　[I/O View]ウィンドウの内容

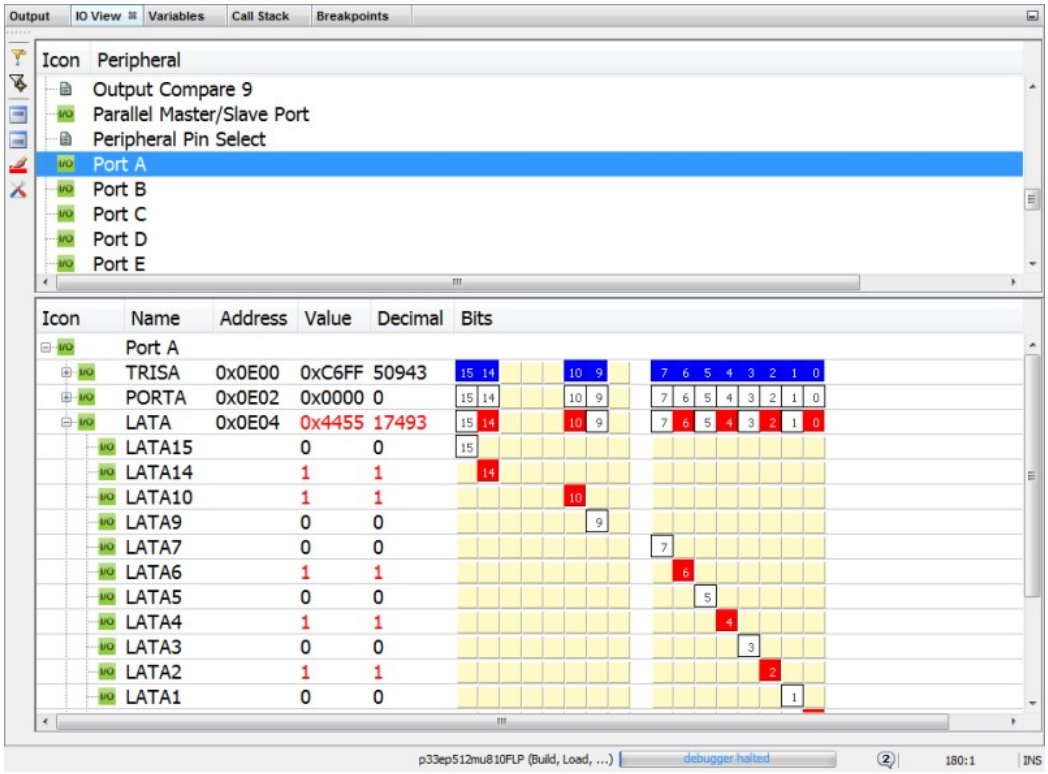


表12-15 [I/O View]のアイコン

アイコン	機能
	グループにフィルタをかける(利用可能な周辺モジュールのグループを絞り込んで表示) アイコンをクリックすると[Select]ダイアログが開きます。<Shift>+クリック(連続した複数モジュール)または<Ctrl>+クリック(個別モジュール)で周辺モジュールを選択します。または、「Register Text Filter」ファイルを指定する事もできます。
	全グループを選択するフィルタ
	周辺モジュールペインをトグルする 周辺モジュール(上部)ペインの表示/非表示を切り換えます。
	レジスタペインをトグルする レジスタ(下部)ペインの表示/非表示を切り換えます。
	標準ビットを表示する(変更されたビットのハイライト表示をトグルする) 変更されたビットが標準ビット(ハイライトなし)で表示されます。 青線アイコンをクリックして、赤線アイコンに変更します。
	変更されたビットをハイライト表示する(標準ビットをトグルする) 変更されたビットがハイライト表示されます。 赤線アイコンをクリックして、青線アイコンに変更します。

(続き)	
アイコン	**機能**
	変更されたビットのハイライト表示をトグルする 前のアイコンを「変更されたビットをハイライト表示する」として選択した場合、このアイコンをクリックすると変更されたビットのハイライト表示をトグルできます。 [IO View Settings]ダイアログが開きます。

表12-16 [I/O View]のコンテクスト メニュー

項目	機能
Expand All	ペイン/セクションを全て展開します。
Expand Row	選択した行を展開します。
Collapse All	ペイン/セクションを全て折り畳みます。
Adjust Table Columns	レジスタペインのみ(テーブル列の幅を調整します)

12.9 [Memory]ウィンドウ - 8/16ビットデバイス

[Memory]ウィンドウ([Window] > [Target Memory Views])は各種デバイスメモリ(SFR、コンフィグレーション ビット等)を表示します。このウィンドウは[Memory]および[Format]ドロップダウン ボックスを使ってカスタマイズできます。

詳細は4.18「デバイスメモリの表示と変更」を参照してください。

関連ダイアログの詳細は12.11「[Memory]ウィンドウのダイアログ」を参照してください。

図12-7 [Window] > [Target Memory Views] - [PIC16F1939]

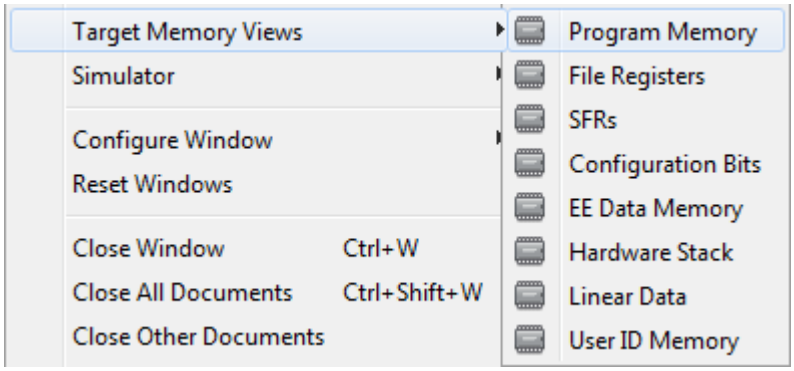
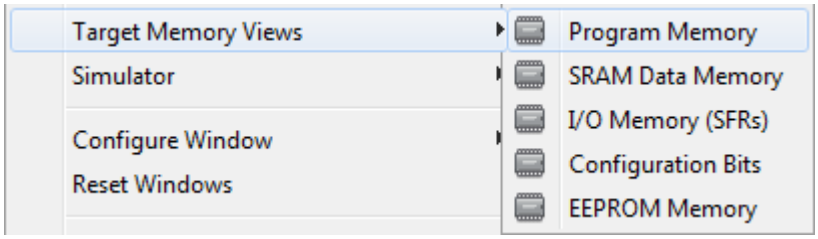


図12-8 [Window] > [Target Memory Views] - [ATtiny817]



12.9.1 [Program Memory]ウィンドウ

[Program Memory]ウィンドウは、現在選択しているプロセッサのプログラムメモリ レンジ内の位置を表示します。選択したデバイスが外部プログラムメモリをサポートしており有効な場合、その外部プログラムメモリも[Program Memory]ウィンドウに表示されます。

MPLAB Xのシミュレータでは、プログラムメモリの値が変化した場合またはプロセッサが停止した場合、[Program Memory]ウィンドウのデータが更新されます。

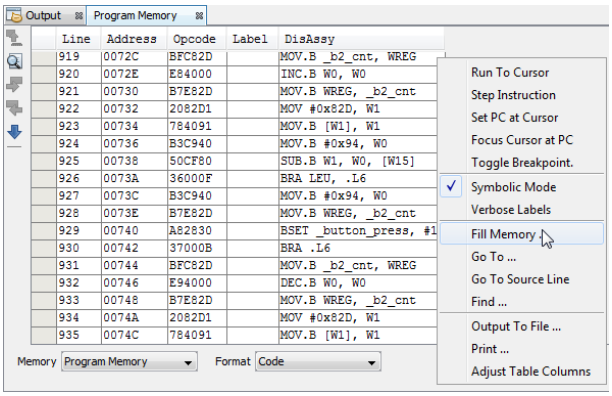
Microchip社のハードウェア デバッグツール(例: MPLAB ICD 4インサーキット デバッガ)では、プログラムメモリの値が変化した場合、またはプロセッサが停止した場合、[Program Memory]ウィンドウのデータは更新されません。デバイスメモリの読み出しを実行する必要があります。

デバッグ用ハードウェア ツールを使う場合、一部のレジスタは予約済みリソースを表すためにデータのニブルごとに「R」を表示する事があります。

プロジェクト デバイスがRAMバンクを使う場合、以下も参照してください。

11.2「ウィンドウに表示されるバンク化されたデータメモリと値」

図12-9　[Program Memory]ウィンドウの内容



12.9.1.1　[Program Memory]ウィンドウの表示

[Program Memory]ウィンドウ下部の[Format]ドロップダウン ボックスから選択する事で、ウィンドウ内のメモリの表示方法を指定できます。

デバッグ用ハードウェア ツールを使う場合、一部のレジスタは予約済みリソースを表すためにデータのニブルごとに「R」を表示する事があります。

[Code]フォーマット

[Code]フォーマットは逆アセンブルHEXコードをシンボルで表示します。このウィンドウは以下の列を表示します

- デバッグ情報 – デバッグに便利な情報です。
  ポインタはプログラム カウンタの現在の位置を示します。
- Line – 参照行番号です。
- Address – オペコードの16進アドレスです。
- Opcode – 16進オペコードです。2バイトまたは3バイトのブロックで表示されます。ほとんどのPIC MCUでは、これらのブロックはワードを意味します。
  PIC18CXXXデバイスではブロックは2バイトに相当します。dsPIC DSCデバイスではブロックは3バイトに相当します。
- Label(シンボリックのみ) – シンボリック フォーマットのオペコードラベルです。
- DisAssy – オペコード ニーモニックの逆アセンブル バージョンです。

[Hex]フォーマット

このフォーマットはプログラムメモリ情報をHEXコードで表示します。このウィンドウは以下の列を表示します

- Address – 隣の列のオペコードの16進アドレスです。
- オペコード ブロック – 16進オペコードです。2バイトまたは3バイトのブロックで表示されます。
  大部分のPIC MCUでは、これらのブロックはワードを意味します。PIC18CXXXデバイスではブロックは2バイトに相当します。dsPIC DSCデバイスではブロックは3バイトに相当します。
  ハイライト表示のオペコード ブロックはプログラム カウンタの現在の位置を表します。
- ASCII – 対応するオペコード行のASCII表示です。

PSV Mixed (dsPIC DSC/PIC24デバイスのみ)

このフォーマットはプログラムメモリをオペコードとPSV領域(CORCONレジスタ、PSVビットセット)で表示します。このウィンドウは以下の列を表示します

- デバッグ情報 – デバッグに便利な情報です。ポインタはプログラム カウンタの現在の位置を示します。
- Line – 参照行番号です。
- Address – オペコードの16進アドレスです。
- PSV Address – オペコードの16進データ空間アドレスです。
- Data – データとしてフォーマット化されたオペコードです。
- Opcode – 16進オペコードです。3バイトのブロックで表示されます。
- Label – シンボリック フォーマットのオペコードラベルです。
- DisAssy – オペコード ニーモニックの逆アセンブル バージョンです。

詳細は『dsPIC30Fファミリ リファレンス マニュアル』(DS70046)を参照してください。

PSV Data (dsPIC DSC/PIC24デバイスのみ)

このフォーマットはプログラムメモリをファイルレジスタで表示します(プログラム空間がデータ空間(CORCONレジスタ、PSVビットセット)内に表示されている場合)。このウィンドウは以下の列を表示します

- Address – データの16進プログラム空間アドレスです。
- PSV Address – データの16進データ空間アドレスです。
- データブロック – 16進データです。3バイトのブロックで表示されます。
  ハイライト表示のデータブロックはプログラム カウンタの現在の位置を表します。
- ASCII – 対応するデータ行のASCII表示です。

詳細は『dsPIC30Fファミリ リファレンス マニュアル』(DS70046)を参照してください。

12.9.1.2 [Program Memory]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-17 [Program Memory]ウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Refresh by Read Device Memory	デバッグツール バーの[Read Device Memory]アイコンと同じ機能 - デバイスメモリをMPLAB X IDEにロードします。
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Go To	指定した行番号またはアドレスに移動します。

12.9.1.3 [Program Memory]ウィンドウのメニュー

このメモリウィンドウの[CODE]または[PSV MIXED]フォーマットに設定されているデータ領域で右クリックすると、このメニューが表示されます。デバイスによっては表示されない項目があります。

表12-18 [Program Memory]ウィンドウのメニュー - Code/PSV Mixed

項目	説明
Run to Cursor	現在のカーソル位置までプログラムを実行します。
Step Instruction	ステップ命令を1つ実行します。
Set PC at Cursor	プログラム カウンタ(PC)を現在のカーソル位置に設定します。
Focus Cursor at PC	カーソルを現在のPCアドレス位置へ移動し、このアドレスをウィンドウの中心に表示します。

(続き)

項目	説明
Toggle Breakpoint	既存のブレークポイントをトグルします。
Symbolic Mode	逆アセンブルHEXコードをシンボルで表示します。
Verbose Labels	**[PSV Mixed]フォーマットのみ** 内部コンパイララベルを表示します。
Fill Memory	メモリの[Start Address]～[End Address]に[Data]内の値を書き込みます。[Fill Memory]ダイアログで、その他のオプションを指定します。
Go To	[Go To]ダイアログで指定したアドレス/関数に移動します。
Go To Source Line	エディタでソースコード内の対応する行に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	ウィンドウの表示内容をテキストファイルに書き出します。
Print	ウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

メモリウィンドウの[HEX]または[PSV DATA]フォーマットに設定されているデータ領域で右クリックすると、このメニューが表示されます。デバイスによっては表示されない項目があります。

表12-19 [Program Memory]ウィンドウのメニュー - Hex/PSV Data

項目	説明*
Hex Display Width	**[Hex]フォーマットのみ** 16進数の表示幅を設定します(選択肢はデバイスによって異なります)。
Fill Memory	メモリの[Start Address]～[End Address]に[Data]内の値を書き込みます。[Fill Memory]ダイアログで、その他のオプションを指定します。
Go To	[Go To]ダイアログで指定したアドレス/関数に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Import Table	[Import Table]ダイアログを使って、ファイルから[Memory]ウィンドウに表形式のデータをインポートします。
Export Table	[Export Table]ダイアログを使って、[Memory]ウィンドウからファイルへ表形式のデータをエクスポートします。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの[General]タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.9.2 [File Registers]ウィンドウ

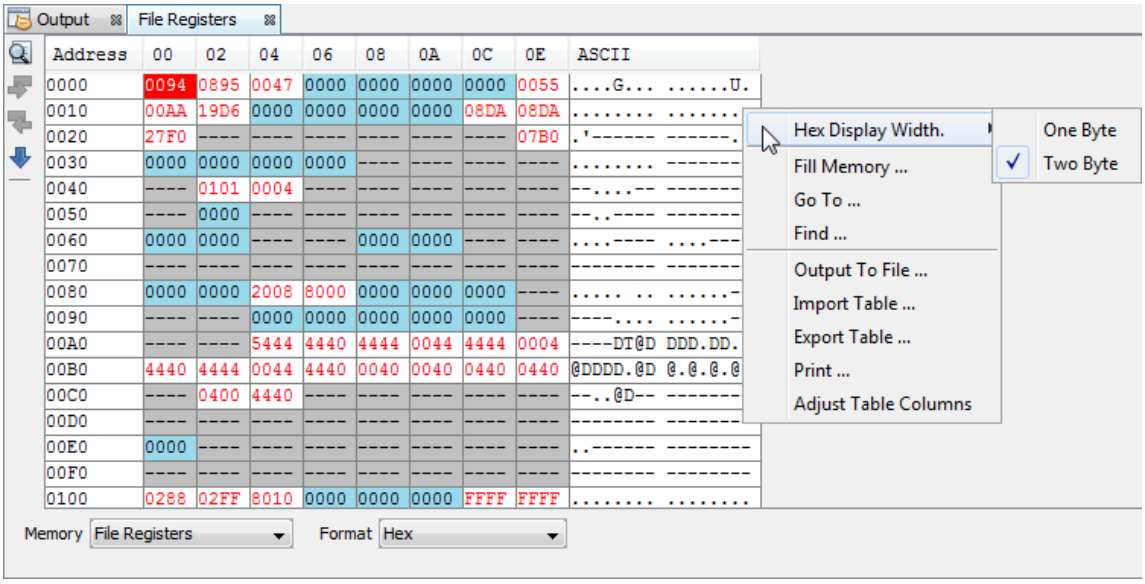
[File Registers]ウィンドウは、選択したデバイスの全ファイルレジスタを表示します。ファイルレジスタの値が変化した場合、またはプロセッサがデバッガに応答した場合、[File Registers]ウィンドウのデータが更新されます(更新された値は赤で表示される)。

一部のデータは色付きセルに表示されます。

- 青色セルはSFRを示します。
- 緑色セルはプロジェクトの変数を示します。
- 灰色セルは予約済みメモリを示します。

Note: 特定のハードウェア ツールでデバッグ速度を改善するには、このウィンドウを閉じます。代わりに[SFR]または[Watches]ウィンドウを使います。

図12-10 [File Registers]ウィンドウの内容



12.9.2.1 [File Registers]ウィンドウの表示

ウィンドウ下部の[Format]ドロップダウン ボックスで選択してウィンドウ内のメモリ表示方法を変更できます。

デバッグ用ハードウェア ツールを使う場合、一部のレジスタは予約済みリソースを表すためにデータのニブルごとに「R」を表示する事があります。

[Hex]

このフォーマットはファイルレジスタ情報をHEXデータで表示します。このウィンドウは以下の列を表示します。

- Address – 隣の列のデータの16進アドレスです。
- データブロック – 16進データです。1バイトまたは2バイトのブロックで表示されます。
- ASCII – 対応するデータ行のASCII表示です。

[Symbol]

このフォーマットは、HEX、10進、バイナリ形式の対応データで各ファイルレジスタをシンボル表示します。このウィンドウは以下の列を表示します。

- Address – データの16進アドレスです。
- Symbol – データのシンボル名です。
- 基数情報 – Hex、Decimal、Binary、Char
  基数情報はこれらの4列に表示されます。HEXは1バイトまたは2バイトのブロックで表示されます。

Dual Port (dsPIC33FJ DSC/PIC24HJ MCUのみ)

このフォーマットはファイルレジスタ情報をHEXデータで表示します。このウィンドウは以下の列を表示します。

- Address – データの16進アドレスです。
- DMA Address – シリコンのDMAアドレスからのオフセットです。
- データブロック – 16進データです。1バイトまたは2バイトのブロックで表示されます。
- ASCII – 対応するデータ行のASCII表示です。

dsPIC33F DSCおよびPIC24H MCUの詳細は、弊社ウェブサイトのデバイス データシートと『dsPIC33F/PIC24Hファミリ リファレンス マニュアル』の各セクションでご確認ください。

[XY Data] (dsPIC DSCデバイスのみ)

このフォーマットはファイルレジスタ情報をHEXデータで表示します。このウィンドウは以下の列を表示します。

- Address – データの16進Xアドレスです。
- Y Bus – データの16進Yアドレスです(サポートされている場合)。
- データブロック – 16進データです。2バイトのブロックで表示されます。
- ASCII – 対応するデータ行のASCII表示です。

dsPIC DSCデバイスの詳細は『dsPIC30Fファミリ リファレンス マニュアル』(DS70046)を参照してください。

12.9.2.2 [File Registers]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-20 [File Registers]のアイコン

アイコン	アイコンテキスト	機能
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Go To	指定した行番号またはアドレスに移動します。

12.9.2.3 [File Registers]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、このメニューが表示されます。デバイスによっては表示されない項目があります。

表12-21 [File Registers]ウィンドウのコンテクスト メニュー

項目	説明
Hex Display Width	**[Hex]フォーマットのみ** 16進数の表示幅を設定します(選択肢はデバイスによって異なります)。
Fill Memory	メモリの[Start Address]～[End Address]に[Data]内の値を書き込みます。[Fill Memory]ダイアログで、その他のオプションを指定します。
Go To	[Go To]ダイアログで指定したアドレス/関数に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Import Table	**[Hex]、[Dual Port]、[XY Data]フォーマットのみ** [Import Table]ダイアログを使って、ファイルから[Memory]ウィンドウに表形式のデータをインポートします。
Export Table	**[Hex]、[Dual Port]、[XY Data]フォーマットのみ** [Export Table]ダイアログを使って、[Memory]ウィンドウからファイルへ表形式のデータをエクスポートします。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

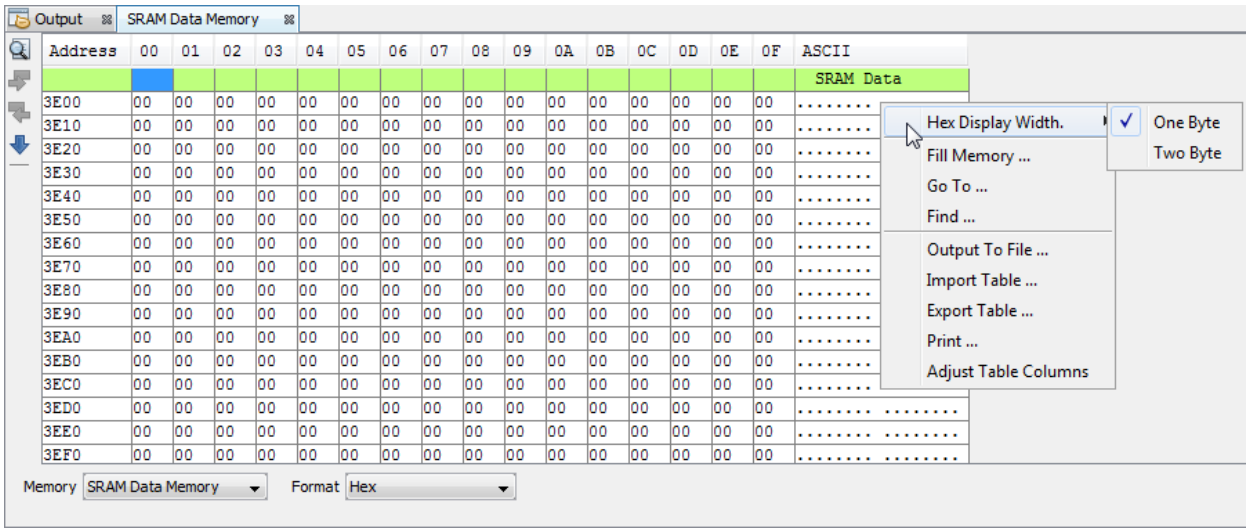
12.9.3 [SRAM Data Memory]ウィンドウ

[SRAM Data Memory]ウィンドウは、選択したATデバイスの全てのSRAMを表示します。レジスタの値が変化した場合、またはプロセッサがデバッガに応答した場合、このウィンドウのデータが更新されます(更新された値は赤で表示)。

デバッグ用ハードウェア ツールを使う場合、一部のレジスタは予約済みリソースを表すためにデータのニブルごとに「R」を表示する事があります。

Note: 特定のハードウェア ツールでデバッグ速度を改善するには、このウィンドウを閉じます。代わりに[I/O Memory (SFRs)]ウィンドウまたは[Watches]ウィンドウを使います。

図12-11 [SRAM Data Memory]ウィンドウの内容



12.9.3.1 [SRAM Data Memory]ウィンドウの表示

ウィンドウ下部の[Format]ドロップダウン ボックスを選択する事で、ウィンドウ内のメモリの表示方法を変更できます。

[Hex]

このフォーマットはファイルレジスタ情報をHEXデータで表示します。このウィンドウは以下の列を表示します。

- Address – 隣の列のデータの16進アドレスです。
- データブロック – 16進データです。1バイトまたは2バイトのブロックで表示されます。
- ASCII – 対応するデータ行のASCII表示です。

[Symbol]

このフォーマットは、HEX、10進、バイナリ形式の対応データで各ファイルレジスタをシンボル表示します。このウィンドウは以下の列を表示します。

- Address – データの16進アドレスです。
- Symbol – データのシンボル名です。
- 基数情報 – Hex、Decimal、Binary、Char

基数情報は4列に表示されます。HEXは1バイトまたは2バイトのブロックで表示されます。

12.9.3.2 [SRAM Data Memory]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-22 [SRAM Data Memory]ウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。

(続き)

アイコン	アイコンテキスト	機能
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Go To	指定した行番号またはアドレスに移動します。

12.9.3.3 [SRAM Data Memory]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、このメニューが表示されます。デバイスによっては表示されない項目があります。

表12-23 [SRAM Data Memory]ウィンドウのコンテクスト メニュー

項目	説明
Hex Display Width	**[Hex]フォーマットのみ** 16進数の表示幅を設定します(選択肢はデバイスによって異なります)。
Fill Memory	メモリの[Start Address]～[End Address]に[Data]内の値を書き込みます。[Fill Memory]ダイアログで、その他のオプションを指定します。
Go To	[Go To]ダイアログで指定したアドレス/関数に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Import Table	**[Hex]フォーマットのみ** [Import Table]ダイアログを使って、ファイルから[Memory]ウィンドウに表形式のデータをインポートします。
Export Table	**[Hex]フォーマットのみ** [Export Table]ダイアログを使って、[Memory]ウィンドウからファイルへ表形式のデータをエクスポートします。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.9.4 [SFRs]ウィンドウ

特殊機能レジスタ(SFR)ウィンドウは、選択したプロセッサのSFRの内容を表示します。このウィンドウには各SFR名と複数のフォーマットが表示されているため、通常のファイルレジスタ ウィンドウよりSFRの表示に便利です。少数のSFRを表示する場合、[Watches]ウィンドウを推奨します。ハードウェア デバッグツール使用時に速度問題対策に役立つ事があります(ウィンドウ更新レートが速いため)。

ブレークが発生すると常に特殊機能レジスタの内容が更新されます。

デバッグ用ハードウェア ツールを使う場合、一部のレジスタは予約済みリソースを表すためにデータのニブルごとに「R」を表示する事があります。

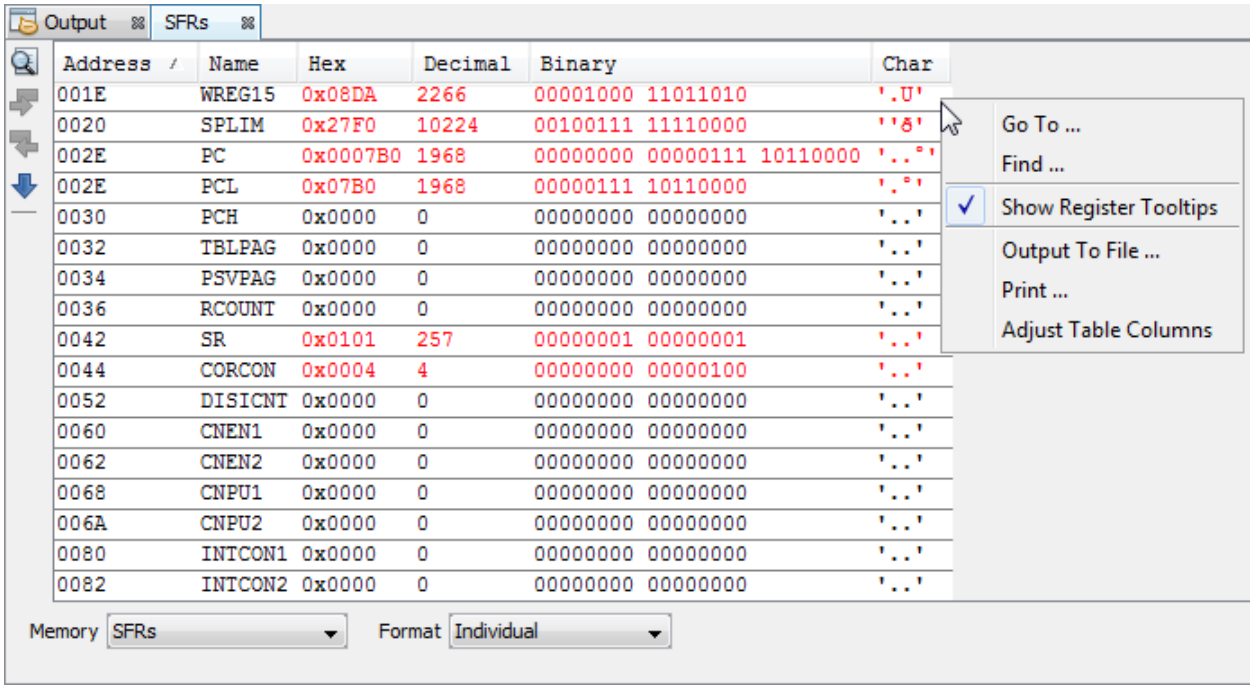
可視レジスタ

データメモリ レジスタが物理的にデバイス上に実装されていない場合、SFRの一覧に表示されない事があります。シミュレータ等の一部のツールを使うと、実際のデバイス上に存在しないプリスケーラ等のレジスタを表示できます。

シングルステップ実行

[Freeze Peripherals On Halt]を選択した場合、SFRのI/Oポートビットまたは[Watches]ウィンドウはシングルステップ実行では更新されません。ピンは変更されますが、新規の値を取得するための読み出し要求はフリーズによって阻止され、次のステップまたは実行コマンドまで更新されません。

図12-12 [SFRs]ウィンドウの内容



12.9.4.1 [SFRs]ウィンドウの表示

ウィンドウ下部の[Format]ドロップダウン ボックスを選択する事で、ウィンドウ内のメモリの表示方法を変更できます。選択肢は以下の通りです。

- Individual - 全SFRレジスタを一括して表示します。
- Peripherals - SFRレジスタを機能グループごとに表示します。

[Individual]

この表示画面では、SFRはアドレスで表示されます。データは以下の列に表示されます。

- Address – SFRの16進アドレスです。
- Name – SFRのシンボル名です。
- 基数情報 – Hex、Decimal、Binary、Char
  基数情報はこれらの4列に表示されます。HEXは1バイトのブロックで表示されます。

[Peripheral]

この表示画面では、SFRは関連するデバイスの周辺モジュールに従ってグループ化されます。データは以下の列に表示されます。

- Address – SFRの16進アドレスです。
- Name – 周辺モジュール名またはSFRのシンボル名です。
- 基数情報 – Hex、Decimal、Binary、Char
  基数情報はこれらの4列に表示されます。HEXは1バイトのブロックで表示されます。

[Filter Peripheral]ボタンをクリックして選択した周辺モジュールのみのSFRを表示します。

12.9.4.2 [SFRs]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-24 [SFRs]ウィンドウのアイコン

アイコン		機能
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Go To	指定した行番号またはアドレスに移動します。
	Filter Peripherals	**[Peripheral]フォーマットのみ** 利用可能な周辺モジュールを絞り込んで表示します。 アイコンをクリックすると[Select]ダイアログが開きます。<Shift>+クリック(連続した複数モジュール)または<Ctrl>+クリック(個別モジュール)で周辺モジュールを選択します。
	Filter Select All Peripherals.	**[Peripheral]フォーマットのみ** 全ての周辺モジュールを選択するフィルタ

12.9.4.3 [SFRs]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、このメニューが表示されます。

表12-25 [SFRs]ウィンドウのコンテクスト メニュー

項目	説明*
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.9.5 [I/O Memory (SFRs)]ウィンドウ

[I/O Memory]ウィンドウは、選択したATプロセッサのSFR(特殊機能レジスタ)の内容を表示します。このウィンドウには各SFR名と複数のフォーマットが表示されているため、通常のSRAMデータメモリ ウィンドウよりSFRの表示に便利です。少数のSFRを表示する場合、[Watches]ウィンドウを推奨します。ハードウェア デバッグツール使用時に速度問題対策に役立つ事があります(ウィンドウ更新レートが速いため)。

ブレークが発生すると常に特殊機能レジスタの内容が更新されます。

デバッグ用ハードウェア ツールを使う場合、一部のレジスタは予約済みリソースを表すためにデータのニブルごとに「R」を表示する事があります。

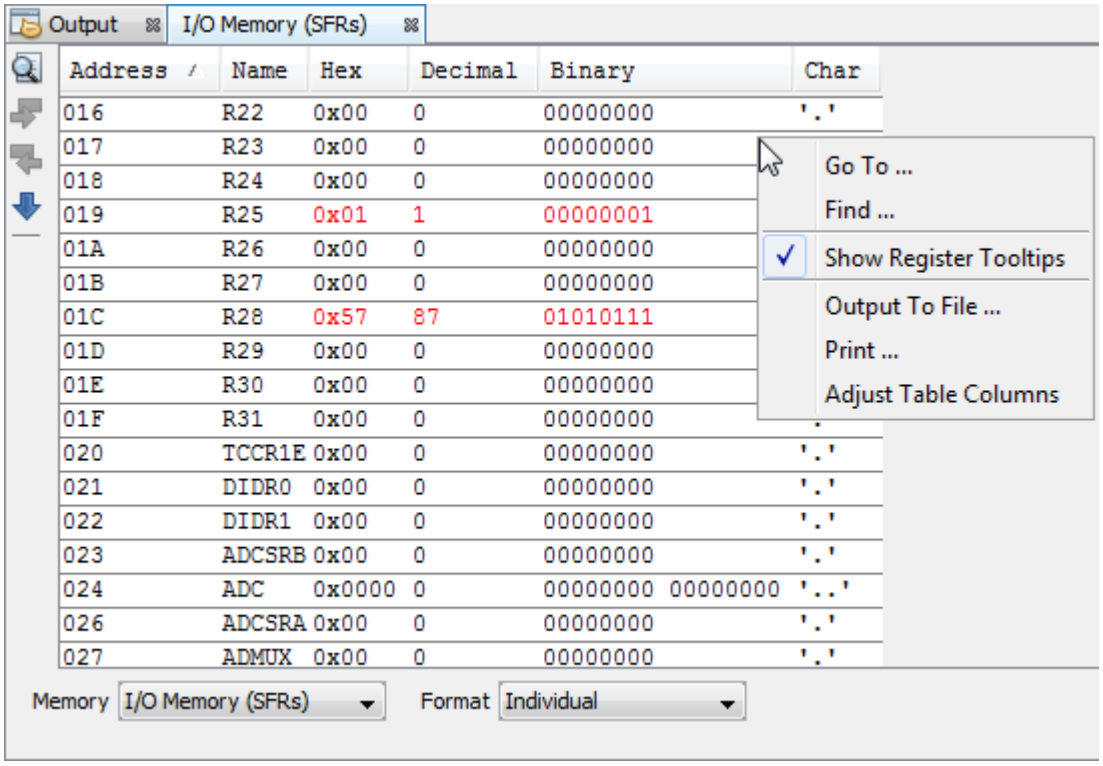
可視レジスタ

データメモリ レジスタが物理的にデバイス上に実装されていない場合、SFRの一覧に表示されない事があります。シミュレータ等の一部のツールを使うと、実際のデバイス上に存在しないレジスタを表示できます。

シングルステップ実行

[Freeze Peripherals On Halt]を選択した場合、SFRのI/Oポートビットまたは[Watches]ウィンドウはシングルステップ実行では更新されません。ピンは変更されますが、新規の値を取得するための読み出し要求はフリーズによって阻止され、次のステップまたは実行コマンドまで更新されません。

図12-13 [I/O Memory]ウィンドウの内容



12.9.5.1 [I/O Memory]ウィンドウの表示

ウィンドウ下部の[Format]ドロップダウン ボックスを選択する事で、ウィンドウ内のメモリの表示方法を変更できます。

[Individual]

この表示画面では、SFRはアドレスで表示されます。データは以下の列に表示されます。

- Address – SFRの16進アドレスです。
- Name – SFRのシンボル名です。
- 基数情報 – Hex、Decimal、Binary、Char
  基数情報はこれらの4列に表示されます。HEXは1バイトのブロックで表示されます。

[Peripheral]

この表示画面では、SFRは関連するデバイスの周辺モジュールに従ってグループ化されます。データは以下の列に表示されます。

- Address – SFRの16進アドレスです。
- Name – 周辺モジュール名またはSFRのシンボル名です。
- 基数情報 – Hex、Decimal、Binary、Char
  基数情報はこれらの4列に表示されます。HEXは1バイトのブロックで表示されます。

[Filter Peripheral]アイコンをクリックして選択した周辺モジュールのみのSFRを表示します。

[CPU Registers]

この表示画面では、CPUレジスタ(PCとワーキング レジスタRn)はアドレスで表示されます。

データは以下の列に表示されます。

- Address – SFRの16進アドレスです。
- Name – 周辺モジュール名またはSFRのシンボル名です。
- 基数情報 – Hex、Decimal、Binary、Char
  基数情報はこれらの4列に表示されます。HEXは1バイトのブロックで表示されます。

12.9.5.2 [I/O Memory]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-26 [I/O Memory]ウィンドウのアイコン

アイコン		機能
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Go To	指定した行番号またはアドレスに移動します。
	Filter Peripherals	**[Peripheral]フォーマットのみ** 利用可能な周辺モジュールを絞り込んで表示します。 アイコンをクリックすると[Select]ダイアログが開きます。<Shift>+クリック(連続した複数モジュール)または<Ctrl>+クリック(個別モジュール)で周辺モジュールを選択します。
	Filter Select All Peripherals.	**[Peripheral]フォーマットのみ** 全ての周辺モジュールを選択するフィルタ

12.9.5.3 [I/O Memory]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、このメニューが表示されます。

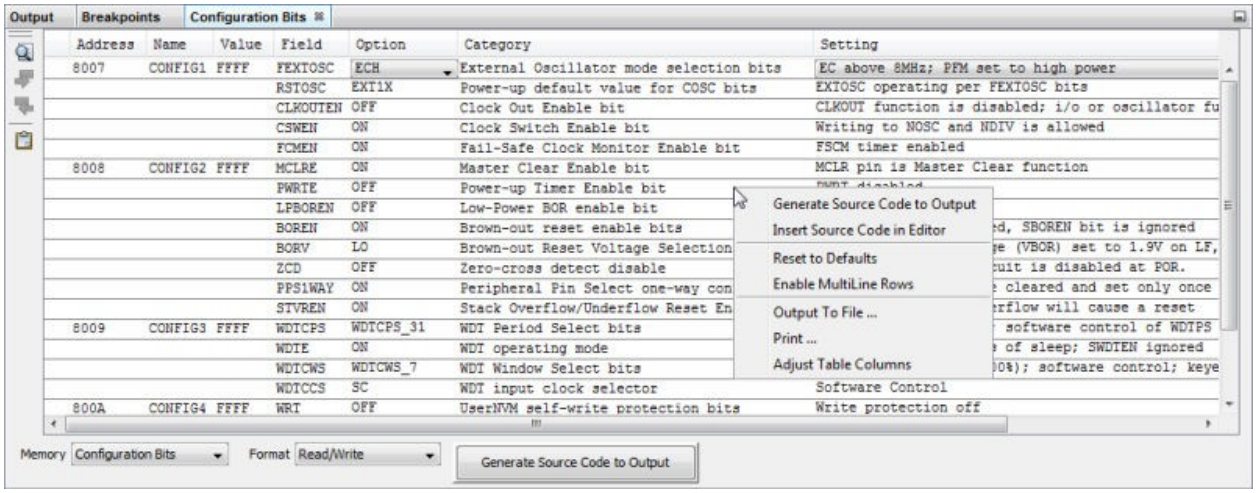
表12-27 [I/O Memory]ウィンドウのコンテクスト メニュー

項目	説明
Go To	[Go To]ダイアログで指定したアドレス/関数に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Show Register Tooltips	レジスタ用ツールチップの表示/非表示を切り換える事ができます。レジスタ名の上にマウスカーソルを置くと、情報が表示されます。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.9.6 [Configuration Bits]ウィンドウ

[Configuration Bits]ウィンドウの使い方の詳細は4.19「[Configuration Bits]ウィンドウでのコンフィグレーション値の設定」で説明しています。

図12-14 [Configuration Bits]ウィンドウの内容



12.9.6.1 [Configuration Bits]ウィンドウの表示

ウィンドウ下部の[Format]ドロップダウン ボックスを選択する事で、ウィンドウ内のメモリの表示方法を変更できます。利用可能なフォーマットはデバイスによって異なります。

ウィンドウでコンフィグレーション ビットの設定が完了したら**[Generate Source Code to Output]**ボタンをクリックし、[Output]ウィンドウにデバイス固有のコンフィグレーション コードを書き込む事ができます。その後、エディタ ウィンドウでこのコードをアプリケーションにペーストできます。

表12-28 [Configuration Bits]ウィンドウの表示

列見出し	定義
Address	コンフィグレーション ワード/バイトのアドレスです。
Name	コンフィグレーション レジスタの名前です。
Value	コンフィグレーション ワード/バイトの現在の値です。
Field*	コード内で設定されたコンフィグレーション ビットのマクロのフィールド部です。 例えば、WDTEはマクロ_WDTE_OFFのフィールド部です。
Option*	コード内で設定されたコンフィグレーション ビットのマクロのオプション部です。 例えば、OFFはマクロ_WDTE_OFFのオプション部です。
Category	対応するコンフィグレーション ワード/バイトのコンフィグレーション ビット名です。
Setting	コンフィグレーション ビットの現在の値です。設定を変更するには、ドロップダウン リストを使います。それに従ってコンフィグレーション ワード/バイトの値が変更されます。
* デバイスによってはサポートしていません。	

12.9.6.2 [Configuration Bits]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-29 [Configuration Bits]ウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Refresh by Read Device Memory	デバッグツール バーの[Read Device Memory]アイコンと同じ機能 - デバイスメモリをMPLAB X IDEにアップロードします。
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。

(続き)

アイコン	アイコンテキスト	機能
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Paste	生成されたソースコードをエディタのカーソル位置に貼り付けます。 エディタにフォーカスされていない場合、警告とソースコードが[Output]ウィンドウに表示されます。

12.9.6.3 [Configuration Bits]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、このメニューが表示されます。MCUによっては表示されない項目があります。

表12-30 [Configuration Bits]ウィンドウのメニュー

項目	説明
Generate Source Code to Output	コンフィグレーション ビットを設定します。次に、このオプションを選択して[Output]ウィンドウでコードを生成すると、このコードをプログラムにペーストしてコンフィグレーション ビットを設定できます。 ウィンドウのボタンと同じ機能です。
Insert Source Code to Editor	コンフィグレーション ビットを設定します。次に、このオプションを選択して[Editor]ウィンドウ内でコードを生成すると、このコードをプログラムにペーストしてコンフィグレーション ビットを設定できます。
Reset to Defaults	全てのコンフィグレーション ビットの値をMPLAB X IDEの既定値にリセットします。
Enable Multiline Rows	列が内容(カテゴリ等)より短い場合、内容の折り返しを許可します。選択しない場合内容は省略され、省略記号(...)で示されます。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.9.7 [EE Data Memory]ウィンドウ

[EE Data Memory]ウィンドウは、EEPROMデータメモリを備えたMCU(例: PIC16F1829)のEEPROMデータを表示します。選択したデバイスのデータ/オペコードHEX情報が表示されます。

MPLAB Xのシミュレータでは、EEPROMレジスタの値が変化した場合またはプロセッサが停止した場合、[EE Data Memory]ウィンドウのデータが更新されます。

Microchip社のハードウェア デバッグツール(例: MPLAB PICkit 4インサーキット デバッガ)では、EEPROMレジスタの値が変化した場合、またはプロセッサが停止した場合、[EE Data Memory]ウィンドウのデータは更新されません。データを更新するには、デバイスメモリのデバッグ読み出し(デバイス/ヘッダがこれをサポートしている場合)を実行する必要があります。または、デバッグ セッションを終了した後、デバイスのデータを読み出す必要があります。

EEPROMデータメモリの開始アドレスは、プログラマで指定する必要があります。下表に、一般的な開始アドレスを示します。ただし実際のアドレスはデバイスのプログラミング仕様で確認する必要があります。

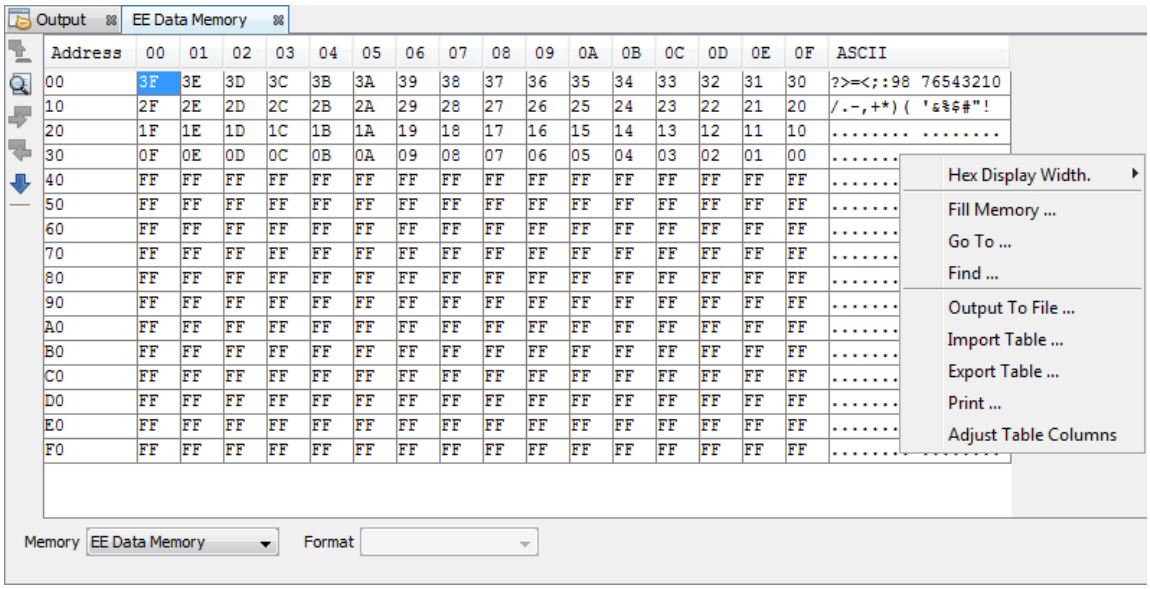
表12-31 プログラマ – データEEPROMの開始アドレス

デバイス	一般的な開始アドレス
ミッドレンジMCU	0x2100

(続き)

デバイス	一般的な開始アドレス
エンハンスト ミッドレンジMCU	0x1E000
PIC18F MCU	0xF00000
PIC24 MCU、dsPIC DSC	0x7FFE00

図12-15 [EE Data Memory]ウィンドウの内容



12.9.7.1 [EE Data Memory]ウィンドウの表示

この画面は以下の列にデータを表示します。

- Address – 隣の列のデータの16進アドレスです。
- データブロック – 16進データです。メニューから選択できる1バイト、2バイト、4バイトのブロックで表示されます。
- ASCII – 対応するデータ行のASCII表示です。

12.9.7.2 [EE Data Memory]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-32 [EE Data Memory]ウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Refresh by Read Device Memory	デバッグツール バーの[Read Device Memory]アイコンと同じ機能 - デバイスメモリをMPLAB X IDEにアップロードします。
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Go To	指定した行番号またはアドレスに移動します。

12.9.7.3 [EE Data Memory]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、下表のメニューが表示されます。

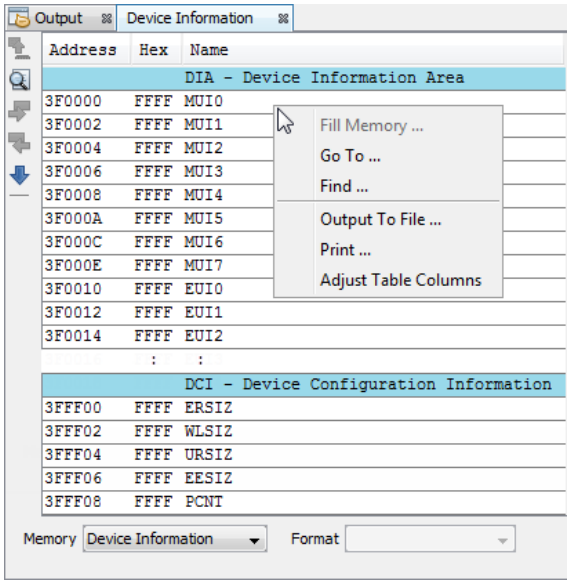
表12-33 [EE Data Memory]ウィンドウのコンテクスト メニュー

項目	説明*
Hex Display Width	**[Hex]フォーマットのみ** 16進数の表示幅を設定します。 (選択肢はデバイスによって異なります。)
Fill Memory	メモリの[Start Address]～[End Address]に[Data]内の値を書き込みます。[Fill Memory]ダイアログで、その他のオプションを指定します。
Go To	[Go To]ダイアログで指定したアドレス/関数に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Import Table	[Import Table]ダイアログを使って、ファイルから[Memory]ウィンドウに表形式のデータをインポートします。
Export Table	[Export Table]ダイアログを使って、[Memory]ウィンドウからファイルへ表形式のデータをエクスポートします。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの[General]タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.9.8 [Device Information]ウィンドウ

[Device Information]ウィンドウは、このメモリ空間をサポートするデバイス(現時点ではPIC18F24K42と多くのPIC16Fxxxxx)用メモリのDIA (Device Information Area)の内容を表示します。DIAは内部温度インジケータ モジュールの校正データを収めており、Microchip社固有の識別子ワードと、mVで計測された固定参照電圧読み値を保存します。詳細はデバイスのデータシートを参照してください。

図12-16 [Device Information]ウィンドウ



12.9.8.1 [Device Information]ウィンドウの画面

この画面は以下の列にデータを表示します。

- Address – 隣の列のデータの16進アドレスです。

- Hex – 16進データです。メニューから選択できる1バイト、2バイト、4バイトのブロックで表示されます。
- Name – データの対応行のASCII表示です。

12.9.8.2 [Device Information]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-34 [Device Information]ウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Refresh by Read Device Memory	デバッグツール バーの[Read Device Memory]アイコンと同じ機能 - デバイスメモリをMPLAB X IDEにアップロードします。
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Go To	指定した行番号またはアドレスに移動します。

12.9.8.3 [Device Information]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、このメニューが表示されます。デバイスによっては表示されない項目があります。

表12-35 [Device Information]ウィンドウのコンテクスト メニュー

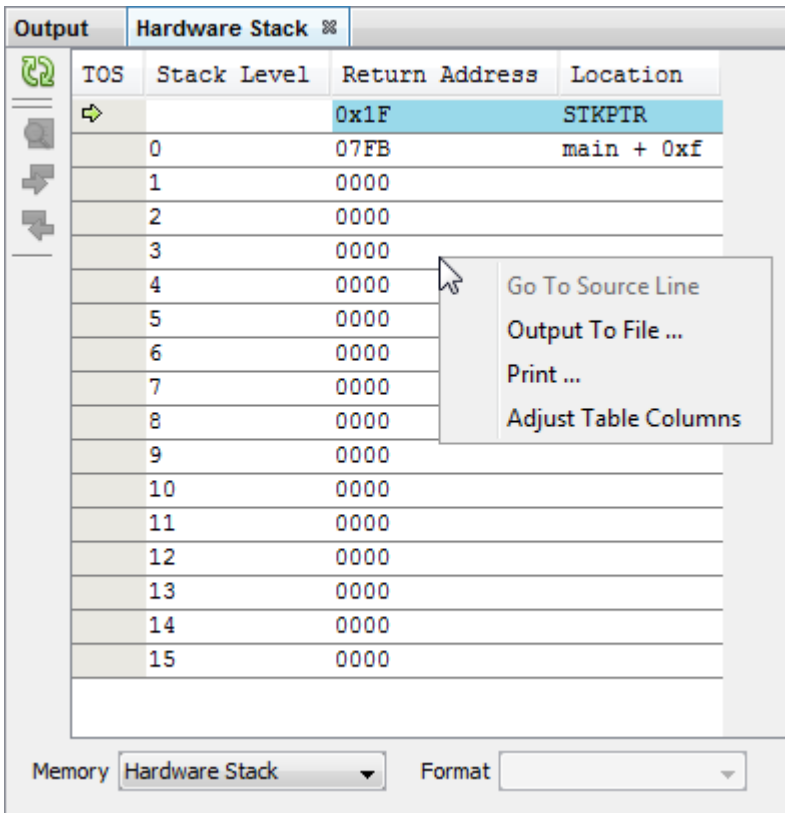
項目	説明
Fill Memory	メモリの[Start Address]～[End Address]に[Data]内の値を書き込みます。[Fill Memory]ダイアログで、その他のオプションを指定します。
Go To	[Go To]ダイアログで指定したアドレス/関数に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.9.9 [Hardware Stack]ウィンドウ

停止時にハードウェア スタックの内容を表示します。

[Hardware Stack]ウィンドウは、[Window] > [PIC Memory Views] > [Hardware Stack]と選択して開きます。最初のスタックレベルは「0」で表します。

図12-17 [Hardware Stack]ウィンドウの内容



12.9.9.1 [Hardware Stack]ウィンドウの表示

スタックのレベル数はデバイスによって異なります。この画面は以下の列にデータを表示します。

- TOS – Top of Stack現在位置
- Stack Level – スタック上のアイテム位置
- Return Address – スタックからのリターンアドレス16進
- Location – リターンアドレス コードの逆アセンブル

デバッグ用ハードウェア ツールを使う場合、一部のレジスタは予約済みリソースを表すためにデータのニブルごとに「R」を表示する事があります。

12.9.9.2 [Hardware Stack]ウィンドウのアイコン

一時停止または停止時にこのウィンドウで使えるアイコンは以下の通りです。

表12-36 [Hardware Stack]ウィンドウのアイコン

アイコン	アイコン テキスト	説明
	Auto Refresh Manual Refresh	ウィンドウの内容を自動または手動で更新します。 [Tools] > [Options] > [Embedded] > [Generic Settings]の[Disable auto refresh for call stack view during debug sessions]の値(チェックを外す = 偽(既定値)、チェックを入れる = 真)の影響を受けます。12.16.1「[Generic Settings]タブ」を参照してください。 偽 = 自動更新(緑のアイコン): ウィンドウの内容は、一時停止または停止時に自動更新されます。ウィンドウが閉じられている場合、またはフォーカスされていない場合、ウィンドウを選択しこのボタンをクリックすると更新が表示されます。再度実行および一時停止/停止する必要はありません。 真 = 手動更新(オレンジのアイコン): ウィンドウの内容は、一時停止または停止しても自動では更新されません。一時停止/停止時にウィンドウの内容を更新するには、このボタンをクリックする必要があります。

(続き)

アイコン	アイコン テキスト	説明
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。

12.9.9.3 [Hardware Stack]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、このメニューが表示されます。

表12-37 [Hardware Stack]ウィンドウのコンテクスト メニュー

項目	説明*
Go To Source Line	エディタでソースコード内の対応する行に移動します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

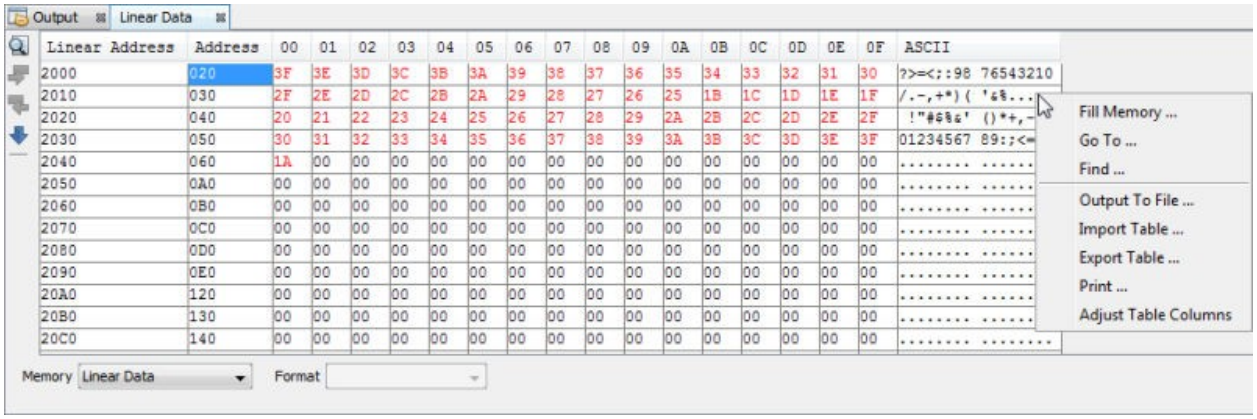
12.9.10 [Linear Data]ウィンドウ

[Linear Data]ウィンドウは、このタイプのメモリを備えたMCUのフラッシュデータを表示します。選択したデバイスのデータ/オペコードHEX情報が表示されます。

MPLAB Xのシミュレータでは、リニアデータ レジスタの値が変化した場合またはプロセッサが停止した場合、[Linear Data]ウィンドウのデータが更新されます。

Microchip社のハードウェア デバッグツール(例: MPLAB ICD 4インサーキット エミュレータ)では、リニアデータ レジスタの値が変化した場合、またはプロセッサが停止した場合、[Linear Data]ウィンドウのデータは更新されません。データを更新するには、デバイスメモリのデバッグ読み出し(デバイス/ヘッダがこれをサポートしている場合)を実行する必要があります。または、デバッグ セッションを終了した後、デバイスのデータを読み出す必要があります。

図12-18 [Linear Data]ウィンドウの内容



12.9.10.1 [Linear Data]ウィンドウの表示

この画面は以下の列にデータを表示します。

- Address – 隣の列のデータの16進アドレスです。

- データブロック – 16進データです。メニューから選択できる1バイト、2バイト、4バイトのブロックで表示されます。
- ASCII – 対応するデータ行のASCII表示です。

12.9.10.2 [Linear Data]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-38 [Linear Data]ウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Go To	指定した行番号またはアドレスに移動します。

12.9.10.3 [Linear Data]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、下表のメニューが表示されます。

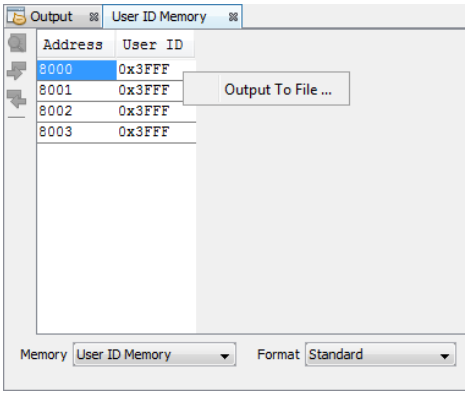
表12-39 [Linear Data]ウィンドウのコンテクスト メニュー

項目	説明*
Fill Memory	メモリの[Start Address]～[End Address]に[Data]内の値を書き込みます。[Fill Memory]ダイアログで、その他のオプションを指定します。
Go To	[Go To]ダイアログで指定したアドレス/関数に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Import Table	[Import Table]ダイアログを使って、ファイルから[Memory]ウィンドウに表形式のデータをインポートします。
Export Table	[Export Table]ダイアログを使って、[Memory]ウィンドウからファイルへ表形式のデータをエクスポートします。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.9.11 [User ID Memory]ウィンドウ

一部のデバイスは、ユーザがチェックサムまたはその他のコード識別(ID)番号を格納できるメモリ位置を持っています。これらの位置はプログラム/ベリファイ中に読み書き可能です。デバイスによっては、通常の実行中もTBLRDおよびTBLWT命令を使ってこれらの位置をアクセスできます。

図12-19 [User ID Memory]ウィンドウの内容

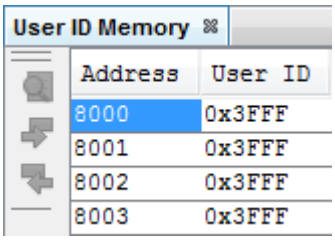


12.9.11.1 [User ID Memory]ウィンドウの表示

この欄に入力する値はデバイスのプログラミング仕様を参照してください。ほとんどのデバイスでは、この値でデバイスのIDワードの下位ニブルを設定し、上位ニブルは「0」に設定されます。上位ニブルは、テーブル書き込み等を使ってプログラム的にのみ書き込む事ができます。

ウィンドウ下部の[Format]ドロップダウン ボックスを選択する事で、ウィンドウ内のメモリの表示方法を変更できます。

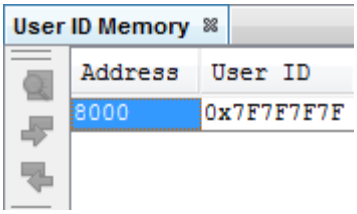
図12-20 標準表示



標準表示では、以下の列にデータを表示します。

- Address – ユーザIDの16進アドレスです。
- User ID – ユーザIDメモリの内容(HEX形式)です。

図12-21レガシー表示



レガシー表示では、以下の列にデータを表示します。

- Address – ユーザIDの16進開始アドレスです。
- User ID – ユーザIDメモリの内容(HEX形式)です。

12.9.11.2 [User ID Memory]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-40 [User ID Memory]ウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。

(続き)		
アイコン	アイコンテキスト	機能
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。

12.9.11.3 [User ID Memory]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、下表のメニューが表示されます。

表12-41 [User ID Memory]ウィンドウのコンテクスト メニュー

項目	説明*
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。

12.10 [Memory]ウィンドウ - 32ビットデバイス

[Memory]ウィンドウ([Window] > [Target Memory Views])は各種のデバイスメモリ(SFR、コンフィグレーション ビット等)を表示します。このウィンドウは、[Memory]および[Format]ドロップダウン ボックスを使ってカスタマイズできます。

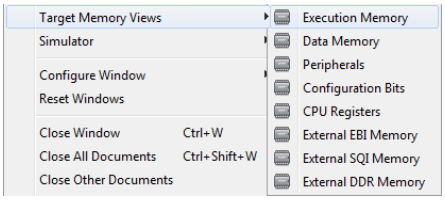
詳細は4.18「デバイスメモリの表示と変更」を参照してください。

関連ダイアログの詳細は12.11「[Memory]ウィンドウのダイアログ」を参照してください。

Note: [Configuration Bits]ウィンドウと[User ID]は、8/16/32ビットデバイスで似ています。唯一の違いは、各ウィンドウの値がデバイスで使えるプログラムメモリを反映している事です。

図12-22 [Window] > [Target Memory Views] - [PIC32MX795F512L]

図12-23 [Window] > [Target Memory Views] - [ATSAME70Q21B]



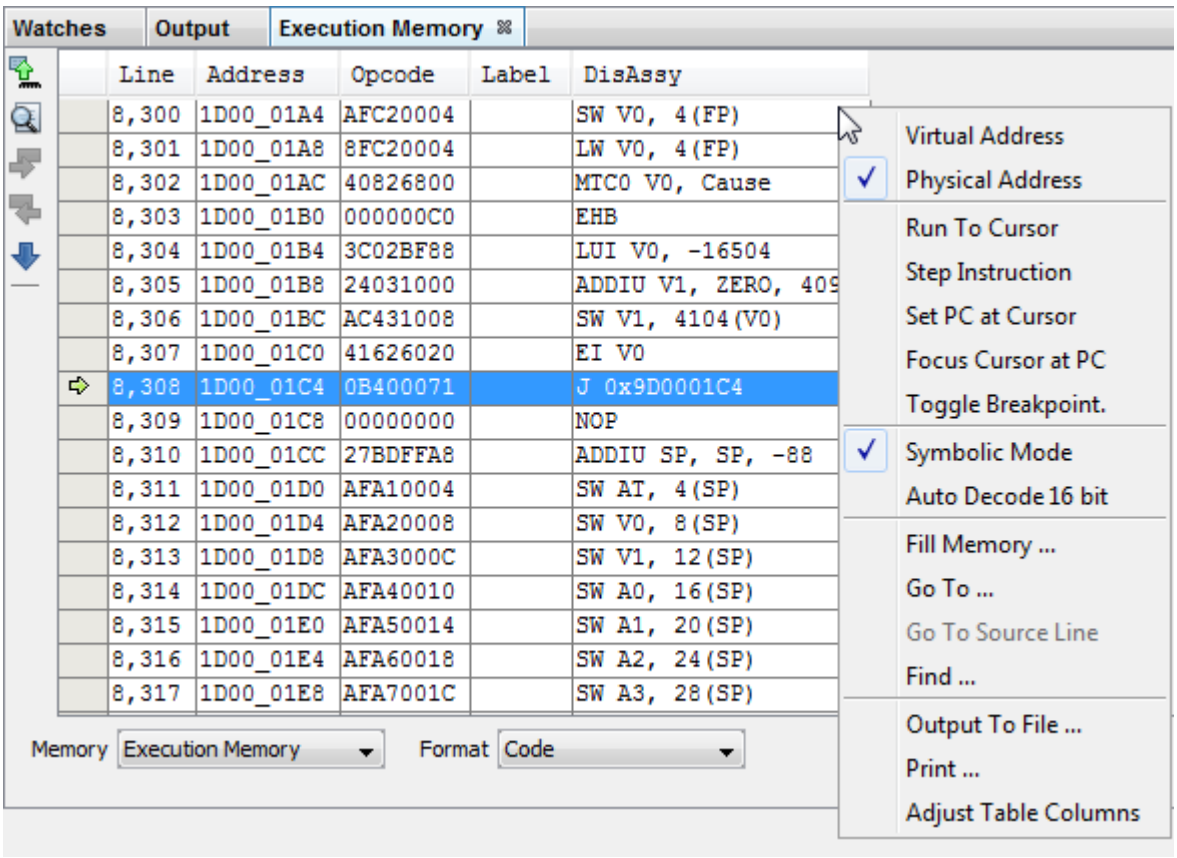
12.10.1 [Execution Memory]ウィンドウ

[Execution Memory]ウィンドウは、現在選択している32ビットデバイスのプログラムおよび/またはデータメモリのレンジ内の位置を表示します。

MPLAB Xのシミュレータでは、実行メモリの値が変化した場合またはプロセッサが停止した場合、[Execution Memory]ウィンドウのデータが更新されます。

Microchip社のハードウェア デバッグツール(例: MPLAB ICD 4インサーキット エミュレータ)では、実行メモリの値が変化した場合、またはプロセッサが停止した場合、[Execution Memory]ウィンドウのデータは更新されません。デバイスメモリの読み出しを実行する必要があります。

図12-24 [Execution Memory]ウィンドウの内容



12.10.1.1 [Execution Memory]ウィンドウの表示

ウィンドウ下部の[Format]ドロップダウン ボックスを選択する事で、ウィンドウ内のメモリの表示方法を変更できます。

デバッグ用ハードウェア ツールを使う場合、一部のレジスタは予約済みリソースを表すためにデータのニブルごとに「R」を表示する事があります。

[Code]フォーマット

このフォーマットは実行メモリ情報をHEXコードで表示します。このウィンドウは以下の列を表示します。

- Line – メモリアドレスに対応する参照行番号です。
- Address – オペコードの16進物理アドレスです。
- Opcode – 16進オペコードです。4バイトのブロックで表示されます。ハイライト表示のオペコードはプログラム カウンタの現在の位置を表します。
- Label – シンボリック フォーマットのオペコードラベルです。
- DisAssy – オペコード ニーモニックの逆アセンブル バージョンです。

[Data]フォーマット

このウィンドウは以下の列を表示します。

- Address – 隣の列のデータの16進アドレスです。
- データブロック – 16進データです。4バイトのブロックで表示されます。
- ASCII – 対応するデータ行のASCII表示です。

12.10.1.2 [Execution Memory]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-42 [Execution Memory]ウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Refresh by Read Device Memory	デバッグツール バーの[Read Device Memory]アイコンと同じ機能 - デバイスメモリをMPLAB X IDEにアップロードします。
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Go To	指定した行番号またはアドレスに移動します。

12.10.1.3 [Execution Memory]ウィンドウのメニュー

[CODE]フォーマットに設定したメモリウィンドウのデータ領域を右クリックすると、このメニューが表示されます。全ての32ビットMCUで全ての項目が表示されるわけではありません。

表12-43 [Execution Memory]ウィンドウのコンテクスト メニュー - [Code]フォーマット

項目	説明
Virtual Address (KSEG[0:1])	表示する仮想アドレスを選択します。詳細はデバイスのデータシートを参照してください。
Physical Address	表示する物理アドレスを選択します。
Run to Cursor	現在のカーソル位置までプログラムを実行します。
Step Instruction	ステップ命令を1つ実行します。
Set PC at Cursor	プログラム カウンタ(PC)を現在のカーソル位置に設定します。
Focus Cursor at PC	カーソルを現在のPCアドレス位置へ移動し、このアドレスをウィンドウの中心に表示します。
Toggle Breakpoint	既存のブレークポイントのON/OFFを切り換えます。
Symbolic Mode	逆アセンブルHEXコードをシンボルで表示します。
Auto Decode 16 bit	MIP16 / microMIPS命令デコードを使います。詳細はデバイスのデータシートを参照してください。
Fill Memory	メモリの[Start Address]～[End Address]に[Data]内の値を書き込みます。[Fill Memory]ダイアログで、その他のオプションを指定します。
Go To	[Go To]ダイアログを使って指定したアドレス/関数に移動します。
Go To Source Line	エディタでソースコード内の対応する行に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

[DATA]フォーマットに設定したメモリウィンドウのデータ領域を右クリックすると、このメニューが表示されます。全ての32ビットMCUで全ての項目が表示されるわけではありません。

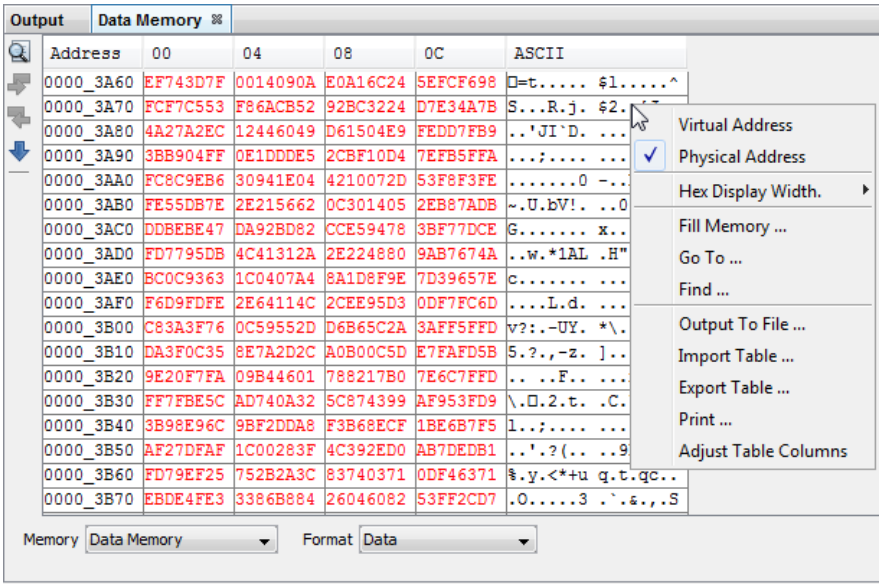
表12-44 [Execution Memory]ウィンドウのコンテクスト メニュー - [Data]フォーマット

項目	説明
Virtual Address (KSEG[0:1])	表示する仮想アドレスを選択します。詳細はデバイスのデータシートを参照してください。
Physical Address	表示する物理アドレスを選択します。
Hex Width Display	16進数の表示幅を設定します。(選択肢はデバイスによって異なります。) 例: One byte(例: 00 01 02 ...0E 0F) Two bytes (例: 00 02 04 ...0C 0E) Four bytes (例: 00 04 08 0C)
Fill Memory	メモリの[Start Address]〜[End Address]に[Data]内の値を書き込みます。[Fill Memory]ダイアログで、その他のオプションを指定します。
Go To	[Go To]ダイアログで指定したアドレス/関数に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Import Table	[Import Table]ダイアログを使って、ファイルから[Memory]ウィンドウに表形式のデータをインポートします。
Export Table	[Export Table]ダイアログを使って、[Memory]ウィンドウからファイルへ表形式のデータをエクスポートします。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.10.2 [Data Memory]ウィンドウ

[Data Memory]ウィンドウは、現在選択している32ビットデバイスのデータおよび/またはプログラムメモリのレンジ内の位置を表示します。

図12-25 [Data Memory]ウィンドウの内容



12.10.2.1 [Data Memory]ウィンドウの表示

ウィンドウ下部の[Format]ドロップダウン ボックスを選択する事で、ウィンドウ内のメモリの表示方法を変更できます。

デバッグ用ハードウェア ツールを使う場合、一部のレジスタは予約済みリソースを表すためにデータのニブルごとに「R」を表示する事があります。

[Data]フォーマット

このウィンドウは以下の列を表示します。

- Address – 隣の列のデータの16進アドレスです。
- データ ブロック – 16進データです。4バイトのブロックで表示されます。
- ASCII – 対応するデータ行のASCII表示です。

[Code]フォーマット

このウィンドウは以下の列を表示します。

- Line – メモリアドレスに対応する参照行番号です。
- Address – オペコードの16進物理アドレスです。
- Opcode – 16進オペコードです。4バイトのブロックで表示されます。
  ハイライト表示のオペコードはプログラム カウンタの現在の位置を表します。
- Label – シンボリック フォーマットのオペコードラベルです。
- DisAssy – オペコード ニーモニックの逆アセンブル バージョンです。

12.10.2.2 [Data Memory]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-45 [Data Memory]ウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Refresh by Read Device Memory	デバッグツール バーの[Read Device Memory]アイコンと同じ機能 - デバイス メモリをMPLAB X IDEにアップロードします。
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。

(続き)

アイコン	アイコンテキスト	機能
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Go To	指定した行番号またはアドレスに移動します。

12.10.2.3 [Data Memory]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、このメニューが表示されます。全ての32ビットMCUで全ての項目が表示されるわけではありません。

表12-46 [Data Memory]ウィンドウのコンテクスト メニュー

項目	説明
Virtual Address	表示する仮想アドレスを選択します。詳細はデバイスのデータシートを参照してください。
Physical Address	表示する物理アドレスを選択します。
Symbolic Mode	**[Code]フォーマットのみ** 逆アセンブルHEXコードをシンボルで表示します。
Hex Width Display	**[Data]フォーマットのみ** 16進数の表示幅を設定します。(選択肢はデバイスによって異なります。) 例: One byte (例: 00 01 02 ...0E 0F) Two bytes (例: 00 02 04 ...0C 0E) Four bytes (例: 00 04 08 0C)
Fill Memory	メモリの[Start Address]～[End Address]に[Data]内の値を書き込みます。[Fill Memory]ダイアログで、その他のオプションを指定します。
Go To	[Go To]ダイアログで指定したアドレス/関数に移動します。
Go To Source Line	**[Code]フォーマットのみ** エディタでソースコード内の対応する行に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Import Table	[Import Table]ダイアログを使って、ファイルから[Memory]ウィンドウに表形式のデータをインポートします。
Export Table	[Export Table]ダイアログを使って、[Memory]ウィンドウからファイルへ表形式のデータをエクスポートします。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.10.3 [Peripherals]ウィンドウ

[Peripherals]ウィンドウは、デバイスの周辺モジュールに関連するSFRの内容を表示します。少数のSFRを表示するには、[Watches]ウィンドウを使う事を推奨します。ハードウェア デバッグツール使用時の速度の問題の対策に役立つ事があります(ウィンドウ更新レートが速いため)。

ブレークが発生すると常にSFRの内容が更新されます。

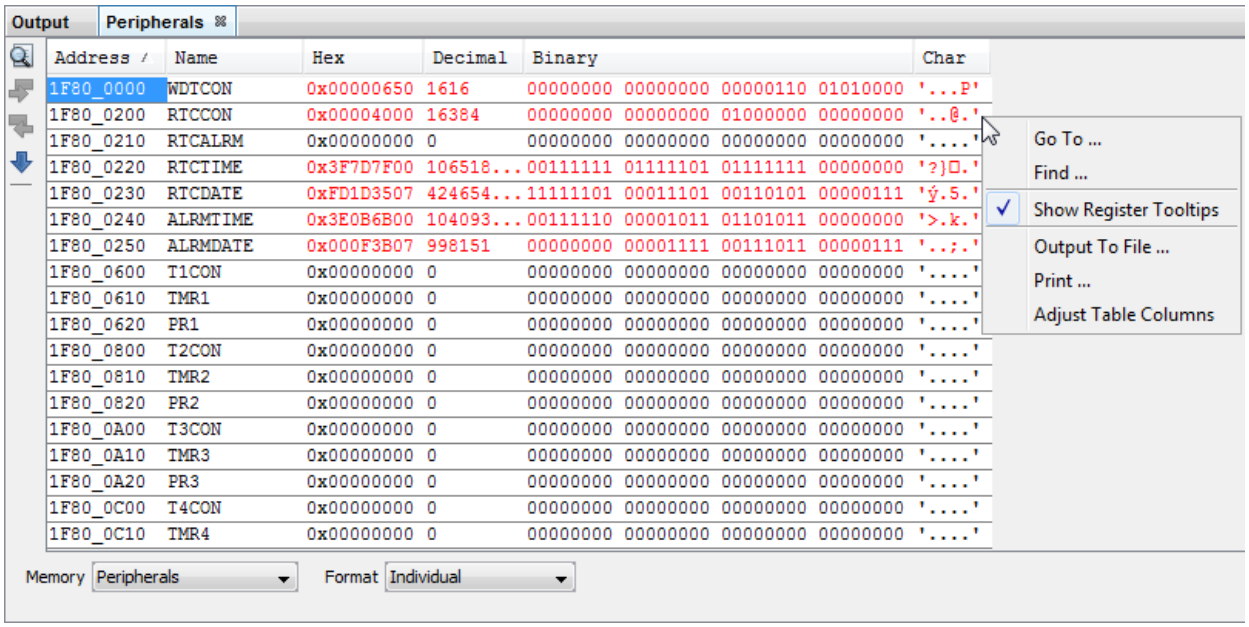
可視レジスタ

データメモリ レジスタが物理的にデバイス上に実装されていない場合、SFRの一覧に表示されない事があります。シミュレータ等の一部のツールを使うと、実際のデバイス上に存在しないプリスケーラ等のレジスタを表示できます。

シングルステップ実行

[Freeze Peripherals On Halt]を選択した場合、SFRのI/Oポートビットまたは[Watches]ウィンドウはシングルステップ実行では更新されません。ピンは変更されますが、新規の値を取得するための読み出し要求はフリーズによって阻止され、次のステップまたは実行コマンドまで更新されません。

図12-26 [Peripherals]ウィンドウの内容



12.10.3.1 [Peripherals]ウィンドウの表示

ウィンドウ下部の[Format]ドロップダウン ボックスを選択する事で、ウィンドウ内のメモリの表示方法を変更できます。

デバッグ用ハードウェア ツールを使う場合、一部のレジスタは予約済みリソースを表すためにデータのニブルごとに「R」を表示する事があります。

- **Individual** - 全周辺モジュール レジスタを一括して表示します。
- **Groups** - 周辺モジュール レジスタを機能グループごとに表示します。

以下の列には、両フォーマットのデータが表示されます。

- Address – SFRの16進物理アドレスです。
- Virtual – バスマトリクスで定義したSFRの16進仮想アドレスです。
- Name – SFRのシンボル名です。
- 基数情報 – Hex、Decimal、Binary、Char
  行の見出しバーを右クリックして画面に基数情報を追加できます。HEXは4バイトのブロックで表示されます。

12.10.3.2 [Peripherals]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-47 [Data Memory]ウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。

(続き)

アイコン	アイコンテキスト	機能
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Go To	指定した行番号またはアドレスに移動します。

12.10.3.3 [Peripherals]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、下表のメニューが表示されます。

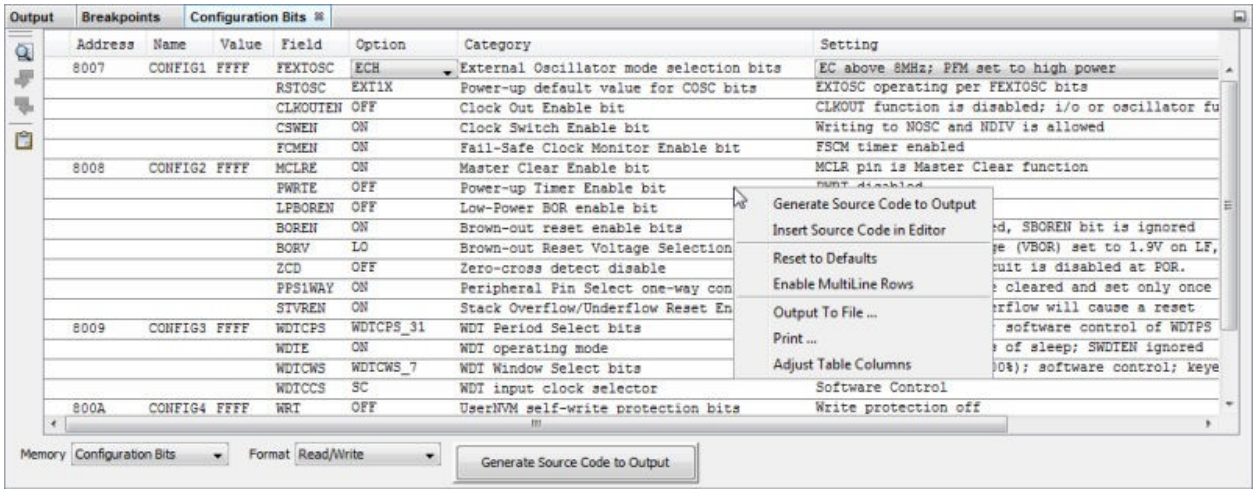
表12-48 [Peripherals]ウィンドウのコンテクスト メニュー

項目	説明
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.10.4 [Configuration Bits]ウィンドウ

[Configuration Bits]ウィンドウの使い方の詳細は4.19「[Configuration Bits]ウィンドウでのコンフィグレーション値の設定」で説明しています。

図12-27 [Configuration Bits]ウィンドウの内容



12.10.4.1 [Configuration Bits]ウィンドウの表示

ウィンドウ下部の[Format]ドロップダウン ボックスを選択する事で、ウィンドウ内のメモリの表示方法を変更できます。利用可能なフォーマットはデバイスによって異なります。

ウィンドウでコンフィグレーション ビットの設定が完了したら、**[Generate Source Code to Output]**ボタンをクリックし、[Output]ウィンドウにデバイス固有のコンフィグレーション コードを書き込む事ができます。その後、エディタ ウィンドウでこのコードをアプリケーションにペーストできます。

表12-49 [Configuration Bits]ウィンドウの表示

列見出し	定義
Address	コンフィグレーション ワード/バイトのアドレスです。
Name	コンフィグレーション レジスタの名前です。
Value	コンフィグレーション ワード/バイトの現在の値です。
Field*	コード内で設定されたコンフィグレーション ビットのマクロのフィールド部です。 例えば、WDTEはマクロ_WDTE_OFFのフィールド部です。
Option*	コード内で設定されたコンフィグレーション ビットのマクロのオプション部です。 例えば、OFFはマクロ_WDTE_OFFのオプション部です。
Category	対応するコンフィグレーション ワード/バイトのコンフィグレーション ビット名です。
Setting	コンフィグレーション ビットの現在の値です。設定を変更するには、ドロップダウン リストを使います。それに従ってコンフィグレーション ワード/バイトの値が変更されます。
* デバイスによってはサポートしていません。	

12.10.4.2 [Configuration Bits]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-50 [Configuration Bits]ウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Refresh by Read Device Memory	デバッグツール バーの[Read Device Memory]アイコンと同じ機能 - デバイスメモリをMPLAB X IDEにアップロードします。
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Paste	生成されたソースコードをエディタのカーソル位置に貼り付けます。 エディタがフォーカスされていない場合、警告とこのウィンドウに生成されたソースコードが[Output]ウィンドウに表示されます。

12.10.4.3 [Configuration Bits]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、このメニューが表示されます。MCUによっては表示されない項目があります。

表12-51 [Configuration Bits]ウィンドウのメニュー

項目	説明
Generate Source Code to Output	ウィンドウ内でコンフィグレーション ビットを必要に応じて設定します。次に、このオプションを選択して[Output]ウィンドウでコードを生成すると、このコードをプログラムにペーストしてコンフィグレーション ビットを設定できます。 ウィンドウのボタンと同じ機能です。
Insert Source Code to Editor	ウィンドウ内でコンフィグレーション ビットを必要に応じて設定します。次に、このオプションを選択して[Editor]ウィンドウ内でコードを生成すると、このコードをプログラムにペーストしてコンフィグレーション ビットを設定できます。
Reset to Defaults	全てのコンフィグレーション ビットの値をMPLAB X IDEの既定値にリセットします。
Enable Multiline Rows	列が内容(カテゴリなど等)の長さよりも短い場合、内容の折り返しを許可します。選択しない場合内容は省略され、省略記号(...)で示されます。

(続き)

項目	説明
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.10.5 [CPU Registers]ウィンドウ

[CPU Registers]ウィンドウは、デバイスのCPUに関連するSFRの内容を表示します。少数のCPU SFRを表示するには、[Watches]ウィンドウを使う事を推奨します。ハードウェア デバッグツール使用時に速度問題対策に役立つ事があります(ウィンドウ更新レートが速いため)。

ブレークが発生すると常にCPUレジスタの内容が更新されます。

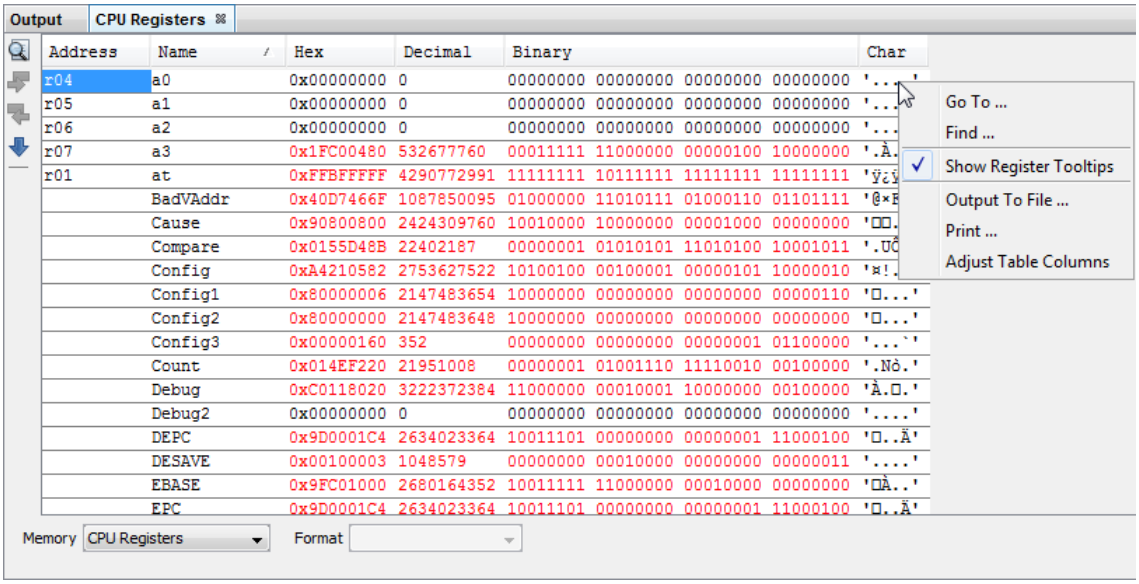
可視レジスタ

データメモリ レジスタが物理的にデバイス上に実装されていない場合、SFRの一覧に表示されない事があります。シミュレータ等の一部のツールを使うと、実際のデバイス上に存在しないプリスケーラ等のレジスタを表示できます。

シングルステップ実行

[Freeze Peripherals On Halt]を選択した場合、SFRのI/Oポートビットまたは[Watches]ウィンドウはシングルステップ実行では更新されません。ピンは変更されますが、新規の値を取得するための読み出し要求はフリーズによって阻止され、次のステップまたは実行コマンドまで更新されません。

図12-28 [CPU Registers]ウィンドウの内容



12.10.5.1 [CPU Registers]ウィンドウの表示

ウィンドウ下部の[Format]ドロップダウン ボックスを選択する事で、ウィンドウ内のメモリの表示方法を変更できます。選択肢は以下の通りです。

- Blank - 1つの画面のみ
- Devices with FPU - FPU(浮動小数点ユニット)付きPIC32デバイスの場合、[Format]の選択肢は以下の通りです。
  - All Registers
  - Only CPU Registers

– Only FPU Registers

[Blank]、[All Registers]、[Only CPU Registers]フォーマット

データは以下の列に表示されます。

- Address – SFRの16進物理アドレスです。
- Name – SFRのシンボル名です。
- 基数情報 – Hex、Decimal、Binary、Char
- Virtual – バスマトリクスで定義したSFRの16進仮想アドレスです。

[Only FPU Registers]フォーマット

データは以下の列に表示されます。

- Address – SFRの16進物理アドレスです。
- Name – SFRのシンボル名です。
- 基数情報 – HEX
- Floating Point – Float, Double

12.10.5.2 [CPU Registers]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-52 [Data Memory]ウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Go To	指定した行番号またはアドレスに移動します。

12.10.5.3 [CPU Registers]ウィンドウのメニュー

このメモリウィンドウのデータ領域で右クリックすると、下表のメニューが表示されます。

表12-53 [CPU Registers]ウィンドウのコンテクスト メニュー

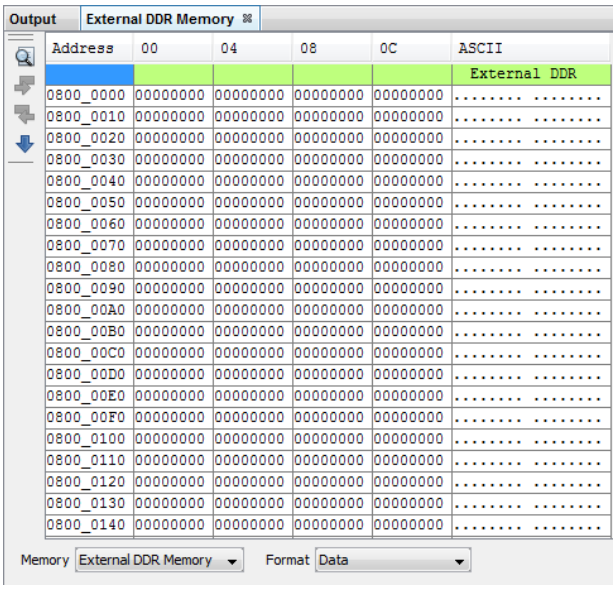
項目	説明
Go To	[Go To]ダイアログで指定したアドレス/関数に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Show Register Tooltip	レジスタ用ツールチップの表示/非表示を切り換える事ができます。レジスタ名の上にマウスカーソルを置くと、ポップアップ情報が表示されます。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.10.6 [External EBI Memory]/[External SQI Memory]/[External DDR Memory]ウィンドウ

外部メモリウィンドウは、以下のバスタイプごとに外部メモリの内容を表示します。

- EBI (外部バスインターフェイス)
- SQI (Serial Quad Interface)
- DDR(ダブル データレート)

図12-29 [External SSR Memory]ウィンドウ



12.10.6.1 External EBI、SQI、DDR Memoryウィンドウの表示

このウィンドウでは以下の手順で外部メモリの内容を表示します。

- コード内で関数/変数を外部メモリに割り当てる - 詳細は『MPLAB XC32/XC32++ Cコンパイラ ユーザガイド』(DS50001686)を参照してください。
- シンボルをロードする - [File] > [Properties]、[Loading]カテゴリで[Load symbols when programming or building for production (slows process)]にチェックを入れます。
- コードをコンパイルする - プロジェクトをビルドします。

ウィンドウ下部の[Format]ドロップダウン ボックスを選択する事で、ウィンドウ内のメモリの表示方法を変更できます。

[Code]フォーマット

このウィンドウは以下の列を表示します。

- Line – メモリアドレスに対応する参照行番号です。
- Address – オペコードの16進物理アドレスです。
- Opcode – 16進オペコードです。4バイトのブロックで表示されます。
  ハイライト表示のオペコードはプログラム カウンタの現在の位置を表します。
- Label – シンボリック フォーマットのオペコードラベルです。
- DisAssy – オペコード ニーモニックの逆アセンブル バージョンです。

[Data]フォーマット

このウィンドウは以下の列を表示します。

- Address – 隣の列のデータの16進アドレスです。
- データブロック – 16進データです。4バイトのブロックで表示されます。
- ASCII – 対応するデータ行のASCII表示です。

12.10.6.2 [External EBI Memory]/[External SQI Memory]/[External DDR Memory]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-54 外部メモリウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Refresh by Read Device Memory	デバッグツール バーの[Read Device Memory]アイコンと同じ機能 - デバイスメモリをMPLAB X IDEにアップロードします。
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。
	Go To	指定した行番号またはアドレスに移動します。

12.10.6.3 External EBI、SQI、DDR Memoryウィンドウのメニュー

[CODE]フォーマットに設定したメモリウィンドウのデータ領域を右クリックすると、このメニューが表示されます。全ての32ビットMCUで全ての項目が表示されるわけではありません。

表12-55 [Execution Memory]ウィンドウのコンテクスト メニュー - [Code]フォーマット

項目	説明
External Address (KSEG[2:3])	表示するアドレスを選択します。詳細はデバイスのデータシートを参照してください。
Physical Address	表示する物理アドレスを選択します。
Run to Cursor	現在のカーソル位置までプログラムを実行します。
Step Instruction	ステップ命令を1つ実行します。
Set PC at Cursor	プログラム カウンタ(PC)を現在のカーソル位置に設定します。
Focus Cursor at PC	カーソルを現在のPCアドレス位置へ移動し、このアドレスをウィンドウの中心に表示します。
Toggle Breakpoint	既存のブレークポイントのON/OFFを切り換えます。
Symbolic Mode	逆アセンブルHEXコードをシンボルで表示します。
Auto Decode 16 bit	MIP16 / microMIPS命令デコードを使います。詳細はデバイスのデータシートを参照してください。
Fill Memory	メモリの[Start Address]～[End Address]に[Data]内の値を書き込みます。[Fill Memory]ダイアログで、その他のオプションを指定します。
Go To	[Go To]ダイアログで指定したアドレス/関数に移動します。
Go To Source Line	エディタでソースコード内の対応する行に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

[DATA]フォーマットに設定したメモリウィンドウのデータ領域を右クリックすると、このメニューが表示されます。全ての32ビットMCUで全ての項目が表示されるわけではありません。

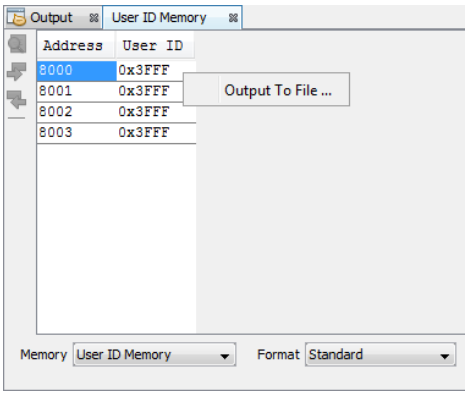
表12-56 [Execution Memory]ウィンドウのコンテクスト メニュー - [Data]フォーマット

項目	説明
External Address (KSEG[2:3])	表示するアドレスを選択します。詳細はデバイスのデータシートを参照してください。
Physical Address	表示する物理アドレスを選択します。
Hex Width Display	16進数の表示幅を設定します(選択肢はデバイスによって異なります)。 例: One byte (例: 00 01 02 ...0E 0F) Two bytes (例: 00 02 04 ...0C 0E) Four bytes (例: 00 04 08 0C)
Fill Memory	メモリの[Start Address]～[End Address]に[Data]内の値を書き込みます。[Fill Memory]ダイアログで、その他のオプションを指定します。
Go To	[Go To]ダイアログで指定したアドレス/関数に移動します。
Find	[Find]ダイアログで指定したテキストを検索します。
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。
Import Table	[Import Table]ダイアログを使って、ファイルから[Memory]ウィンドウに表形式のデータをインポートします。
Export Table	[Export Table]ダイアログを使って、[Memory]ウィンドウからファイルへ表形式のデータをエクスポートします。
Print	[Print]ダイアログを使って、このウィンドウの内容を印刷します。 **Note:** メモリ容量の大きなデバイスの場合、印刷ページ数が非常に多くなる事があります。その場合、ウィンドウの内容をファイルに出力した後([Print]ダイアログの**[General]**タブの[Print to File]チェックボックス)、そのファイルから必要なページだけ選んで印刷する事を推奨します。
Adjust Table Columns	行を自動的に調節します。

12.10.7 [User ID Memory]ウィンドウ

一部のデバイスは、ユーザがチェックサムまたはその他のコード識別(ID)番号を格納できるメモリ位置を持っています。これらの位置はプログラム/ベリファイ中に読み書き可能です。デバイスによっては、通常の実行中もTBLRDおよびTBLWT命令を使ってこれらの位置をアクセスできます。

図12-30 [User ID Memory]ウィンドウの内容

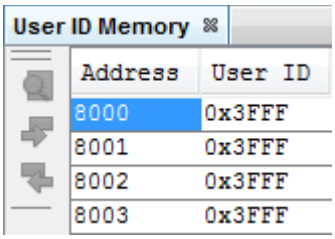


12.10.7.1 [User ID Memory]ウィンドウの表示

この欄に入力する値はデバイスのプログラミング仕様を参照してください。ほとんどのデバイスでは、この値でデバイスのIDワードの下位ニブルを設定し、上位ニブルは「0」に設定されます。上位ニブルは、テーブル書き込み等を使ってプログラム的にのみ書き込む事ができます。

ウィンドウ下部の[Format]ドロップダウン ボックスを選択する事で、ウィンドウ内のメモリの表示方法を変更できます。

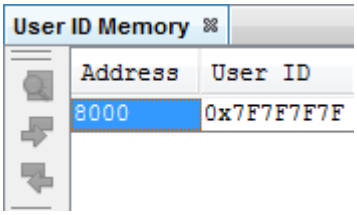
図12-31 標準表示



標準表示では、以下の列にデータを表示します。

- Address – ユーザIDの16進アドレスです。
- User ID – ユーザIDメモリの内容(HEX形式)です。

図12-32 レガシー表示



レガシー表示では、以下の列にデータを表示します。

- Address – ユーザIDの16進開始アドレスです。
- User ID – ユーザIDメモリの内容(HEX形式)です。

12.10.7.2 [User ID Memory]ウィンドウのアイコン

アイコンはウィンドウの左側にあります。

表12-57 [User ID Memory]ウィンドウのアイコン

アイコン	アイコンテキスト	機能
	Find	ウィンドウ内で検索する文字列を指定します。単語単位で一致するか、大文字と小文字を区別するかを選択します。
	Find Next	[Find]に指定した文字列の次のインスタンスを検索します。
	Find Previous	[Find]に指定した文字列の前のインスタンスを検索します。

12.10.7.3 [User ID Memory]ウィンドウのメニュー

このメニューが表示されるように設定したメモリウィンドウのデータ領域を右クリックすると、このメニューが表示されます。

表12-58 [User ID Memory]ウィンドウのコンテクスト メニュー

項目	説明*
Output To File	[Output to File]ダイアログを使って、ウィンドウの表示内容をテキストファイルに書き出します。

12.11 [Memory]ウィンドウの関連ダイアログ

[Memory]ウィンドウにはウィンドウデータのフォーマット設定、管理、出力をサポートするダイアログがあります。これらのダイアログはコンテクスト メニューからアクセスします。

12.11.1 [Memory]ウィンドウ - [Go To]ダイアログ

ターゲット メモリ内の特定位置に移動します。このダイアログへは一部のターゲットメモリ ウィンドウのコンテクスト メニューからアクセスします。

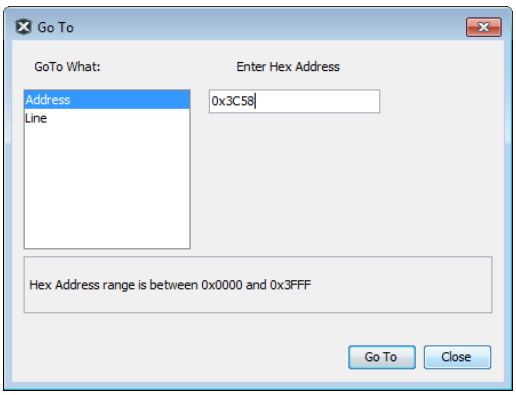


表12-59 [Go To]ダイアログ

項目	説明
Go To What	移動する項目を入力します。その項目をクリックして以下の説明を参照します。 **Address**: 16進アドレスを入力します。 **Symbol**: プログラム シンボルを入力します。 **Line**: 行番号を入力します。 **Label**: プログラムラベルを選択します。 **Function**: プログラム関数を選択します。
Go to description	[Go To What]項目を選択して説明を参照します。

12.11.2 [Memory]ウィンドウ - [Find]ダイアログ

ターゲット メモリ内の特定位置を検索します。このダイアログへは一部のターゲットメモリ ウィンドウのコンテクスト メニューからアクセスします。

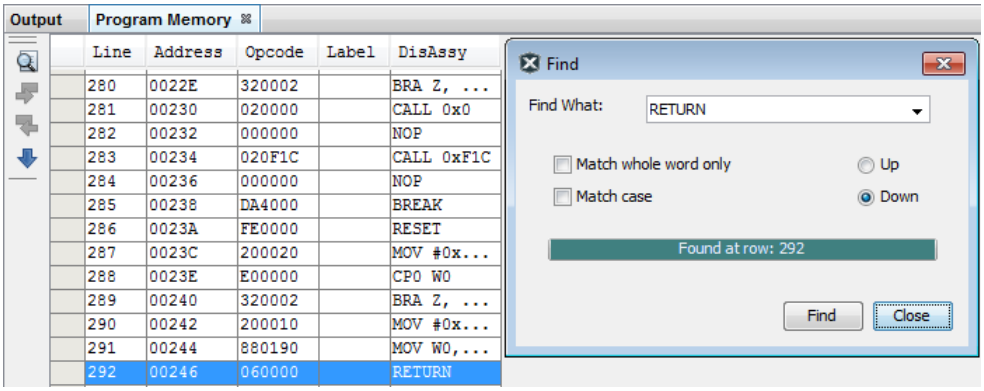


表12-60 [Find]ダイアログ

項目	説明
Find What	検索するテキストを入力します。
Match	[Match whole word only]と[Match case]の両方またはどちらか一方を選択します。
Direction	検索方向([Up]または[Down])を選択します。
Found Atバー	項目が見つかったら、その位置がこのバーに表示されます。

12.11.3 [Memory]ウィンドウ - [Output to File]ダイアログ

指定したメモリレンジをファイルに出力します。このダイアログへは一部のターゲットメモリ ウィンドウのコンテクスト メニューからアクセスします。

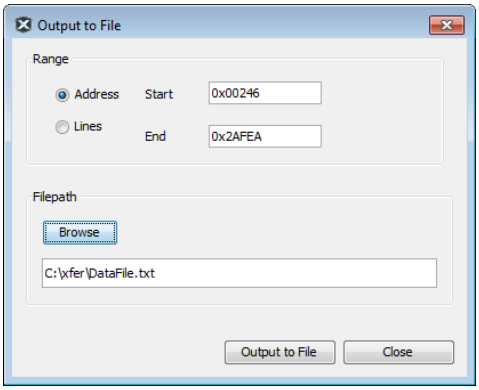


表12-61 [Output to File]ダイアログ

項目	説明
Range	ファイルに出力するデータのレンジを指定します。 **[Address]**: アドレスレンジ - [Start]/[End]アドレスを入力します。 **Lines**: 行レンジ - [Start]/[End]行を入力します。
Filepath	出力ファイルの場所を参照するか、テキストボックスに入力します。

12.11.4 [Memory]ウィンドウ - Import/Export Tableダイアログ

表形式でファイル ターゲットメモリからインポート/エクスポートします。このダイアログへは一部のターゲットメモリ ウィンドウのコンテクスト メニューからアクセスします。

図12-33 [Export Table]ダイアログ

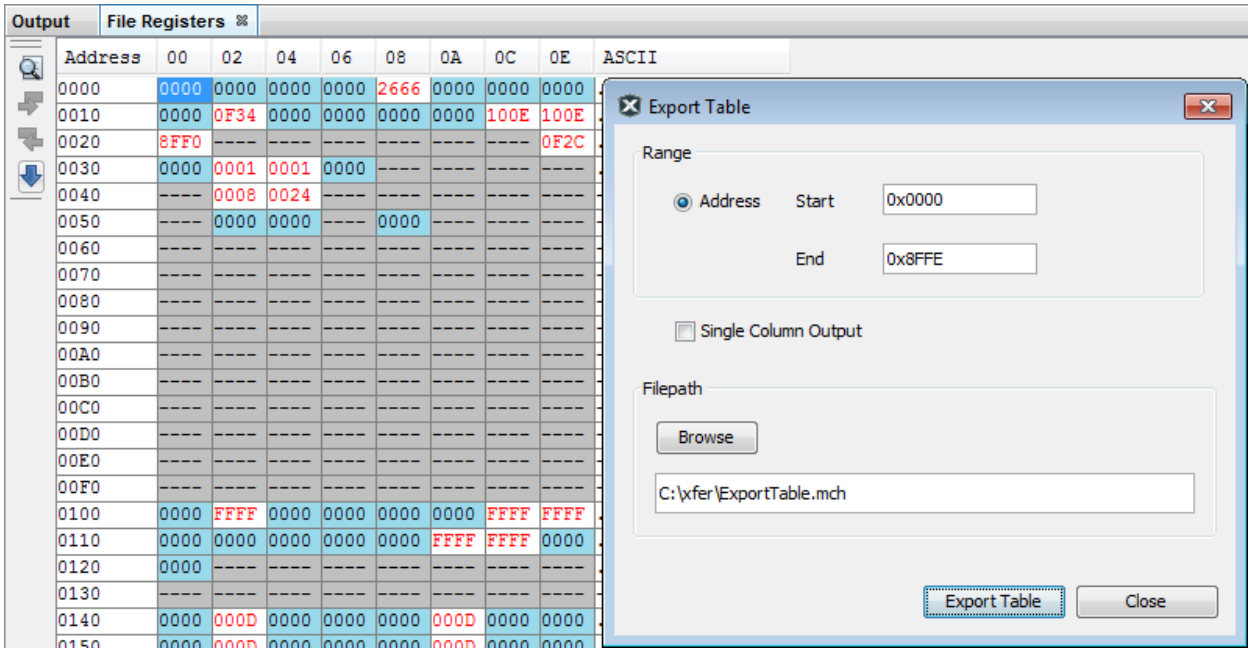


図12-34 Export Tableファイル

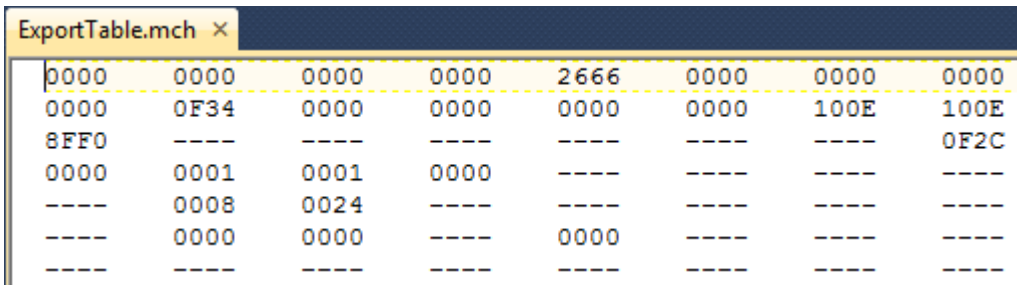


表12-62 Import/Export Tableダイアログ

項目	説明
Range	表形式でファイルにインポートまたはエクスポートするデータのレンジを入力します。 **Address**: アドレスレンジ - [Start]/[End]アドレスを入力します。
Single Column Format	表を単一の列として**エクスポート**します。
Filepath	出力または入力ファイルの場所を参照するか、テキストボックスに入力します。

12.11.5 [Memory]ウィンドウ - [Print]ダイアログ

メモリウィンドウの内容を印刷します。このダイアログへは一部のターゲットメモリ ウィンドウのコンテクスト メニューからアクセスします。使える印刷オプションは、選択したプリンタによって異なります。

図12-35 [Print]ダイアログ

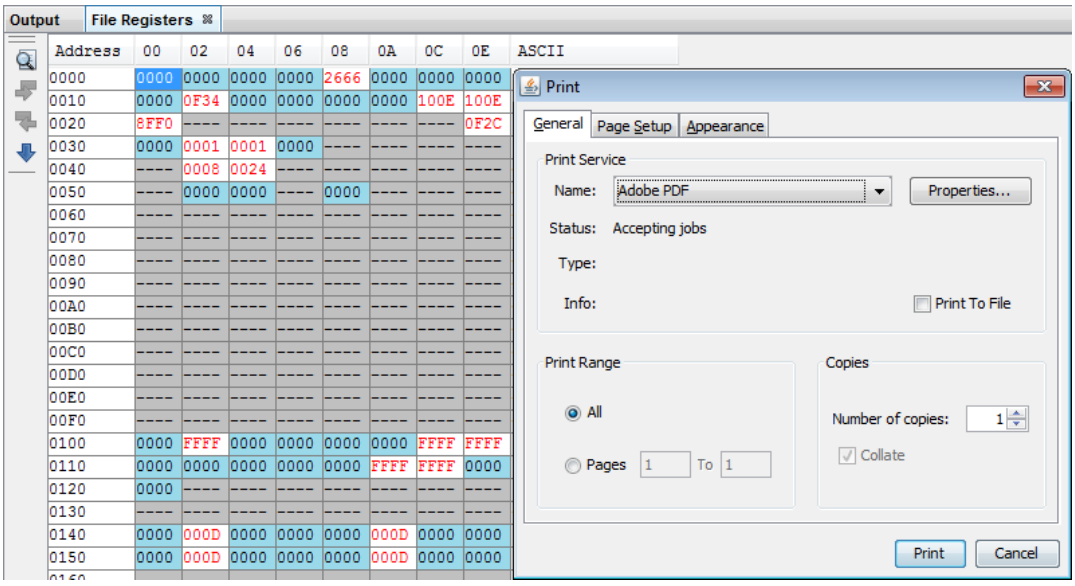


図12-36 [Print]ダイアログの出力

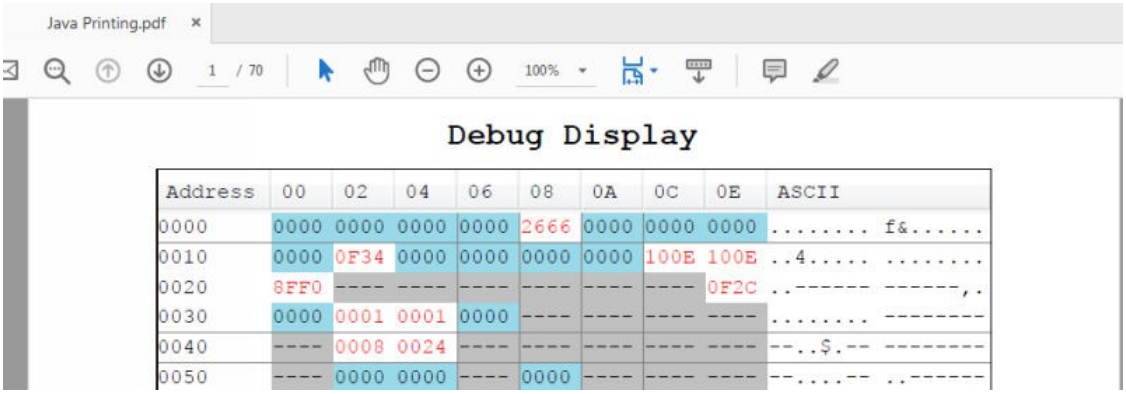


表12-63 [Print]ダイアログ

項目	説明
[General]タブ	一般的な印刷オプションを設定します。 **Print Service**: プリンタ/印刷サービスを選択します。[Properties]をクリックして印刷サービスを設定します。また、チェックを入れてファイルに印刷します。 **Print Range**: 印刷するページの範囲です。 **Copies**: 印刷する部数です。
Page Setup	ページのオプションを設定します。 **Media**: ページのサイズとソースを指定します。 **Orientation**: ページの向きを指定します。 **Margin**: ページの余白を指定します。
Appearance	印刷レイアウトを設定します。 **Color appearance**: カラーまたは白黒を指定します。 **Quality**: ドラフト、通常、高品質を指定します。 **Sides**: 側面と複数側面の向きを指定します。 **Job Attributes**: バナーを印刷し、ジョブ名とユーザを指定します。

12.12 [Message Center]ウィンドウ

[Message Center]ウィンドウは、[Window] > [Debugging] > [Output] > [Message Center]と選択して開きます。

[Message Center]ウィンドウは、全ての[Output]ウィンドウのメッセージを収集し、時系列で1つの画面に表示します。各種のフィルタを選択して関連するメッセージを表示できます。

図12-37 [Message Center]ウィンドウ

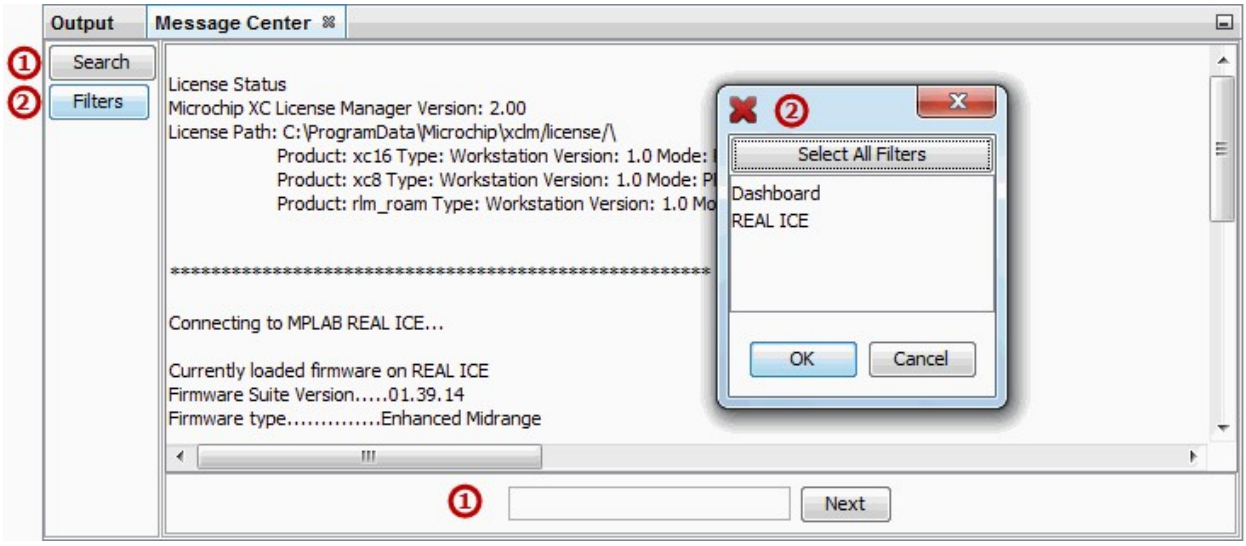


表12-64 [Message Center]ボタン

番号	ボタン	機能
1	Search	検索テキストボックスをトグルします。検索ボックスが表示されている場合、テキストを入力するとその文字列を[Message Center]ウィンドウ内で検索します。
2	Filters	既定値では、[Output]ウィンドウに表示される全てのメッセージが[Message Center]ウィンドウに表示されます。 このボタンをクリックすると、[Output]ウィンドウのタブ項目のリストを含むウィンドウが開きます。リストから項目を選択して、表示メッセージにフィルタをかけます。

12.13 [Output]ウィンドウ

このウィンドウは[Window] > [Output]と選択して開きます。

[Task]ペインはNetBeansプラットフォームから継承したウィンドウ、MPLAB X IDE専用ウィンドウ等、多くのウィンドウを表示します。[Output]ウィンドウは、MPLAB X IDE出力情報を表示します。情報はウィンドウ内のタブ上に表示されます。

表12-65 [Output]ウィンドウのタブ項目

項目	説明
Debugger Console	[User program running]等の主要なデバッグ動作を表示します。
Tool-specific	ツールのファームウェア バージョン、デバイスID、動作ステータスを表示します。
Build, Load	ビルドとプログラムロードに関する情報とステータスを表示します。
Clean, Build, Load	クリーン、ビルド、プログラミングに関する情報とステータスを表示します。
Peripheral Output	シミュレータを使ったデバッグにおいて、UART等の周辺モジュールの出力を表示します。

12.13.1 内容のスクロール

実行中のプログラミングは出力ウィンドウに表示され、スクロールして行きます。最後に実行された操作はこのウィンドウで一番下に表示されます。

内容を表示するためにスクロールアップしてプログラミングを再度実行しても、カーソルは現在の位置から動きません。これは、下のウィンドウに多くの内容が表示されて位置が分からなくなる事を防ぐためです。新しい内容を表示するには、自分でスクロールする必要があります。

12.13.2 コンテクスト メニュー

[Output]ウィンドウで右クリックすると、下表の項目が表示されます。

表12-66 [Output]ウィンドウのコンテクスト メニュー

項目	説明
Copy	選択しているテキストをクリップボードにコピーします。
Paste	現在選択しているテキストをクリップボードから[Output]ウィンドウに貼り付けます。
Find	現在選択しているテキスト、または入力したテキストを[Output]ウィンドウ内で検索します。正規表現、大文字/小文字の区別にも対応しています。
Find Next	テキストを順方向に検索します。
Find Previous	テキストを逆方向に検索します。
Filter	テキストまたは正規表現によって出力をフィルタ処理します。
Wrap Text	[Output]ウィンドウの行送りを有効にします。
Larger Font	フォントサイズを大きくします。
Smaller Font	フォントサイズを小さくします。
Choose Font	フォントタイプ、スタイル、サイズを選択します。
Save As	選択したテキストをファイルに保存します。
Clear	[Output]ウィンドウ内のテキストを全てクリアします。
Close	[Output]ウィンドウを閉じます。

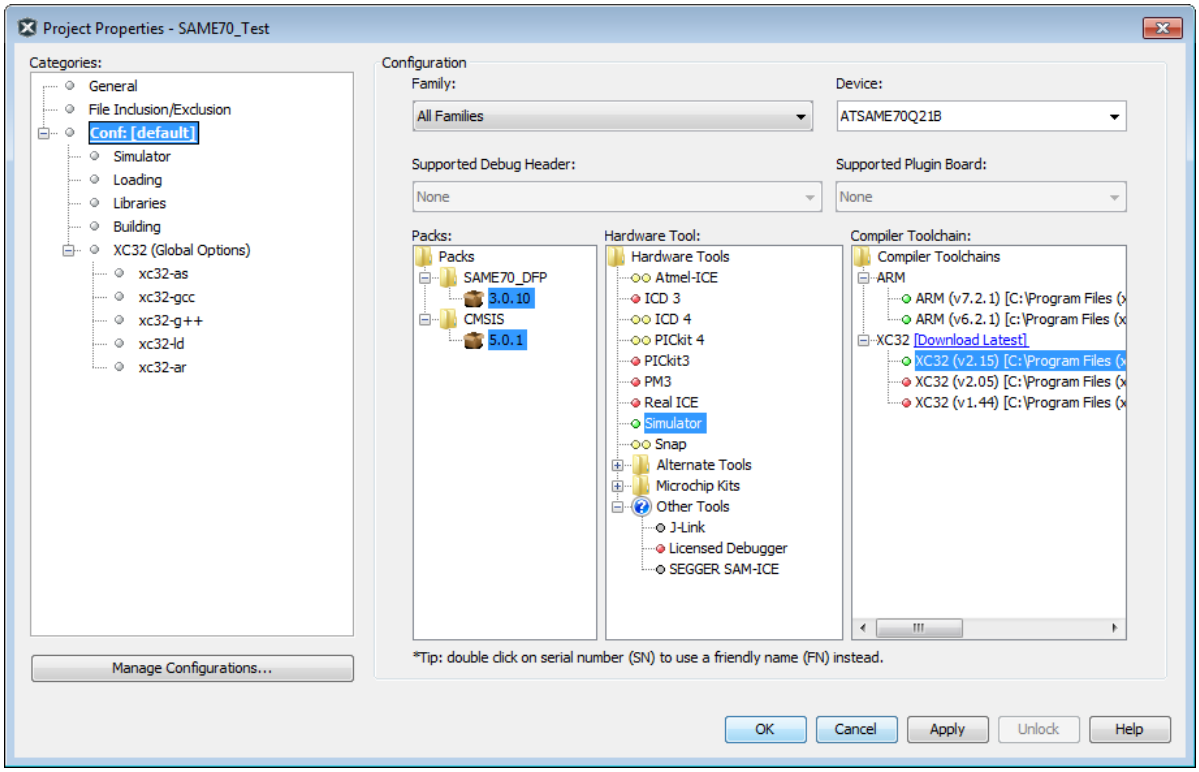
12.14 [Project Properties]ウィンドウ

このウィンドウはいくつかの方法で開く事ができます。4.3「[Project Properties]ウィンドウを開く」を参照してください。

このウィンドウは、プロジェクトのデバイス、ツール、ツール設定を表示または変更するために使います。詳細は以下を参照してください。

- 4.4 「プロジェクト プロパティの表示と変更」
- 4.5 「デバッガ/プログラマツール オプションを設定または変更する方法」
- 4.6 「言語ツールオプションを設定または変更する方法」
- 5.5 「ロード可能なプロジェクト、ファイル、シンボル」
- 4.9.5「ライブラリやオブジェクト ファイルの追加と設定」
- 4.10「ビルドプロパティの設定」

図12-38 [Project Properties]ウィンドウ



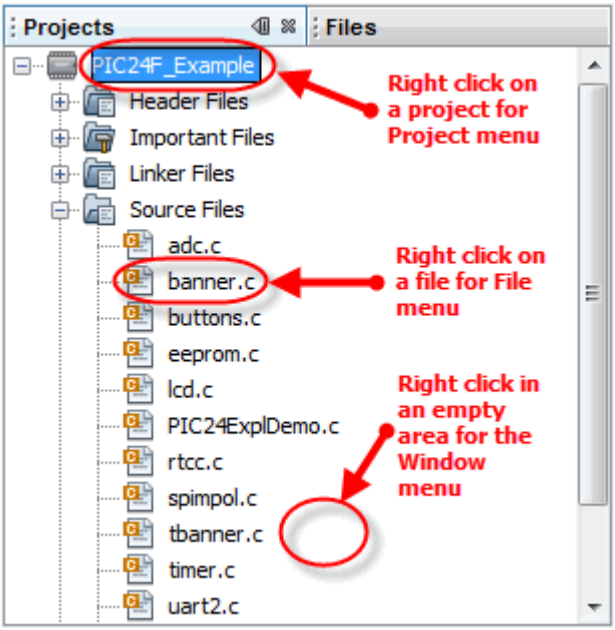
12.15 [Projects]ウィンドウ

このウィンドウは[Window] > [Projects]と選択して開きます。

[Projects]ウィンドウはNetBeansプラットフォーム ウィンドウです。しかし、MPLAB X IDEプロジェクトに関連する論理(仮想)フォルダを表示するために、カスタマイズしています。また、右クリックで開くコンテクスト メニューには、MPLAB X IDE専用の項目を追加しています。

- [Projects]ウィンドウ – 論理フォルダ
- [Projects]ウィンドウ – [Project]メニュー
- [Projects]ウィンドウ – [File]メニュー
- [Projects]ウィンドウ – [Window]メニュー

図12-39 [Projects]ウィンドウのコンテクスト メニュー



[Projects]ウィンドウ – 論理フォルダ

MPLAB Xプロジェクト用に論理フォルダを追加しています。そのため、表示された全てのフォルダは以下のファイルタイプのいずれかに分類できます。

- **Header Files** – MPLAB X IDEはビルド時にこのカテゴリを使いません。このフォルダは、ヘッダファイルに対するプロジェクトの依存性を示し、また、これらのファイルに容易にアクセスするために設けられています。[Projects]ウィンドウでファイルをダブルクリックすると、エディタで開きます。
- **Important Files** – 他のどのカテゴリにも属さないファイルは、全てこのフォルダに保存します。
  あるいは、プロジェクトに関連するデータシート(PDF)を、[Projects]ウィンドウ内のこのフォルダに追加しておけば、ダブルクリックでデータシートを開く事ができます(PDFファイルのリーダーをインストールしておく必要があります)。
- **Linker Files** – リンカスクリプトをプロジェクトに追加する必要はありません。プロジェクトの言語ツールが、デバイスに適合する汎用リンカスクリプトを自動的に選択するためです。従って、リンカスクリプトをユーザが自分で作成しない限りこのフォルダは空になります。
  ファイルは1つだけ保存します。複数のリンカスクリプトを保存した場合、最初の1つだけが有効となります。
- **Source Files** – ツールチェーンがファイルコマンドへの入力として受け入れるのは、これらのファイルのみです。
- **Libraries** – ツールチェーンは、このフォルダ内の全てのファイルとオブジェクト ファイルを最終リンクステップでインクルードします。
- **Loadables** – プロジェクトHEXファイルと結合または置換するプロジェクトとファイルです。

[Projects]ウィンドウ – [Project]メニュー

[Projects]ウィンドウでプロジェクト名を右クリックすると、[Project]メニューが開きます。下表にメニュー項目を示します。

表12-67 [Project]コンテクスト メニューの項目

メニュー項目	概要
New	ファイルを新規作成します。 MPLAB® X IDEでは組み込みファイルタイプを選択できます。
Add Existing Item	既存のファイルをプロジェクトに追加します。 ファイルへのパスは、[Auto] (MPLAB X IDEが選択)、[Absolute]、[Relative]のいずれかに設定できます。さらに、そのファイルをプロジェクト フォルダにコピーできます。

(続き)

メニュー項目	概要
Add Existing Items from Folders	指定したフォルダに含まれる全てのファイルを追加します。 また、特定のフォルダを含めないようにパターンを指定する事もできます。**[Default]**をクリックすると既定値のパターンを参照できます。また、[Tools] > [Options]で**[Miscellaneous]**ボタンをクリックし、**[Files]**タブで[Files ignored by the IDE]も参照してください。
New Logical Folder	新規の論理フォルダを作成します。 実際のフォルダの表示は、[Files]ウィンドウを参照してください。
Locate Headers	Cコード プロジェクト ファイル内で呼び出されている全てのヘッダファイル(.h)を検索し、プロジェクトに追加できるようにチェックリスト ウィンドウに表示します。 プロジェクトにヘッダを追加する理由は複数あります。 • PC上でヘッダファイルの保存場所を探す事なく、[Projects]ウィンドウの[Header Files]から開く事ができます。 • [Save Project As]でプロジェクトを作成した場合、ヘッダファイルはユーザが追加しない限りプロジェクトには含まれません。従って、ユーザ作成のヘッダファイルが必要な場合、このプロジェクトはビルドできません。 • [Package]で圧縮済みプロジェクトを作成した場合、ヘッダファイルはユーザが追加しない限りプロジェクトには含まれません。従って、ユーザ作成のヘッダファイルが必要な場合、これを解凍したプロジェクトはビルドできません。 この機能を使うと、Microchip社の全ツールチェーンのユーザ作成ヘッダファイルを検索できます。
Add Item to Important Files	既存のアイテムをプロジェクトに追加します。 ファイルへのパスは、[Auto] (MPLAB X IDEが選択)、[Absolute]、[Relative]のいずれかに設定できます。さらに、そのファイルをプロジェクト フォルダにコピーできます。 これは、シミュレータ ファイル、データシート(PDF)、その他のファイルを参考資料としてプロジェクトに追加するのに便利です。
Export Hex	プロジェクトのビルド結果をHEXファイルとしてエクスポートします。 MPLAB IDE v8ではメモリ オブジェクトの抽出結果をエクスポートします。MPLAB X IDEではビルドの結果を納めたHEXファイルをエクスポートします。従って、このファイルは、コード内のコンフィグレーションとEEPROMに必要な全ての補助的メモリ設定を含む必要があります。
Build	[Project Properties]ウィンドウで選択したオプションに従ってビルドします。 **Note:** [Debug]または[Run]を実行すると、プロジェクトは自動的にビルドされます。
Clean and Build	[Clean]を実行した後、[Project Properties]ウィンドウで選択したオプションに従ってビルドします。 **Note:** [Debug]または[Run]を実行すると、プロジェクトは自動的にビルドされます。
Clean	以前のビルドで生成されたファイルを削除してプロジェクトを空にします。
Package	現在のプロジェクトを1つのZIPファイルにアーカイブします。
Set Configuration	プロジェクトのコンフィグレーションを設定するために[Project Properties]ウィンドウを開きます。
Run	プロジェクト コードを実行します。
Debug	プロジェクト コードをデバッグ環境で実行します。
Step In	デバッグ環境で実行中の一時停止したコードを順番に実行します。
Make and Program Device	ターゲット デバイスをプログラミングした後、プログラムを実行せずにリセット状態に保持します。
Set as Main Project	このプロジェクトをメイン プロジェクトに設定します。 これは複数のプロジェクトを使う場合に便利な機能です。

(続き)

メニュー項目	概要
Open Required Projects	選択したプロジェクトの実行に必要なその他の全てのプロジェクトをロードします。
Close	選択したプロジェクトを閉じます。
Rename	プロジェクト名を変更します。
Move	プロジェクトを別の場所に移動します。 プロジェクトと一緒に、プロジェクト フォルダ内のソースファイルも移動します。プロジェクト フォルダ外部のファイルは移動しません。しかしプロジェクトは、プロジェクト フォルダ外のファイルへの参照を維持します。これは設計仕様に基づいた動作です。
Copy	プロジェクトのコピーを作成します。 ソースファイルがプロジェクト フォルダ内にある場合、ソースファイルも新規の場所にコピーします。プロジェクト フォルダ外にあるファイルはコピーしません。しかしプロジェクトは、プロジェクト フォルダ外のファイルへの参照を維持します。これは仕様です。
Delete	プロジェクト ファイルを削除します。 プロジェクト ファイルは削除しますが、プロジェクト フォルダ内のファイルは削除しません。[Also delete sources]を選択した場合のみ、全ソースファイルを削除します。
Code Assistance	コード生成支援機能(例: コード折り畳み、コード補完)を選択します。
Find	このプロジェクトのファイル内で特定のテキストを検索します。
Share on Team Server	チームサーバでこのプロジェクトを共有します。
Versioning	バージョン管理システムを使ってプロジェクトのバージョンを管理します。
Local History	プロジェクトのローカル履歴を表示します。または、そのローカル履歴に戻します。
Properties	プロジェクトのプロパティを設定します。 MPLAB X IDEの[Project Properties]ウィンドウは、組み込み開発専用にカスタマイズされています。

[Projects]ウィンドウ – [File]メニュー

[Projects]ウィンドウでファイル名を右クリックすると、[File]メニューが開きます。下表にメニュー項目を示します。

表12-68 [File]コンテクスト メニューの項目

メニュー項目	概要
Open	このファイルをエディタ ウィンドウで開きます。
Cut	ファイルをプロジェクトから削除し、クリップボードに移動します。
Copy	ファイルをクリップボードにコピーします。
Paste	クリップボードのファイルをプロジェクトに貼り付けます。
Exclude file(s) from current configuration	現在のプロジェクトの設定から除外するファイルを選択します。ファイルを除外した後でそのファイルをもう一度右クリックすると、[Include file(s) from the current configuration]が表示されます。[Project Properties]ウィンドウを開き、[File Inclusion/Exclusion]を選択してファイルを含める/除外するのと同じです。
Compile File	現在のファイルのみをコンパイルします。
Remove from Project	ファイルをプロジェクトから削除します。 この操作では、ファイルはPCから削除されません。ファイルをプロジェクトとコンピュータから削除するには、<Delete>キーを使います。

(続き)

メニュー項目	概要
Rename	ファイル名を変更します。
Save as Template	現在のファイルをテンプレート ファイルとして保存します。
Local History	ファイルのローカル履歴を表示します。または、そのローカル履歴に戻します。
Tools	ファイルツールは以下を含みます。 • **Apply Diff Patch**: Diffが作成した既存のパッチを適用します。 • **Diff to**: 2つのファイルの差分を表示します。 • **Add to Favorites**: このファイルを[Favorites]ウィンドウに追加します。
Properties	プロジェクト プロパティとは異なるファイル プロパティを設定します。 「Exclude from build」にチェックを入れると、このファイルをプロジェクトのビルドから除外します。「Override build options」にチェックを入れると、ここで選択したビルドオプションをプロジェクト コンフィグレーション内のビルドオプションよりも優先させます。

[Projects]ウィンドウ – [Window]メニュー

[Projects]ウィンドウの空いた場所で右クリックすると、[Window]メニューが表示されます。下表に専用のメニュー項目を示します。

表12-69 [File]コンテクスト メニューの項目

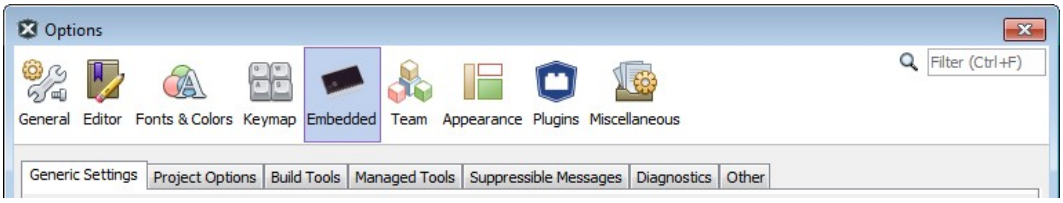
メニュー項目	概要
New Project	[New Project]ウィザードを起動します。
New File	[New File]ウィザードを起動します。
Open Project	既存のプロジェクトを開きます。
Open Recent Project	最近編集したプロジェクトの一覧の中から、既存のプロジェクトを開きます。
Open Project Group	プロジェクト グループの一覧の中から、既存のプロジェクト グループを開きます。
Run Project	メイン プロジェクトを実行します。
Set Main Project	開かれているプロジェクトの一覧の中から、メイン プロジェクトを設定します。

12.16 [Tools] > [Options]ウィンドウ、[Embedded]

[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択して**[Embedded]**ボタンをクリックすると、下図に示すタブとオプションが表示されます。

[Options]ウィンドウはNetBeansプラットフォームのウィンドウです。ただし、MPLAB X IDEプロジェクト用にカスタマイズされています。

図12-40 [Tools] > [Options] > [Embedded]タブ



12.16.1 [Generic Settings]タブ

[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択して**[Embedded]**ボタン、**[Generic Settings]**タブをクリックすると、以下のオプションが表示されます。

表12-70 [Generic Settings]タブの項目

項目	説明
Open source file and locate line in editor when debugger halts	ブレークポイントのように停止したコード行にジャンプし、コードを確認できます。 無効にすると単に実行を停止します。
Clear tool output window on new session (debug, program, upload)	実行、デバッグ、アップロードを開始する際に、[Output]ウィンドウの内容をクリアします。
Halt build on first failure	ビルド時に、障害が最初に発生した時点でプロセスを停止します。 どのエラーが発生したらプロセスを停止するかは、選択したプロジェクト言語ツールで設定できます。[Project Properties]ウィンドウの[Categories]で言語ツールを選択します。[Option Categories] では、エラー、警告、メッセージの設定もできます。
Maintain active connection to hardware tool	チェックを入れると、実行時以外もハードウェア ツールを常時接続したまま(MPLAB® IDE v8の挙動)にします。 この機能を有効にしたまままするようにプロジェクトを切り換える場合(例: ブートローダ アプリケーションの開発)、ツールとデバイスの設定が同じである事を確認します。異なる場合、エラーメッセージが表示されます。 **Note:** このオプションは現在のプロジェクトとコンフィグレーションにのみ適用されます。プロジェクトを切り換えるか、[Project Properties]で項目を変更すると、再接続が行われます。
Read Device Memory to File: Export only memory used	[Read Device Memory to File]アイコン/機能では、ファイルに使われているデバイスメモリのみを保存します。
Silent build	ビルドの際に、[Output]ウィンドウにメッセージを表示しません。
Enable alternate watch list views during debug session	[Watches]ウィンドウに3つのウォッチビュー ダイアモンドを表示し、ウォッチビューを観察する変数に関連付けます。ウォッチ ビュー ダイアモンドをクリックすると、そのビューに関連付けられた変数だけが表示されます。いわばフィルタのような機能です。
Disable auto refresh for call stack view during debug sessions	有効にすると、一時停止または停止時に内容を自動で更新します。 ウィンドウが閉じられている場合、またはフォーカスされていない場合、ウィンドウを選択しこのボタンをクリックすると更新が表示されます。再度実行および一時停止/停止する必要はありません。 無効にすると、一時停止または停止時にウィンドウを手動で更新します。 一時停止/停止時にウィンドウの内容を更新するには、このボタンをクリックする必要があります。
On mouse over structure and array expressions, evaluate integral members only	チェックを入れると、コードの構造体または配列の式をマウスオーバーした際、必須メンバーのみを表示します。 チェックを外すと、コードの構造体または配列の式をマウスオーバーした際、全てのメンバーを表示します。
Show unresolved variable names in watch window during debug session	チェックを入れると、[Watches]ウィンドウに未解決変数を表示します。
Allow switch between pack versions	チェックを入れると、[Project Properties]に表示するパックを切り換える事ができます。
Enable gathering of compiler symbols	コンパイラをバックグラウンドで実行し、コンパイラのビルトイン/プライベート マクロに関する情報を要求します。 既定値ではこのオプションは無効です。有効にすると、IDEがシンボル情報を収集する時間が長くなります。 **Note:** コンパイラの内部情報を使わない場合、この機能を有効にする必要はありません。

(続き)

項目	説明
On mouseover source lines in editor, evaluate break point status	評価をトリガする方法を選択します。 **Disable**: 評価なし **Mouse Only, Alt+Mouse, Shirt+Mouse, Shift+Alt+Mouse**: これらのキー操作のいずれかにより、ブレークポイントで値を評価します。
Hold-off period before memory view synchronization: Give priority to debug events (step over, step in, continue)	デバイスが停止してからデバイスのデータがメモリウィンドウに同期するまでの間の遅延時間を指定します。 長い遅延時間を設定すると、更新の前により多くのシングルステップ実行が発生します。この機能は、ファイルレジスタとデータメモリ等の一般的な[PIC Memory Views]のみに適用できます。
Debug Reset @ (following reset action during paused debug session)	デバッグリセット時の動作を選択します(例: [Debug] > [Reset])。 **Main**: リセット時に`main()`で停止します。 **Reset Vector**: リセット時にリセットベクタ位置で停止します。 **Note:** これは、デバイスのハードウェア リセットとは関係ありません。
Debug start-up (following debug project action)	[Debug] > [Debug Project]のコード実行を指定します。 **Run**: 即座に実行を開始します。 **Main**: `main()`で停止します。 **Reset Vector**: リセットベクタ位置で停止します。
Default Charset	プロジェクトの既定値の文字セットを選択します。

12.16.2 [Project Options]タブ

Select [Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択して**[Embedded]**ボタン、**[Project Options]**タブをクリックすると、以下のオプションが表示されます。

表12-71 [Project Options]タブの項目

項目	説明
Make Options	プロジェクトをビルドする際に使うmakeオプションを入力します。 これらのオプションは、ツールチェーンによって異なります。詳細は使用する言語ツールの文書を参照してください。
File Path Mode	プロジェクト内のファイルパス情報の保存方法を指定します。 **Auto**: プロジェクト フォルダ内のファイルへのパスを相対パスとして保存し、プロジェクト フォルダ外のファイルへのパスを絶対パスとして保存します。 **Always Relative**: 全てのパスを、プロジェクト フォルダに対する相対パスとして保存します。 **Always Absolute**: 全てのパスを絶対パスとして保存します。
Save All Modified Files Before Running Make	これを選択した場合、makeを実行する前に、IDE内の全ての未保存ファイルを保存します。このチェックは入れたままにしておく事を推奨します。外した場合、変更したファイルをあらかじめ保存しておかないと、ファイルへの変更がmakeに反映されません。
Reuse Output Tabs from Finished Process	選択すると、[Output]ウィンドウの同じタブに出力メッセージが表示されます。選択しない場合、新規プロセスごとに新規タブが開きます。

(続き)

項目	説明
Use parallel make (`make -j 2n`)	これにチェックを入れると、makeによって複数のプロセスが同時に実行されます。`-j`(または`--jobs`)は並列実行を指定するオプション、2nはプロセス数、nはコンピュータで使用可能なプロセッサ数です。コンピュータが並列処理をサポートしていない場合、並列makeは無効化されます。 また、[Make Options]で2より大きいプロセスを指定する事も可能です例: `-j 10`)。 **Note:** MPLAB XC16またはMPLAB C30の場合、並列makeを使うには手続きの抽象化を無効にする必要があります([File] > [Project Properties]、コンパイラ カテゴリ、[Optimizations]オプション カテゴリで、[Unlimited procedural abstraction]のチェックを外します)。 **Note:** MPASMXは、ツールチェーンとして使う場合もMPLAB C18プロジェクトの一部として使う場合も並列makeでは実行できません。並列make機能は、MPASMXツールチェーンを使うプロジェクトまたはC18ツールチェーンを使うプロジェクト(`.asm`ファイルを含む)では無視されます。
Force makefile regeneration when opening a project	チェックを外すと、プロジェクトが既に開いている場合(例: 別のコンピュータ上でプロジェクトを開いている場合)のmakefileの再生成の要否をMPLAB IDEが決定します(これが既定値です)。 チェックを入れると、プロジェクトを開いた時にmakefileの再生成を強制的に実行します。makefileを再生成する必要があるにもかかわらず再生成されない場合、この機能を使います。 **Note:** 再生成には若干の時間を要します。
Use “clean” target from Makefile	チェックを外すとstandard cleanが使われます。すなわち、ビルド時に生成された全てが削除されます。この方法がより高速です。 チェックを入れるとmake cleanが使われます。すなわち、clean時にMakeファイルが使われます。この方法では、削除対象をより詳細に設定できます。

12.16.3 [Build Options]タブ

[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択して**[Embedded]**ボタン、**[Build Options]**タブをクリックすると、以下のオプションが表示されます。

インストールしていない言語ツールは[Toolchain]リストに表示されません。インストールしたはずの言語ツールがこのリストに表示されない場合、[Scan for compilers]をクリックします。それでも表示されない場合、[Add]をクリックして、そのツールをリストに追加します。

以下の言語ツールはMPLAB X IDEに付属しています。

- MPASM toolchain – MPASMアセンブラ、MPLINKリンカ、各種ユーティリティを含みます。

その他のツールは、Microchip社ウェブサイト(www.microchip.com)またはサードパーティから入手できます。

表12-72 [Build Tools]タブの項目

項目	説明
Toolchain	インストールされている言語ツールのリストを表示します。 プロジェクトに使うツールを選択し、右側に表示されるパスが正しい事を確認します。 **Base directory**: ツールのメインフォルダへのパス **C compiler**: Cコンパイラへのパス(利用可能な場合) **Assembler**: アセンブラへのパス(利用可能な場合) **Make command**: MPLAB® X IDEが生成したmakeコマンドの名前
Add	リストに新しい言語ツールを追加します。**[Scan for Build Tools]**を使う方法もあります。
Add Custom Compiler	カスタマイズしたコンパイラを追加します。 この例では、サポートしているMicrochip社製デバイスを指定する必要があります。

(続き)

項目	説明
Remove	リストから言語ツールを削除します。 言語ツールそのものはPCから削除されません。
Default	ツールをクリックして**[Default]**を選択すると、選択したデバイス用の既定値コンパイラ/アセンブラとなります。
Scan for Build Tools	インストールされているコンパイラ/アセンブラをスキャンします。これは、システム全体ではなく、複数の既定値フォルダだけをスキャンします。 それ以外の場所にインストールした場合、手動でコンパイラを追加する必要があります。

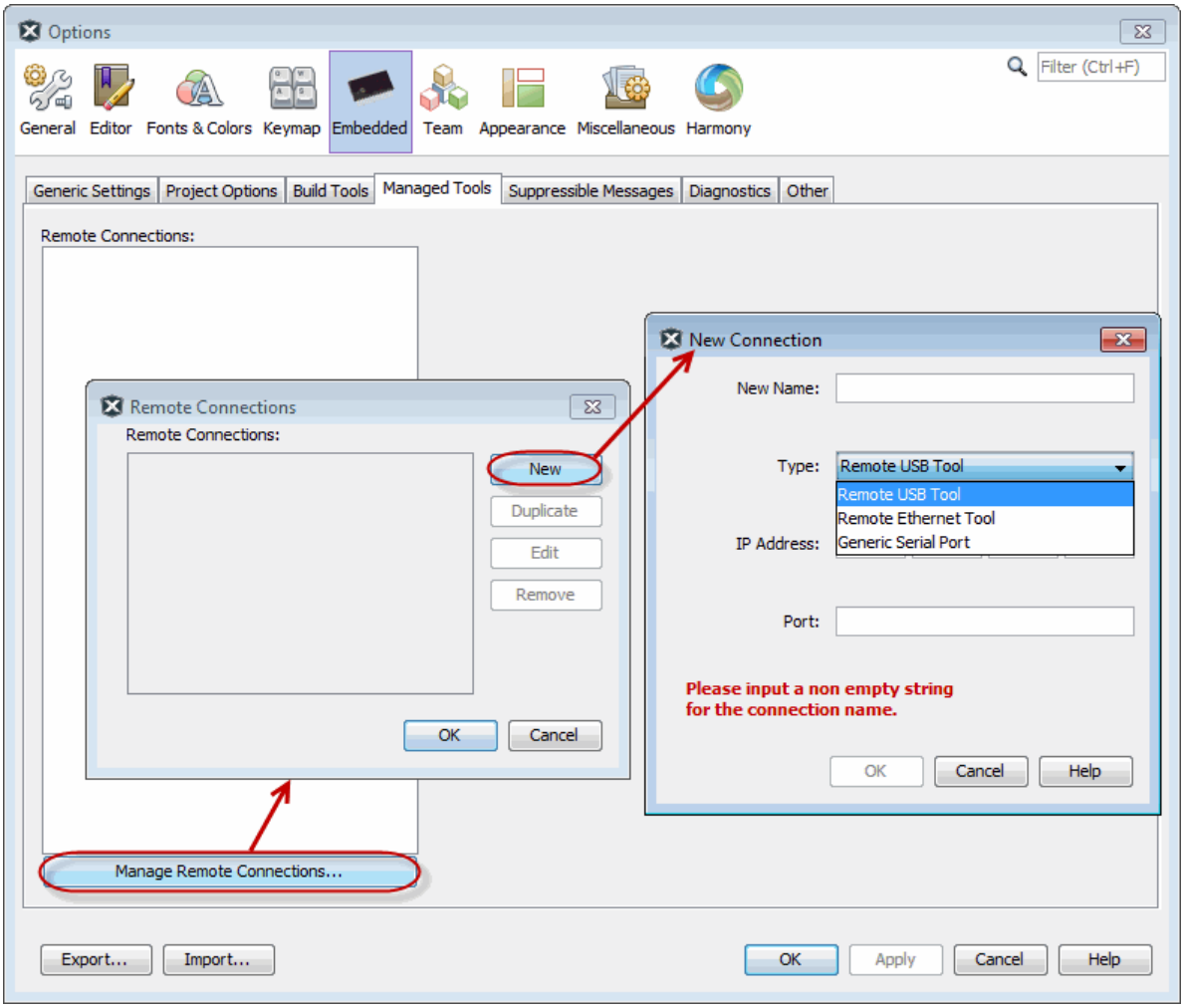
12.16.4 [Managed Tools]タブ

[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択して**[Embedded]**ボタン、**[Managed Tools]**タブをクリックすると、以下のオプションが表示されます。

使用するリモート接続を選択します。以下の接続を新たに追加します。

- Remote USB Tool (プラグインが必要)
- Remote Ethernet Tool
- Generic Serial Port

図12-41　新規の管理ツール

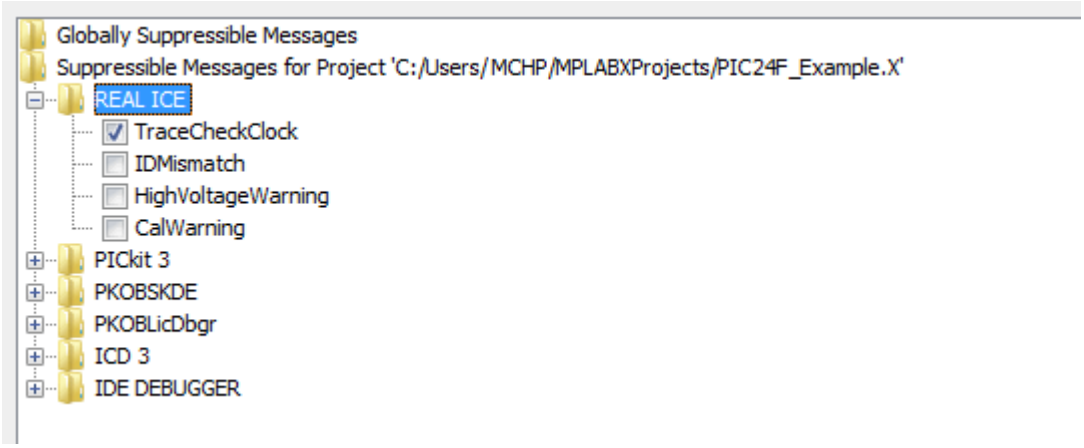


12.16.5 [Suppressible Messages]タブ

[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択して**[Embedded]**ボタン、**[Suppressible Messages]**タブをクリックすると、以下のオプションが表示されます。

抑止するエラーおよび警告メッセージを選択します。表示されたフォルダをダブルクリックして、抑止可能な項目を表示させてチェックを入れます。

図12-42 抑止可能なメッセージの一覧



12.16.6 [Diagnostics]タブ

[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択して**[Embedded]**ボタン**[Diagnostics]**タブをクリックすると以下のオプションが表示されます。

ログファイル等の診断機能を設定します。

表12-73 [Diagnostics]タブの項目

項目	説明
Logging Level	メッセージログのレベルを設定します。 **OFF**: 記録しません。 **SEVERE**: 重大な(エラー) メッセージのみ記録します。 **WARNING**: 警告メッセージのみ記録します。 **INFO**: 情報メッセージのみ記録します。 **CONFIG**: コンフィグレーション情報のみ記録します。 **FINE**: 一部のモジュール間通信を記録します。 **FINER**: FINEよりも詳細にモジュール間通信を記録します。 **FINEST**: 全てのモジュール間通信を記録します。
Log File	ログファイルのパスと名前です。
USB Circular Log	MPLAB REAL ICEインサーキット エミュレータ、MPLAB ICD 3、PICkit 3でのみ利用できます。
Start New Logging Session/Pause Logging	ボタンをクリックすると、新規のログセッションが開始します。または、進行中のログセッションが一時停止します。
Circular USB log file location	ログファイルを保存する場所(パス)を指定します。
Circular USB log file max size in KBs	ログファイルのサイズを指定します。

12.16.7 [Other]タブ

[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択して**[Embedded]**ボタン、**[Other]**タブをクリックすると、以下のオプションが表示されます。

C/C++およびアセンブラ ソースファイルとヘッダファイル用に使えるファイル拡張子のリストを編集します。また、各ファイルタイプの既定値拡張子(太字で表示)も設定します。

表12-74 [Other]タブの項目

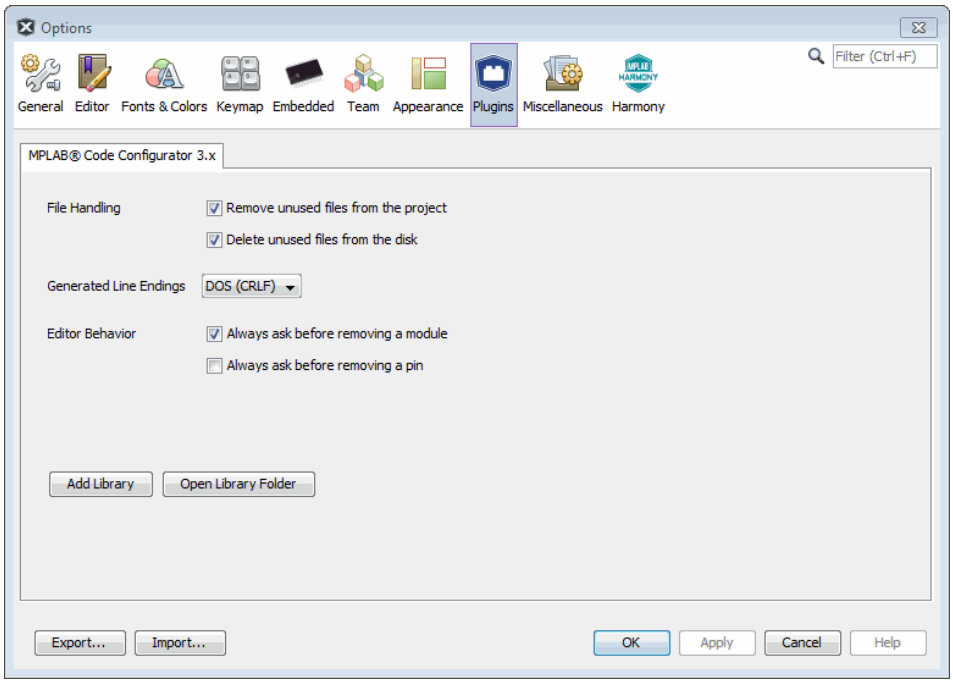
ファイルの種類	既定値の拡張子
C/C++ヘッダ	H, SUNWCCh, **h**, hpp, hxx, tcc
C++	C, c++, cc, **cpp**, cxx, mm
C	**c**, i, m
Fortran	なし MPLAB X IDEはFortranによるプログラミングをサポートしていません。
アセンブラ	AS, ASM, S, as, asm, **s**
アセンブラ インクルード	INC, **inc**

12.17 [Tools Options]ウィンドウ、[Plugins]

このウィンドウは、[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])と選択して**[Plugins]**ボタンをクリックすると開きます。

[Options]ウィンドウはNetBeansプラットフォームのウィンドウです。ただし、MPLAB X IDEプロジェクト用にカスタマイズされています。[Tools] > [Plugins]を使ってプラグインをインストールしており、これらのプラグインがオプションを備えている場合、利用可能なプラグイン オプションはこのウィンドウに表示されます。

図12-43 プラグイン オプション例



12.18 [Trace]ウィンドウ

[Trace]ウィンドウは[Window] > [Debugging] > [Trace]と選択して開きます。トレースデータを表示するにはデバッグモード中である事が必要です。

トレースは、[Trace]ウィンドウでコード実行をステップごとに記録し、記録された結果を検討する機能です。現在トレース機能を利用できるツールは以下の通りです。

- シミュレータ
- MPLAB REAL ICEインサーキット エミュレータ

ウィンドウでトレース列を右クリックすると、コンテクスト メニュー(以下の1番目の表)が開きます。表示されるメニュー項目は使うツールによって異なります。

以下の2番目の表に、トレースに関連するダイアログを示します。

表12-75 [Trace]ウィンドウのコンテクスト メニュー

メニュー項目	概要
Symbolic Mode	[Instruction]列の表示を、リテラルレジスタ アドレス(例: 0x5)とシンボリック レジスタマクロ(例: PORTA)の間でトグルします。
Go To	[Go To]ダイアログが開きます。 **Trigger**: トリガ行(0)に移動します。 **Top**: トレース行の先頭に移動します。 **Bottom**: トレース行の末尾に移動します。 **Trace Line**: 指定したトレース行に移動します。
Go To Source Line	トレース行を選択してこのオプションを選択すると、ソースコード内の対応する行に移動します。
Display Time	[Cycle]列の表示を以下のように設定します。 **As Hex Cycle Count**: サイクル数を16進数で表示します。 **As Decimal Cycle Count**: サイクル数を10進数で表示します。 **In Seconds Elapsed**: サイクル数を経過秒数で表示します。 **In Engineering Format**: サイクル数を適切なエンジニアリング フォーマット(10^3の累乗)で表示します。
Clear Trace File	トレース表示内のデータをクリアします。
Find	トレース表示内の項目を検索します。[Find]ダイアログが開きます。
Output To File	トレースデータをファイルに保存します。 [Define Range]ダイアログが開き、そこから[Save]ダイアログを開きます。
Print	データを印刷します。 [Print]ダイアログが開きます。
Adjust Table Columns	トレース表示の列幅をデータに合わせて自動調整します。
Reload View	一時停止時にトレース表示の元のデータビューを再ロードします。

表12-76 [Trace]ダイアログ

ダイアログ	説明
Go To	移動するトレース行を指定します。
Find	トレース表示で行番号やその他のデータを検索します。
Output to File Range	ファイルに出力する行範囲を指定します。 範囲指定後**[OK]**をクリックして、[Save]ダイアログに進み、データをテキストファイルに保存します。
Save	データをテキストファイルに保存します。

(続き)

ダイアログ	説明
Print	印刷前にプリンタ、ページ設定、表示設定を指定します。

12.19 [Watches]ウィンドウ

[Watches]ウィンドウは[Window] > [Debugging] > [Watches]と選択して開きます。シンボル値の変化を観察するにはデバッグモード中である事が必要です。

グローバル シンボルとSFRを選択してプログラム停止時または(サポートされている場合)実行中の値の変化を観察します。[Watches]ウィンドウの使い方は、4.16「シンボル値の変化の観察」を参照してください。

12.19.1 [Watches]の表示

以下のセクションで表示機能(アイコン、列、操作(ボタン))を説明します。

アイコン

アイコンは[Name]行の名前のオブジェクトの左に表示されています。

表12-77[Name]行のアイコン

アイコン	説明
	ウォッチ オブジェクト
	プログラムメモリのウォッチ オブジェクト
	オブジェクトの静的フィールド
	オブジェクトの非静的フィールド

列

ウィンドウに表示する列は以下の方法で変更できます。

- 見出しを右クリックして、表示する見出しをトグルします。
- ウィンドウの左上にある[Change Visible Columns]アイコンをクリックします  。[Change Visible Columns]ダイアログで、表示する見出しをトグルします。

操作

各種操作はウィンドウ左側のボタンで行います。

表12-78 操作ボタンのアイコン

ボタン	操作
	トグルボタン: • メンバー フィールドの詳細名(正規の名前)を表示します。 • メンバー フィールドの要約名(相対的な名前)を表示します。
	リストファイルからウォッチをインポートします。右クリックしてインポート方法を選択します。
	全てのウォッチをリストファイルにエクスポートします。

(続き)

ボタン	操作
	[Value]フィールドの既定値数値フォーマットを設定します。

12.19.2 [Watches]メニュー

[Watches]ウィンドウ内の(行ではなく)空いた場所で右クリックすると、基本的な機能を備えた[Watches]ウィンドウ メニューが表示されます。

表12-79 [Watches]ウィンドウ メニュー – ウィンドウ

メニュー項目	概要
New Watch*	ウォッチに新しいシンボルを追加します。
New Runtime Watch*	**MPLAB REAL ICEインサーキット エミュレータ、シミュレータのみ** 選択したシンボルに新しいランタイム ウォッチを追加します。
Export All Watches to List File	観察するシンボルに関する情報をファイル(.xwatch)にエクスポートします。
Delete All	[Watches]ウィンドウから観察対象シンボルを全て削除します。
* コード内のシンボルを右クリックして、[New Watch]または[New Runtime Watch]に追加する事もできます。	

行を右クリックすると、その他の機能を備えた[Watches]ウィンドウ メニューが表示されます。表示されるメニュー項目は、選択した行がシンボルかどうか、そして使うデバッグツールによって異なります。

表12-80 [Watches]ウィンドウ メニュー – 行

メニュー項目	概要
New Watch	ウォッチに新しいシンボルを追加します。
New Runtime Watch	**MPLAB REAL ICEインサーキット エミュレータのみ** 選択したシンボルに新しいランタイム ウォッチを追加します。
Run Time Update Interval	**MPLAB REAL ICEインサーキット エミュレータのみ** シンボル値の更新速度を指定します。 [No Delay]を選択すると、コンピュータが処理できる最大速度で更新します。実行時データにエラーが発生する場合、遅延を追加して更新速度を緩めます。
Export Selected Watches to List File	観察するシンボルに関する情報をファイル(`.xwatch`)にエクスポートします。 <Shift>+クリック(連続した複数行を選択)または<Ctrl>+クリック(複数行を個別に選択)を使って複数の行を選択できます。
Export Data	ウォッチデータをCSVファイルにエクスポートします。または、文字列リテラルとしてエクスポートします。
Delete	選択したシンボル行をウォッチリストから削除します。行を選択して<Delete>キーを押しても削除できます。
Delete All	[Watches]ウィンドウから観察対象シンボルを全て削除します。
Display Value Column as	選択した行のシンボルの値を、フォーマット一覧のうちの1つで表示します。以下も参照してください。 12.19.3「[Fractional Integer Properties]ダイアログ」
User Defined Size	プログラムメモリのシンボルのサイズを一覧から指定します。

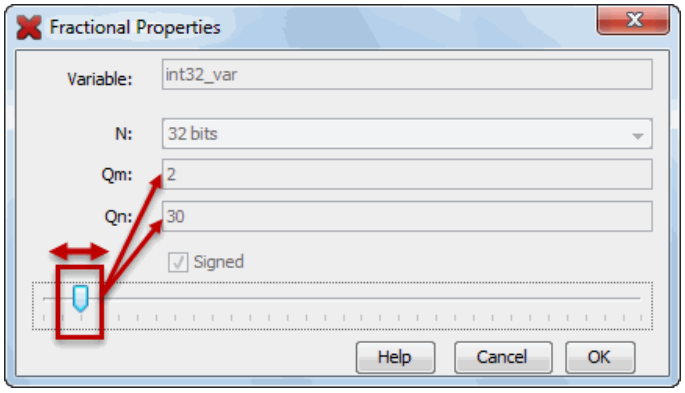
12.19.3 [Fractional Integer Properties]ダイアログ

このダイアログの目的は、dsPICおよびPIC32デバイスがサポートする全ての固定小数点演算モードを評価しやすくする事です。ポップアップ メニュー項目[Display Value Column as] > [Fractional Integer]からこのダイアログを表示してこの機能を選択します。その後、必要に応じて[Display Value Column as] > [Fractional Properties]を選択します。

表12-81 [Fractional Properties]ダイアログの項目

ダイアログ項目	説明
Variable	変数の名前です(参照用)。
N*	整数を表現するために利用可能な総ビット数です。
Qm	固定小数点データ形式の整数部です。 「m」は整数を指定するのに使うビット数を表します。
Qn	固定小数点データ形式の小数部です。 「n」は小数を指定するのに使うビット数を表します。
Signed*	チェックを入れると、1ビットを使って符号付き値を表します。
Slider	スライダをクリックおよびドラッグして、固定小数点値の計算に適用する固定小数点の位置(Qm対Qn)を選択します。
* 変数タイプが未知の場合、これらの項目は編集可能です。	

図12-44 [Fractional Properties]ダイアログ

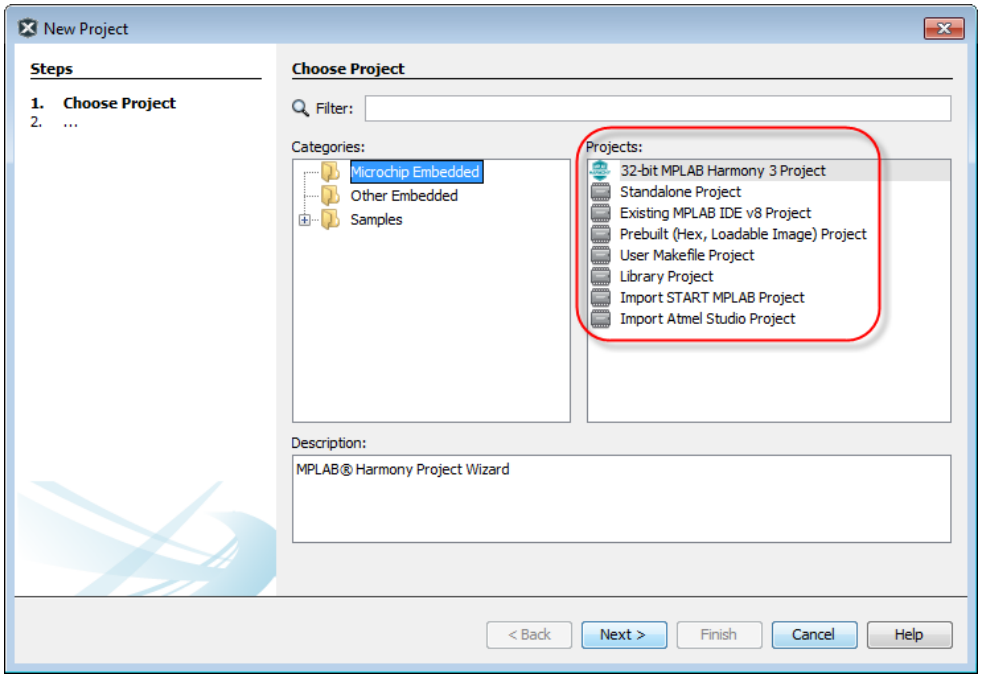


12.20 ウィザード

ウィザードは手順をガイドする一連のダイアログ ウィンドウです。MPLAB X IDEで良く使うウィザードは以下の2つです。

- [New Project]ウィザード(「新規プロジェクトの作成」参照)。
- [New File]ウィザード(「ファイルの新規作成」参照)

図12-45 新規プロジェクト タイプ



13. NetBeansのウィンドウとダイアログ

MPLAB X IDEのウィンドウとダイアログは、NetBeansのウィンドウ/ダイアログとMPLAB X IDE専用のウィンドウ/ダイアログを組み合わせています。ヘルプトピックは、NetBeans項目を白、MPLAB X IDE項目を黄色の背景で表示します。

ここでは、NetBeansのウィンドウとダイアログを解説します。MPLAB X IDE専用のウィンドウとダイアログは12.「MPLAB X IDEのウィンドウとダイアログ」を参照してください。

13.1 NetBeansウィンドウとウィンドウ メニュー

NetBeansウィンドウの説明は、NetBeansヘルプまたはウェブ上の文書に記載されています(「About This Help File」または「Preface」参照)。ウィンドウに関するヘルプを開くには、[Window]タブをクリックで選択してから<F1>キーを押します。ヘルプが見つからない場合、[Help] > [Help Contents]を選択してヘルプファイルの**[Search]**タブをクリックし、そのウィンドウの情報を検索するか、NetBeansヘルプトピックの目次「Managing IDE Windows」を参照します。

大部分のウィンドウは、10.1.11「ウィンドウ メニュー」に記載したメニューから開く事ができます。

ウィンドウはドッキング/ドッキング解除でき(ウィンドウタブを右クリック)、そのウィンドウ専用のポップアップメニューまたはコンテクスト メニューから例えば以下の項目を選択できます。

- Fill
- Goto
- Edit Cells

コンテクスト メニューは、ウィンドウ内またはウィンドウ内のいずれかのアイテム上で右クリックすると開きます。これらのコンテクスト メニューの大部分は、デスクトップ メニューバーからも利用できます(10.1「メニュー」参照)。

[Tools] > [Options] (macOSでは[MPLAB X IDE] > [Preferences])を選択して**[Miscellaneous]**ボタン、**[Appearance]**タブをクリックし、ウィンドウのプロパティを設定します。

13.2 NetBeansダイアログ

NetBeansダイアログの説明は、NetBeansヘルプに記載されています。ダイアログで**[Help]**ボタンをクリックすると、そのダイアログに関するヘルプが開きます。このボタンがない場合、<F1>キーを押します。必要なヘルプが見つからない場合、[Help] > [Help Contents]を選択し、ヘルプファイルの**[Search]**タブをクリックすると、ウィンドウ上で情報を検索できます。

大部分のダイアログは、10.1「メニュー」に記載したメニューから開く事ができます。

14. コンフィグレーション設定のまとめ

本章では、コード内でコンフィグレーション ビットを設定する方法を、各種の言語ツールと関連デバイス向けに例を示しながら説明します。コンフィグレーション ビット設定の詳細は言語ツールのマニュアルを参照してください。一部の言語ツールには、デバイスの全てのコンフィグレーション設定の一覧を記載したコンフィグレーション設定マニュアルが用意されています。それ以外の言語ツールに関しては、デバイスのマクロ用ヘッダファイルを参照してください。

コンフィグレーション ビットを設定するもう1つの方法は、[Configuration Memory]ウィンドウを使ってビットを設定し、[Generate Source Code to Output]をクリックする事です。4.19「[Configuration Bits]ウィンドウでのコンフィグレーション値の設定」を参照してください。

8ビットおよび16ビットデバイス向けのMPLAB Code Configurator (MCC)と32ビットデバイス向けのMPLAB Harmonyは、コンフィグレーション ビットとプロジェクト コードの生成に使えるツールです。これらのツールの詳細はMicrochip社のウェブサイトを参照してください。

最新のツールチェーン: AVR/Arm GCCコンパイラ、MPLAB XC Cコンパイラ、MPASMアセンブラ

レガシー ツールチェーン: HI-TECH PICC/PICC18コンパイラ、MPLAB C18コンパイラ、ASM30アセンブラ、MPLAB C30/MPLAB C32コンパイラ

14.1 AVR GCCツールチェーン

大部分のコンフィグレーション ビットはデバイスのヒューズバイトに格納されています。ヒューズビットをプログラミングするには、Fuse APIを使います。このAPIを使うには、`<avr/io.h>`ヘッダファイルをインクルードします。このファイルはAVRヒューズを設定するために必要な全てを備えています。

コード保護コンフィグレーション ビットまたはロックビットをプログラミングするには、Lockbit APIを使います。このAPIを使うには、
`<avr/io.h>`ヘッダファイルをインクルードします。

`FUSES`および`LOCKBITS`マクロの使用例を以下に示します。Fuse APIとLockbit APIの詳細は、以下を参照してください。

https://www.microchip.com/webdoc/AVRLibcReferenceManual/group__avr__fuse.html

https://www.microchip.com/webdoc/AVRLibcReferenceManual/group__avr__lock.html

例: ATtiny861 MCU

```
// ATtiny817 Configuration Bit Settings

// 'C' source line config statements

#include <avr/io.h>

FUSES = {
    0x00, // WDTCFG{PERIOD=OFF, WINDOW=OFF}
    0x02, // BODCFG{SLEEP=SAMPLED, ACTIVE=DIS, SAMPFREQ=1KHz, LVL=BODLEVEL0}
    0x00, // OSCCFG{OSCLOCK=CLEAR}
    0x00, // Reserved
    0xC4, // TCD0CFG{CMPA=CLEAR, CMPB=CLEAR, CMPC=SET, CMPD=CLEAR, CMPAEN=CLEAR,
CMPBEN=CLEAR, CMPCEN=SET, CMPDEN=SET}
    0xF6, // SYSCFG0{EESAVE=CLEAR, RSTPINCFG=UPDI, CRCSRC=NOCRC}
    0x00, // SYSCFG1{SUT=0MS}
    0x00, // APPEND
    0x00, // BOOTEND
};

LOCKBITS = {
    0xC5, // LOCKBIT{LB=NOLOCK}
};
```

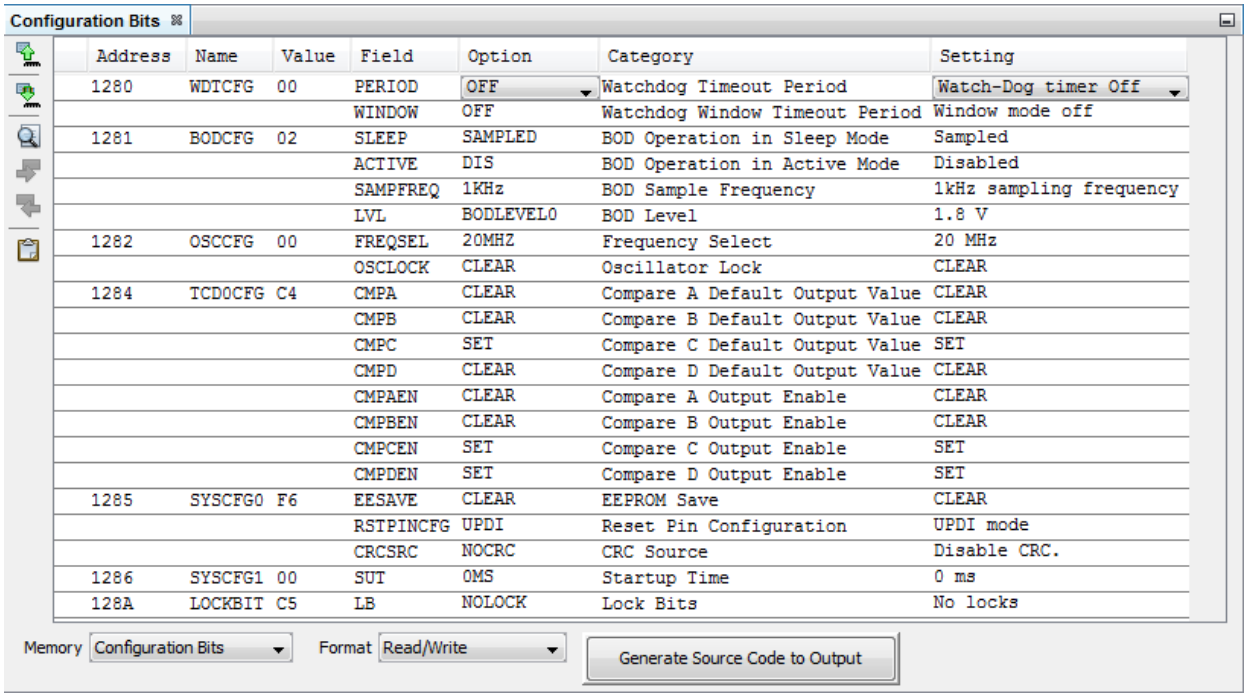
上の例は、[Configuration Bits]ウィンドウから生成しました([Window] > [Target Memory Views] > [Configuration Bits])。

初めて[Configuration Bits]ウィンドウを開くと、データは赤で表示されています。つまり、デバイスメモリから読み出す必要があります。[Read Configuration Bits]アイコンを使います。この[Program Configuration Bits]アイコンを使ってウィンドウ内でデータを変更し、デバイスに書き戻します。

設定をコードに保存するには、[Generate Source Code to Output]ボタンをクリックして

[Output]ウィンドウにコードを置いてコピー&ペーストするか、[Insert Source Code into Editor]アイコンをクリックしてエディタ ウィンドウ内のカーソル位置にコードを挿入します。

図14-1　例: ATtiny817 MCU

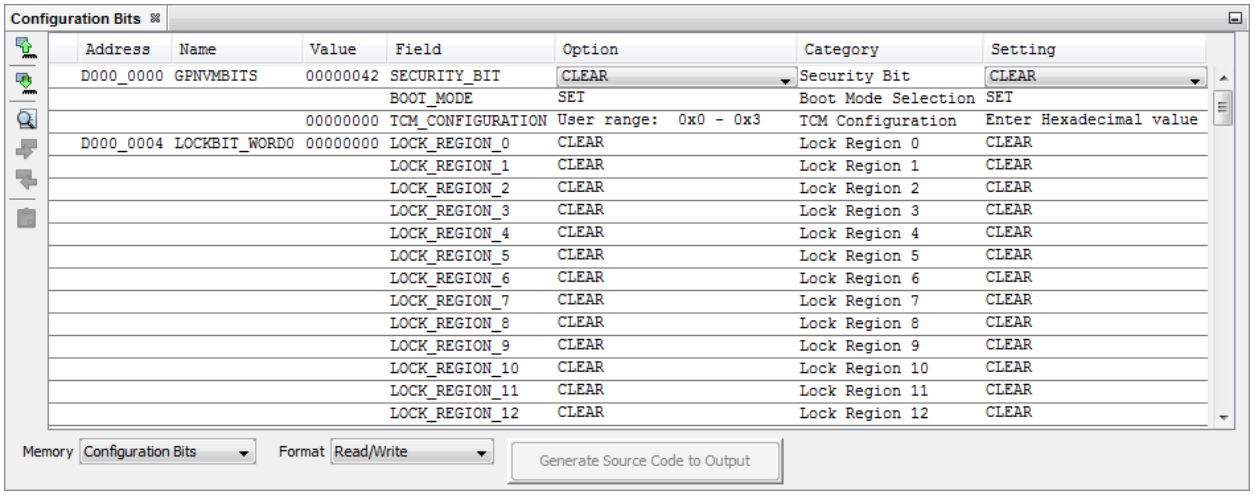


14.2　Arm GCCツールチェーン

大部分のコンフィグレーション ビットはデバイスのGPNVMビットに格納されています。具体的には、GPNVM0はセキュリティビットを表します。コード保護コンフィグレーション ビットをプログラミングするには、ロックビットを使います。

コンフィグレーション ビットをプログラミングするには、[Configuration Bits]ウィンドウ([Window] > [Target Memory Views] > [Configuration Bits])を使います。初めて[Configuration Bits]ウィンドウを開くと、データは赤で表示されています。つまり、デバイスメモリから読み出す必要があります。[Read Configuration Bits]アイコンを使います。この[Program Configuration Bits]アイコンを使ってウィンドウ内でデータを変更し、デバイスに書き戻します。

例: ATSAME70Q21 MCU



14.3 XCツールチェーン

コードの移植性を最大限に高める、以下の各文書の「CCI (Common Compiler Interface)」の章、`config` macroのセクションを参照してください。

- 『MPLAB XC8 C Compiler User's Guide』(DS50002053)または対応するヘルプファイル
- 『MPLAB XC16 Cコンパイラ ユーザガイド』(DS50002071)または対応するヘルプファイル
- 『MPLAB XC32 C/C++コンパイラ ユーザガイド』(DS50001686)または対応するヘルプファイル

コンフィグレーション ビットの読み出し、変更、書き込みの方法は4.19「[Configuration Bits]ウィンドウでのコンフィグレーション値の設定」を参照してください。

14.4 MPASMツールチェーン

デバイス コンフィグレーション ビットの設定には、2種類のアセンブラ ディレクティブ(`_config`と`config`)を使います。同一コード内で`_config`と`config`の両方を使う事はできません。

14.4.1 __config

ディレクティブ「`_config`」はPIC10/12/16 MCU用に使います。PIC18 MCU (PIC18FXXJを除く)にも使えますが、「`config`」を推奨します。以下に構文を示します。

```
__config expr        ;For a single configuration word
```

または

```
__config addr, expr ;For multiple configuration word
```

addr	コンフィグレーション ワードのアドレスです。リテラルでも指定できますが、通常はマクロにより表現します。 **Note:** マクロは、レジスタ番号順(昇順)に列記する必要があります。
expr	指定したコンフィグレーション ビットに設定する値を表現する式です。リテラルでも指定できますが、通常はマクロまたは複数マクロの論理積(AND)により表現します。

マクロはデバイス インクルード ファイル(*.inc)で定義されます。Windowsの場合、このファイルは以下の既定値フォルダに保存されます。

```
C:\Program Files\Microchip\MPLABX\mpasmx
```

ディレクティブの大文字と小文字は区別されないため、_CONFIGと_configのどちらでも使えます。マクロの大文字/小文字は、ヘッダファイル内の記述に合わせる必要があります。

『MPASM Assembler, MPLINK Object Linker, MPLIB Object Librarian User's Guide』(DS30003014)の「_config - Set Processor Configuration Bits」を参照してください。

例 - PIC10/12/16 MCUの場合

```
#include p16f877a.inc
;Set oscillator to HS, watchdog time off, low-voltage prog. off
__CONFIG _HS_OSC & _WDT_OFF & _LVP_OFF
```

例 - PIC18 MCUの場合

```
#include p18f452.inc
;Oscillator switch enabled, RC oscillator with OSC2 as I/O pin.
__CONFIG    _CONFIG1, _OSCS_OFF_1 & _RCIO_OSC_1
;Watch Dog Timer enable, Watch Dog Timer PostScaler count - 1:128
__CONFIG    _CONFIG3, _WDT_ON_3 & _WDTPS_128_3
```

14.4.2 config

ディレクティブ「config」は、PIC18 MCU (PIC18FXXJを含む)用に使います。以下に構文を示します。

```
config setting=value [, setting=value]
```

setting	コンフィグレーション ビットを表現するマクロです。
value	指定したコンフィグレーション ビットに設定する値を表現するマクロです。コンマで区切る事により、一行で複数の設定を定義できます。1つのコンフィグレーション バイトの設定を複数行に分けて定義する事もできます。

マクロはデバイス インクルード ファイル(*.inc)で定義されます。Windowsの場合、このファイルは以下の既定値フォルダに保存されます。

```
C:\Program Files\Microchip\MPLABX\mpasmx
```

ディレクティブの大文字と小文字は区別されないため、_CONFIGと_configのどちらでも使えます。マクロの大文字/小文字は、ヘッダファイル内の記述に合わせる必要があります。

『MPASM Assembler, MPLINK Object Linker, MPLIB Object Librarian User's Guide』(DS30003014)の「config - Set Processor Configuration Bits (PIC18 MCUs)」を参照してください。

例 - PIC18 MCUの場合

```
#include p18f452.inc
;Oscillator switch enabled, RC oscillator with OSC2 as I/O
pin.CONFIG OSCS=ON, OSC=LP
;Watch Dog Timer enable, Watch Dog Timer PostScaler count - 1:128
CONFIG     WDT=ON, WDTPS=128
```

14.5 HI-TECH® PICC™ツールチェーン

htc.hヘッダファイル内で定義されているマクロを使って、PIC10/12/16 MCUのデバイス コンフィグレーション ワードを設定します:

```
_____CONFIG(x);
```

x	指定したコンフィグレーション ビットに設定する値を表現する式です。リテラルでも指定できますが、通常はマクロまたは複数マクロの論理積(AND)により表現します。

マクロはデバイス ヘッダファイル(*.h)で定義されます。Windowsの場合、このファイルは以下の既定値フォルダに保存されます。

```
C:\Program Files\HI-TECH Software\PICC\vx.xx\include
```

vx.xxは、コンパイラのバージョン番号です。

複数のコンフィグレーション ワードアドレスを持つデバイスでは、__CONFIG()を呼び出すたびに次のコンフィグレーション ワードを変更します。

マクロの大文字/小文字は、対応するヘッダファイル内の記述に合わせる必要があります。htc.cに対しては大文字の__CONFIG()を使います(__config()は使えません)。『HI-TECH C for PIC10/12/16 User's Guide』(DS51865)の「Configuration Fuses」を参照してください。

PICCの例

```
#include <htc.h>
__CONFIG(WDTDIS & XT & UNPROTECT); // Program config. word 1
__CONFIG(FCMEN);                   // Program config. word 2
```

14.6 HI-TECH® PICC-18™ツールチェーン

htc.hヘッダファイルで定義されているマクロを使って、PIC18 MCUのデバイス コンフィグレーション ワードを設定します。

```
__CONFIG(n,x);
```

n	コンフィグレーション レジスタ番号です。
x	指定したコンフィグレーション ビットに設定する値を表現する式です。リテラルでも指定できますが、通常はマクロまたは複数マクロの論理積(AND)により表現します。

マクロはデバイス ヘッダファイル(*.h)で定義されます。Windowsの場合、このファイルは以下の既定値フォルダに保存されます。

```
C:\Program Files\HI-TECH Software\PICC\vx.xx\include
```

vx.xxは、コンパイラのバージョン番号です。

マクロの大文字/小文字は、対応するヘッダファイル内の記述に合わせる必要があります。htc.cに対しては大文字の__CONFIG()を使います(__config()は使えません)。『HI-TECH C Tools for the PIC18 MCU Family』の「Configuration Fuses」を参照してください。

PICC-18の例

```
#include <htc.h>
//Oscillator switch enabled, RC oscillator with OSC2 as I/O pin.
__CONFIG(1, LP);
//Watch Dog Timer enable, Watch Dog Timer PostScaler count - 1:128
__CONFIG(2, WDTEN & WDTPS128);
```

14.7 C18ツールチェーン

ディレクティブ「#pragma config」は、アプリケーションで使うデバイスに固有のコンフィグレーション設定(コンフィグレーション ビット)を指定します。

```
#pragma config setting-list
```

setting-list	コンマで区切った「setting-name = value-name」マクロペアです。

マクロはデバイス ヘッダファイル(*.h)で定義されます。Windowsの場合、このファイルは以下の既定値フォルダに保存されます。

```
C:\program files\microchip\mplabc18\vx.xx\.h
```

pragmaの大文字と小文字は区別されないため、#PRAGMA CONFIGと#pragma configのどちらでも使えます。マクロの大文字/小文字は、ヘッダファイル内の記述に合わせる必要があります。

『MPLAB C18 C Compiler User's Guide』(DS50001288)の「Pragms」、「#pragma config」を参照してください。

例

```
#include <p18cxxx.h>
/*Oscillator switch enabled, RC oscillator with OSC2 as I/O
pin.*/#pragma config OSCS = ON, OSC = LP
/*Watch Dog Timer enable, Watch Dog Timer PostScaler count - 1:128*/
#pragma config WDT = ON, WDTPS = 128
```

14.8 ASM30ツールチェーン

デバイス インクルード ファイルで定義されているマクロを使ってコンフィグレーション ビットを設定します。

```
config_reg, value
```

__reg	コンフィグレーション レジスタ名を表現するマクロです。
value	指定したコンフィグレーション ビットに設定する値を表現する式です。リテラルでも指定できますが、通常はマクロまたは複数マクロの論理積(AND)により表現します。

マクロはデバイス インクルード ファイル(*.inc)で定義されます。Windowsの場合、このファイルは以下の既定値フォルダに保存されます。

```
C:\Program Files\Microchip\MPLAB ASM30 Suite\Support\device\inc
```

deviceは、選択した16ビットデバイスの略称です(PIC24H、dsPIC33F)。

マクロの大文字/小文字は、対応するヘッダファイル内の記述に合わせる必要があります。例えば、ヘッダファイルで小文字のconfigが使われている場合、大文字のCONFIGは使えません。

『MPLAB® Assembler, Linker and Utilities for PIC24 MCUs and dsPIC® DSCs User's Guide』(DS50001317)の「Output Sections in Configuration Memory」を参照してください。

例

```
.include "p30fxxxx.inc"
;Clock switching off, Fail-safe clock monitoring off,
; Use External Clock
config_FOSC, CSW_FSCM_OFF & XT_PLL16
;Turn off Watchdog Timer
config_FWDT, WDT_OFF
```

14.9 C30ツールチェーン

デバイス コンフィグレーション ビットの設定には2種類のコンパイラマクロ(PIC24F MCU用とdsPIC30F/dsPIC33F/PIC24H用)を使います。

14.9.1 PIC24Fのコンフィグレーション設定

デバイスヘッダ ファイルで定義されているマクロを使ってコンフィグレーション ビットを設定します。

```
_confign(value);
```

n	コンフィグレーション レジスタ番号です。
value	指定したコンフィグレーション ビットに設定する値を表現する式です。リテラルでも指定できますが、通常はマクロまたは複数マクロの論理積(AND)により表現します。

マクロはデバイス ヘッダファイル(*.h)で定義されます。Windowsの場合、このファイルは以下の既定値フォルダに保存されます。

```
C:\Program Files\Microchip\MPLAB C30\support\PIC24F\h
```

マクロの大文字/小文字は、対応するヘッダファイル内の記述に合わせる必要があります。例えば、ヘッダファイルで大文字の_CONFIG1が使われている場合、小文字の_config1は使えません。

例 - PIC24F MCUの場合

```
#include “p24Fxxxx.h”
//JTAG off, Code Protect off, Write Protect off, COE mode off, WDT off
_CONFIG1( JTAGEN_OFF & GCP_OFF & GWRP_OFF & COE_OFF & FWDTEN_OFF )
//Clock switching/monitor off, Oscillator (RC15) on,
// Oscillator in HS mode, Use primary oscillator (no PLL)
_CONFIG2( FCKSM_CSDCMD & OSCIOFNC_ON & POSCMOD_HS & FNOSC_PRI )
```

14.9.2 dsPIC30F/33F/PIC24Hのコンフィグレーション設定

デバイスヘッダ ファイルで定義されているマクロを使ってコンフィグレーション ビットを設定します。

```
_reg(value);
```

_reg	コンフィグレーション レジスタ名を表現するマクロです。
value	指定したコンフィグレーション ビットに設定する値を表現する式です。リテラルでも指定できますが、通常はマクロまたは複数マクロの論理積(AND)により表現します。

マクロはデバイス ヘッダファイル(*.h)で定義されます。Windowsの場合、このファイルは以下の既定値フォルダに保存されます。

```
C:\Program Files\Microchip\MPLAB C30\support\device\h
```

deviceは、選択した16ビットデバイスの略称です(PIC24H、dsPIC30F、dsPIC33F)。マクロの大文字/小文字は、対応するヘッダファイル内の記述に合わせる必要があります。例えば、ヘッダファイルで大文字の_FOSCが使われている場合、小文字の_foscは使えません。
『MPLAB® C Compiler for PIC24 MCUs and dsPIC® DSCs User’s Guide』(DS50001284)の「Configuration Bits Setup Macros」を参照してください。

例 - dsPIC30F DSC

```
#include “p30fxxxx.h”
//Clock switching and failsafe clock monitoring off,
// Oscillator in HS mode
_FOSC(CSW_FSCM_OFF & HS);
//Watchdog timer off
_FWDT(WDT_OFF);
//Brown-out off, Master clear on
_FBORPOR(PBOR_OFF & MCLR_EN);
```

例 - dsPIC33F/PIC24H

```
#include “p33fxxxx.h”
// Use primary oscillator (no PLL)
_FOSCSEL(FNOSC_PRI);
//Oscillator in HS mode
_FOSC(POSCMD_HS);
//Watchdog timer off
_FWDT(FWDTEN_OFF);
//JTAG off
_FICD(JTAGEN_OFF);
```

14.10 C32ツールチェーン

ディレクティブ「#pragma config」は、アプリケーションで使うデバイスに固有のコンフィグレーション設定(コンフィグレーション ビット)を指定します。

```
# pragma config setting-list
```

setting-list:	コンマで区切った「setting-name = value-name 」マクロペアです。

マクロはデバイス ヘッダファイル(*.h)で定義されます。Windowsの場合、このファイルは以下の既定値フォルダに保存されます。

```
C:\Program Files\Microchip\MPLAB C32\pic32mx\include\proc
```

pragmaの大文字と小文字は区別されないため、#PRAGMA CONFIGと#pragma configのどちらでも使えます。マクロの大文字/小文字は、ヘッダファイル内の記述に合わせる必要があります。

『MPLAB® C Compiler for PIC32 MCUs User’s Guide』(DS50001686)の「Configuration Bit Access」を参照してください。

例

```
#include “p32xxxx.h”
//Enables the Watchdog Timer,
// Sets the Watchdog Postscaler to 1:128
#pragma config FWDTEN = ON, WDTPS = PS128
//Selects the HS Oscillator for the Primary Oscillator
#pragma config POSCMOD = HS
```

15. 用語集

絶対セクション(Absolute Section)
リンカで変更されない固定(絶対)アドレスを持つGCCコンパイラのセクション。

絶対変数/関数(Absolute Variable/Function)
OCGコンパイラの@ address構文を使って絶対アドレスに配置される変数または関数。

アクセスメモリ(Access Memory)
PIC18のみ - PIC18でバンクセレクト レジスタ(BSR)の設定に関わらずアクセスできる特殊なレジスタ。

アクセス エントリポイント(Access Entry Points)
リンク時に定義されていない可能性のある関数に、セグメントの境界を越えて制御を渡すための手段。ブートセグメントとセキュア アプリケーション セグメントを別々にリンクする方法を提供する。

アドレス(Address)
メモリ内の位置を一意に特定する値。

アルファベット文字(Alphabetic Character)
アルファベットの小文字と大文字の総称(a,b,･･･,z,A,B,･･･,Z)。

英数字(Alphanumeric)
アルファベット文字と0～9の10進数(0,1,･･･,9)の数字の総称。

AND条件ブレークポイント(ANDed Breakpoint)
プログラムの実行を停止するために設定するAND条件(ブレークポイント1とブレークポイント2が同時に発生した場合のみプログラム実行を停止する)。AND条件で実行が停止するのは、データメモリのブレークポイントとプログラムメモリのブレークポイントが同時に発生した場合のみ。

匿名構造体(Anonymous Structure)
16ビットCコンパイラ – 無名の構造体。

PIC18 Cコンパイラ - C共用体のメンバーである無名の構造体。匿名構造体のメンバーは、その構造体を包含している共用体のメンバーと同じようにアクセスできる。例えば以下のサンプルコードでは、hiとloは共用体casterに含まれる匿名構造体のメンバーである。

```
union castaway
int intval;
struct {
char lo; //accessible as caster.lo
char hi; //accessible as caster.hi
};
} caster;
```

ANSI
American National Standards Institute(米国規格協会)の略。米国における標準規格の策定と承認を行う団体。

アプリケーション(Application)
PIC® MCUで制御されるソフトウェアとハードウェアを組み合わせたもの。

アーカイブ/ライブラリ(Archive/Library)
アーカイブ/ライブラリは、再配置可能なオブジェクト モジュールの集まり。複数のソースファイルをオブジェクト ファイルにアセンブルした後、アーカイバ/ライブラリアンを使ってこれらオブジェクト ファイルを1つのアーカイブ/ライブラリ ファイルにまとめると生成される。アーカイブ/ライブラリをオブジェクト モジュールや他のアーカイブ/ライブラリとリンクすると、実行コードが生成される。

ASCII
American Standard Code for Information Interchangeの略。7桁の2進数で1つの文字を表現する文字セットエンコード方式。大文字、小文字、数字、記号、制御文字等を含む。

アセンブリ/アセンブラ(Assembly/Assembler)
アセンブリとは、2進数のマシンコードをシンボル表現で記述したプログラミング言語。アセンブラとは、アセンブリ言語のソースコードをマシンコードに変換する言語ツール。

割り当てセクション(Assigned Section)
リンカのコマンドファイルで特定のターゲットメモリ ブロックに割り当てられたGCCコンパイラのセクション。

非同期(Asynchronously)
複数のイベントが同時には発生しない事。一般に、プロセッサ実行中の任意の時点で発生する割り込みに言及する際に使う。

非同期スティミュラス(Asynchronous Stimulus)
シミュレータ デバイスへの外部入力をシミュレートするために生成されるデータ。

属性(Attribute)
GCCのC言語プログラムの変数または関数の特徴を表す情報で、マシン固有の特性を記述する目的で使う。

属性(セクション属性) (Attribute, Section)
「executable」、「readonly」、「data」等、セクションの特徴を表す情報。アセンブラの.sectionディレクティブでフラグとして指定できる。

2進数(Binary)
0と1の数字を使う、2を底とした記数法。一番右の桁が1の位、次の桁が2の位、その次の桁が2_2 = 4の位を表す。

ブックマーク(Bookmark)
ファイル内の特定の行に簡単な操作でアクセスできるようにする機能。

[Editor]ツールバーの[Toggle Bookmarks]を選択してブックマークを追加または削除する。このツールバーの他のアイコンをクリックすると、次または前のブックマークに移動する。

C/C++
C言語は、簡潔な表現、現代的な制御フローとデータ構造、豊富に用意された演算子等を特長とする汎用プログラミング言語。C++とは、C言語のオブジェクト指向バージョン。

校正メモリ(Calibration Memory)
PIC MCUの内蔵RCオシレータやその他の周辺モジュールの校正値を格納するための特殊機能レジスタ。

中央演算処理装置(Central Processing Unit)
デバイス内で、実行する正しい命令をフェッチし、デコードして実行する装置。必要に応じて、算術論理演算装置(ALU)と組み合わせて命令実行を完了する。プログラムメモリのアドレスバス、データメモリのアドレスバス、スタックへのアクセスを制御する。

クリーン(Clean)
クリーンする事により、アクティブなプロジェクトのオブジェクト ファイル、HEXファイル、デバッグファイル等、全ての中間ファイルが削除される。これらのファイルは、プロジェクトのビルド時に他のファイルから再構築される。

COFF
Common Object File Formatの略。このフォーマットのオブジェクト ファイルは、マシンコードの他、デバッグ等に関する情報を含む。

コマンドライン インターフェイス(Command Line Interface)
プログラムとユーザのやり取りをテキストの入出力だけで行う方法。

コンパイルド スタック(Compiled Stack)
コンパイラが管理するメモリの領域で、この領域内で変数に静的に空間を割り当てる。ターゲット デバイス上にソフトウェア スタックまたはハードウェア スタックのメカニズムを効率的に実装できない場合、コンパイルド スタックがソフトウェア スタックまたはハードウェア スタックに置き換わる。

コンパイラ(Compiler)
高級言語で記述されたソースファイルをマシンコードに変換するプログラム。

条件付きアセンブリ(Conditional Assembly)
アセンブリ言語で、ある特定の式のアセンブル時の値に基づいて含まれたり除外されたりするコード。

条件付きコンパイル(Conditional Compilation)
プログラムの一部を、プリプロセッサ ディレクティブで指定した特定の定数式が真の場合のみコンパイルする事。

コンフィグレーション ビット(Configuration Bit)
PIC MCUとdsPIC DSCの動作モードを設定するために書き込む専用ビット。コンフィグレーション ビットは事前プログラミングされている場合とされていない場合がある。

制御ディレクティブ(Control Directive)
アセンブリ言語コード内で使うディレクティブで、指定した式のアセンブル時の値に基づいてコードを含めるか除外するかを決定する。

CPU
「中央演算処理装置」参照

相互参照ファイル(Cross Reference File)
シンボルテーブルとそのシンボルを参照するファイルリストを参照するファイル。シンボルが定義されている場合、リストの最初のファイルがシンボル定義の位置となる。残りのファイルはシンボルへの参照を含む。

データ ディレクティブ(Data Directive)
アセンブラが行うプログラムメモリまたはデータメモリの割り当てを制御するディレクティブ。データ項目をシンボル(意味のある名前)で参照する手段として使う。

データメモリ(Data Memory)
Microchip社のMCUとDSCでは、データメモリ(RAM)は汎用レジスタ(GPR)と特殊機能レジスタ(SFR)で構成される。EEPROMデータメモリを内蔵したデバイスもある。

データ監視および制御インターフェイス(DMCI: Data Monitor and Control Interface)
MPLAB X IDE内のツール。このインターフェイスは、プロジェクト内のアプリケーション変数の動的な入力制御を提供する。4つの動的に割り当て可能なグラフィック ウィンドウを使って、アプリケーションが生成するデータをグラフィカルに表示できる。

デバッグ/デバッガ(Debug/Debugger)
「ICE/ICD」参照

デバッグ情報(Debugging Information)
コンパイラとアセンブラでこのオプションを選択すると、アプリケーション コードのデバッグに使える各種レベルの情報を出力できる。デバッグ オプションの選択の詳細はコンパイラまたはアセンブラのマニュアルを参照。

推奨しない機能(Deprecated Feature)
後方互換性確保のためにサポートしているだけで現在は使っておらず、いずれ廃止になる事が決まっている機能。

デバイス プログラマ(Device Programmer)
マイクロ コントローラ等、電気的に書き込み可能な半導体デバイスをプログラミングするためのツール。

デジタルシグナル コントローラ(Digital Signal Controller)
デジタル信号処理機能をサポートしたMCU。Microchip社のdsPIC DSC等。

デジタル信号処理/デジタルシグナル プロセッサ(Digital Signal Processing/Digital Signal Processor)
デジタル信号処理(DSP)とは、デジタル信号をコンピュータで処理する事。通常は、アナログ信号(音声または画像)をデジタル形式に変換(サンプリング)して処理する事をいう。デジタルシグナル プロセッサとは、信号処理用に設計されたマイクロプロセッサの事。

ディレクティブ(Directive)
言語ツールの動作を制御するためにソースコードに記述する命令文。

ダウンロード(Download)
ホストから別のデバイス(例: エミュレータ、プログラマ、ターゲットボード)にデータを送信する事。

DWARF
Debug With Arbitrary Record Formatの略。ELFファイルのデバッグ情報フォーマット。

EEPROM
Electrically Erasable Programmable Read Only Memoryの略。電気的に消去可能なタイプのPROM。データの書き込みと消去をバイト単位で行う。EEPROMは電源をOFFにしても内容を保持する。

ELF
Executable and Linking Formatの略。この形式のオブジェクト ファイルはマシンコードを含む。デバッグその他の情報はDWARFで指定する。ELF/DWARFの方がCOFFよりも最適化したコードのデバッグに適している。

エミュレーション/エミュレータ(Emulation/Emulator)
「ICE/ICD」参照

エンディアン(Endianness)
マルチバイト オブジェクトにおけるバイトの並び順。

環境(Environment)
MPLAB PM3 - デバイスのプログラミングに関する設定ファイルを保存したフォルダ。このフォルダをSD/MMCカードに転送できる。

エピローグ(Epilogue)
コンパイラで生成したコードのうち、スタック領域の割り当て解除、レジスタの復帰、ランタイムモデルで指定したその他のマシン固有の要件を実行するコード部分。関数のユーザコードの後、関数リターンの直前にエピローグを実行する。

EPROM
Erasable Programmable Read Only Memoryの略。再書き込みが行えるタイプのROMで、消去は紫外線照射で行うものが主流。

エラー/エラーファイル(Error/Error File)
プログラムの処理を継続できない問題が発生するとエラーとして報告される。可能な場合、エラーは問題が発生したソースファイル名と行番号を特定する。エラーファイルは、言語ツールから出力されたエラーメッセージと診断結果を保存する。

イベント(Event)
アドレス、データ、パスカウント、外部入力、サイクルタイプ (フェッチ、R/W)、タイムスタンプ等、バスサイクルを記述したもの。トリガ、ブレークポイント、割り込みを記述するために使う。

実行可能コード(Executable Code)
読み込んで実行できる形式のソフトウェア。

エクスポート(Export)
MPLAB IDE/MPLAB X IDEから標準フォーマットでデータを外部に出力する事。

式(Expression)
算術演算子または論理演算子で区切った定数または記号の組み合わせ。

拡張マイクロ コントローラ モード(Extended Microcontroller Mode)
拡張マイクロ コントローラ モードでは、内蔵プログラムメモリと外部メモリの両方が利用できる。プログラムメモリのアドレスがPIC18の内部メモリ空間より大きい場合、自動的に外部メモリの実行に切り換わる。

拡張モード(Extended Mode) (PIC18 MCU)
コンパイラの動作モードの1つ。拡張命令(ADDFSR、ADDULNK、CALLW、MOVSF、MOVSS、PUSHL、SUBFSR、SUBULNK)とリテラル オフセットによるインデックス アドレス指定を利用できる。

外部ラベル(External Label)
外部リンケージを持つラベル。

外部リンケージ(External Linkage)
関数または変数が、それを定義したモジュールの外部から参照できる場合、外部リンケージを持つという。

外部シンボル(External Symbol)
外部リンケージを持つ識別子のシンボル。参照の場合と定義の場合がある。

外部シンボル解決(External Symbol Resolution)
リンカが全ての入力モジュールの外部シンボル定義を1つにまとめ、全ての外部シンボル参照を解決しようとするプロセス。外部シンボル参照に対応する定義が存在しない場合、リンカエラーとなる。

外部入力ライン(External Input Line)
外部信号に基づいてイベントを設定するための外部入力信号ロジックプローブ ライン(TRIGIN)。

外部RAM (External RAM)
デバイス外部にある、読み書き可能なメモリ。

致命的エラー(Fatal Error)
コンパイルがただちに停止するようなエラー。エラーの発生後はメッセージも出力されない。

ファイルレジスタ(File Register)
汎用レジスタ(GPR)と特殊機能レジスタ(SFR)からなる内蔵のデータメモリ。

フィルタ(Filter)
トレース ディスプレイまたはデータファイルにどのデータを含めるか/除外するかを選択するもの。

フィックスアップ(Fixup)
リンカによる再配置後にオブジェクト ファイルのシンボル参照を絶対アドレスに置き換える処理。

フラッシュ(Flash)
データの書き込みと消去をバイト単位ではなくブロック単位で行えるタイプのEEPROM。

FNOP
Forced No Operationの略。Forced NOPサイクルは、2サイクル命令の2サイクル目で発生する。PICマイクロ コントローラのアーキテクチャはパイプライン構造となっており、現在の命令を実行中に物理アドレス空間の次の命令をプリフェッチする。しかし、現在の命令でプログラム カウンタが変化した場合、プリフェッチした命令は明示的に無視され、Forced NOPサイクルが発生する。

フレームポインタ(Frame Pointer)
スタックベースの引数とスタックベースのローカル変数の境界となるスタック番地を指し示すポインタ。ここを基準にすると、現在の関数のローカル変数やその他の値に容易にアクセスできる。

フリー スタンディング(Free-Standing)
複素数型を使っておらず、ライブラリ(ANSI C89規格第7節)で規定する機能の使用が標準ヘッダ(`<float.h>`、`<iso646.h>`、`<limits.h>`、`<stdarg.h>`、`<stdbool.h>`、`<stddef.h>`、`<stdint.h>`)の内容のみに限定されている厳密な規格合致プログラムを受理する処理系。
<stdint.h>.

GPR
General Purpose Register(汎用レジスタ)の略。デバイスのデータメモリ(RAM)のうち、汎用目的に使える部分。

Halt
プログラム実行を停止する事。Haltを実行する事は、ブレークポイントで停止する事と同じ。

ヒープ(Heap)
動的メモリ割り当てに使うメモリ空間。メモリブロックの割り当てと解放は実行時に任意の順序で行う。

HEXコード/HEXファイル(Hex Code/Hex File)
HEXコードは、実行可能な命令を16進数形式のコードで保存したもの。HEXファイルは、HEXコードを格納したファイル。

16進数(Hexadecimal)
0～9の数字とA～F(またはa～f)のアルファベットを使った、16を底とした記数法。16進数のA～Fは、10進数の10～15を表す。一番右の桁が1の位、次の桁が16の位、その次の桁が16^2 = 256の位を表す。

高級言語(High Level Language)
プログラムを記述するための言語で、プロセッサから見てアセンブリよりも遠い位置関係にあるもの。

ICE/ICD
インサーキット エミュレータ/インサーキット デバッガの略。ターゲット デバイスのデバッグとプログラミングを行うためのハードウェア ツール。エミュレータは、デバッガよりも多くの機能(トレース等)を備える。

インサーキット エミュレーション/インサーキット デバッグとは、インサーキット エミュレータまたはデバッガを使った作業の事を指す。

-ICE/-ICD: インサーキット エミュレーション/デバッグ用の回路を内蔵したデバイス(MCUまたはDSC)。このデバイスは必ずヘッダ基板にマウントし、インサーキット エミュレータまたはデバッガによるデバッグ用に使う。

ICSP
In-Circuit Serial Programmingの略。Microchip社製の組み込みデバイスを、シリアル通信を利用して最小限のデバイスピンでプログラミングする方法。

IDE
Integrated Development Environmentの略。MPLAB IDE/MPLAB X IDEのIDE。

識別子(Identifier)
関数または変数の名前。

IEEE
Institute of Electrical and Electronics Engineersの略。

インポート(Import)
HEXファイル等の外部ソースからMPLAB IDE/MPLAB X IDEにデータを取り込む事。

初期化済みデータ(Initialized Data)
初期値を指定して定義されたデータ。Cでは、

```
int myVar=5;
```

として定義した変数は初期化済みデータセクションに格納する。

命令セット(Instruction Set)
特定のプロセッサが理解できるマシン語命令の集合。

命令(Instruction)
CPUに対して特定の演算を実行するように指示するビット列。演算の対象となるデータを含める事もできる。

内部リンケージ(Internal Linkage)
関数または変数が、それを定義したモジュールの外部から参照できない場合、内部リンケージを持つという。

国際標準化機構(International Organization for Standardization)
コンピューティングや通信を始めとする、多くのテクノロジとビジネス関連の標準規格の策定を行っている団体。一般的にISOと呼ぶ。

割り込み(Interrpt)
CPUに対する信号の一種。この信号が発生すると、現在動作中のアプリケーションの実行を一時停止し、制御を割り込みサービスルーチン(ISR)に渡してイベントを処理する。ISRの実行が完了すると、通常のアプリケーションの実行を再開する。

割り込みハンドラ(Interrupt Handler)
割り込み発生時に専用のコードを実行するルーチン。

割り込みサービス要求(IRQ: Interrupt Service Request)
プロセッサの通常の命令実行を一時的に停止し、割り込みハンドラルーチンの実行開始を要求するイベント。プロセッサによっては複数の割り込み要求イベントを持ち、優先度の異なる割り込みを処理できるものもある。

割り込みサービスルーチン(ISR: Interrupt Service Routine)
言語ツールの場合: 割り込みを処理する関数。

MPLAB IDE/MPLAB X IDEの場合: 割り込みが発生すると実行されるユーザ作成コード。通常、発生した割り込みの種類によってプログラムメモリ内の異なる位置のコードを実行する。

割り込みベクタ(Interrupt Vector)
割り込みサービスルーチンまたは割り込みハンドラのアドレス。

左辺値(L-value)
検査または変更が可能なオブジェクトを示す式。左辺値は代入演算子の左側で使う。

レイテンシ(Latency)
イベントが発生してからその応答までの時間の長さ。

ライブラリ/ライブラリアン(Library/Librarian)
「アーカイブ/ライブラリ」参照

リンカ(Linker)
オブジェクト ファイルとライブラリを結合し、モジュール間の参照を解決して実行可能コードを生成する言語ツール。

リンカスクリプト ファイル(Linker Script File)
リンカのコマンドファイル。リンカのオプションを定義し、ターゲット プラットフォームで利用可能なメモリを記述する。

リスティング ディレクティブ(Listing Directive)
アセンブラのリスティング ファイルのフォーマットを制御するディレクティブ。タイトルや改ページ指示等、リスティング ファイルに関する各種の設定を行う。

リスティング(Listing File)
ソースファイルにある各Cソース ステートメント、アセンブリ命令、アセンブラ ディレクティブ、マクロに対して生成されたマシンコードを記述したASCIIテキストファイル。

リトル エンディアン(Little Endian)
マルチバイト データで最下位バイト(LSB)を最下位アドレスに格納するデータ並び順方式。

ローカルラベル(Local Label)
マクロ内でLOCALディレクティブを使って定義されたラベル。ローカルラベルは、マクロの同一インスタンス内でのみ有効。すなわち、LOCALとして宣言されたシンボルとラベルには、ENDMマクロ以降はアクセスできない。

ロジックプローブ(Logic Probes)
Microchip社製エミュレータには、最大14のロジックプローブを接続できるものがある。ロジックプローブは、外部トレース入力、トリガ出力信号、+5 V、共通グランドを提供する。

ループバック テストボード(Loop-Back Test Board)
MPLAB REAL ICEインサーキット エミュレータの動作をテストするために用いる。

LVDS
Low Voltage Differential Signalingの略。銅線を使って低ノイズ、低消費電力、低振幅でデータを高速(数Gbps)で伝送する方法。

標準のI/Oシグナリングでは、データストレージは実際の電圧レベルに依存する。電圧レベルは信号線の長さに影響を受ける(信号線が長いと抵抗が増え電圧が下がる)。これに対しLVDSでは、電圧レベルでなく差動入力の電位差が正か負かでのみデータの意味を区別する。従って、長い信号線でもクリアで安定したデータストリームを維持した伝送が可能。

出典: http://www.webopedia.com/TERM/L/LVDS.html

マシンコード(Machine Code)
コンピュータ プログラムをプロセッサが実際に読み出して解釈できる形式で表現したもの。2進数のマシンコードで記述されたプログラムは、マシン命令のシーケンス(命令間にデータを挟む事もある)からなる。ある特定のプロセッサで使える全ての命令の集合を「命令セット」という。

マシン語(Machine Language)
あるCPUが翻訳を必要とせず実行できる命令の集合。

マクロ(Macro)
マクロ命令。一連の命令シーケンスを短い名前で表現した命令。

マクロ ディレクティブ(Macro Directive)
マクロ定義の中で実行とデータ割り当てを制御するディレクティブ。

makeファイル(Makefile)
プロジェクトのMakeに関する指示をファイルにエクスポートしたもの。このファイルは、MPLAB IDE/MPLAB X IDE以外の環境でmakeコマンドを実行してプロジェクトをビルドする際に使う。

Make Project
アプリケーションを再ビルドするコマンド。前回の完全なコンパイル後に変更されたソースファイルのみを再コンパイルする。

MCU
Microcontroller Unitの略。マイクロコントローラの事。「µC」と表記する事もある。

メモリモデル(Memory Model)
Cコンパイラの場合、アプリケーションで利用可能なメモリを表現したもの。PIC18Cコンパイラの場合、プログラムメモリを指し示すポインタのサイズに関する規定を記述したもの。

メッセージ(Message)
言語ツールの動作に問題が発生した事を知らせる文字列。メッセージが表示されても処理は停止しない。

マイクロコントローラ(Microcontroller)
CPU、RAM、プログラムメモリ、I/Oポート、タイマ等、多くの機能を統合したチップ。

マイクロコントローラ モード(Microcontroller Mode)
PIC18MCUで設定可能なプログラムメモリ構成の1つ。マイクロコントローラ モードでは、内部実行のみを許可する。つまり、マイクロコントローラ モードでは内蔵プログラムメモリしか使えない。

マイクロプロセッサ モード(Microprocessor Mode)
PIC18マイクロコントローラで設定可能なプログラムメモリ構成の1つ。マイクロプロセッサ モードでは、内蔵プログラムメモリは使わない。プログラムメモリ全体を外部にマッピングする。

ニーモニック(Mnemonic)
マシンコードと1対1で対応したテキスト命令。オペコードとも呼ぶ。

モジュール(Module)
プリプロセッサ ディレクティブ実行後の前処理済みのソースファイル出力。翻訳単位とも呼ぶ。

MPASM™アセンブラ(MPASM Assembler)
PIC MCU、KeeLoq®デバイス、Microchip社のメモリデバイスに対応したMicrochip社の再配置可能なマクロアセンブラ。

MPLAB (言語ツール名) for (デバイス名) (MPLAB Language Tool for Device)
特定のデバイスに対応したMicrochip社のCコンパイラ、アセンブラ、リンカ。言語ツールは、アプリケーションで使うデバイスに対応したものを選択する必要がある。例えばPIC18 MCU用のCコードを作成する場合、「MPLAB C Compiler for PIC18 MCU」を使う。

MPLAB ICD
MPLAB IDE/MPLAB X IDEと組み合わせて使うMicrochip社のインサーキット デバッガ。「ICE/ICD」参照。

MPLAB IDE/MPLAB X IDE
Microchip社の統合開発環境。エディタ、プロジェクト マネージャ、シミュレータが付属する。

MPLAB PM3
Microchip社提供のデバイス プログラマ。PIC18マイクロ コントローラとdsPICデジタルシグナル コントローラのプログラムに対応。MPLAB IDE/MPLAB X IDEと一緒に使う事も、単体で使う事も可能。PRO MATE IIの後継製品。

MPLAB REAL ICE™インサーキット エミュレータ
MPLAB IDE/MPLAB X IDEと組み合わせて使うMicrochip社の次世代インサーキット エミュレータ。「ICE/ICD」参照。

MPLAB SIM
MPLAB IDE/MPLAB X IDEと組み合わせて使うMicrochip社のシミュレータで、PIC MCUとdsPIC DSCに対応する。

MPLAB Starter Kit for (デバイス名) (MPLAB Starter Kit for Device)
特定のデバイスでの作業を開始する上で必要となるものを全てセットにしたMicrochip社のスタータキット。書き込み済みアプリケーションの動作を確認し、一部を変更してカスタム アプリケーションとしてデバッグとプログラミングを行える。

MPLIB™オブジェクト ライブラリアン(MPLIB Object Librarian)
MPLAB IDE/MPLAB X IDEと組み合わせて使うMicrochip社のライブラリアン。MPLIBライブラリアンは、MPASMアセンブラ(mpasmまたはmpasmwin v2.0)またはMPLAB C18 Cコンパイラで作成したCOFFオブジェクト モジュールに使うオブジェクト ライブラリアン。

MPLINK™オブジェクト リンカ(MPLINK Object Linker)
Microchip社のMPASMアセンブラとC18 Cコンパイラに対応したオブジェクト リンカ。Microchip社のMPLIBライブラリアンとの併用も可能。MPLAB IDE/MPLAB X IDEに統合して使うように設計されているが、MPLAB IDE/MPLAB X IDE以外の環境でも使える。

MRU
Most Recently Usedの略。最近使ったファイルとウィンドウの事。MPLAB IDE/MPLAB X IDEのメインメニューで選択できる。

ネイティブ データサイズ(Native Data Size)
ネイティブ トレースの場合、[Watches]ウィンドウで使う変数のサイズは選択したデバイスのデータメモリと同じサイズ(PIC18の場合は同じバイトサイズ、16ビットデバイスの場合は同じワードサイズ)である必要がある。

入れ子の深さ(Nesting Depth)
マクロに他のマクロを入れ込める階層の数。

ノード(Node)
MPLAB IDE/MPLAB X IDEのプロジェクトを構成するコンポーネント

非拡張モード(Non-Extended Mode) (PIC18 MCU)
コンパイラの動作モードの1つ。拡張命令もリテラル オフセットによるインデックス アドレス指定も使わない。

非リアルタイム(Non Real Time)
ブレークポイントで停止中、またはシングルステップ実行中のプロセッサ、あるいはシミュレータ モードで動作中のMPLAB IDE/MPLAB X IDEを指す。

不揮発性ストレージ(Non-Volatile Storage)
電源をOFFにしても内容が失われないストレージ デバイス。

NOP
No Operationの略。実行してもプログラム カウンタが進むだけで何も動作を行わない命令。

オブジェクト コード/オブジェクト ファイル(Object Code/Object File)
オブジェクト コードとは、アセンブラまたはコンパイラで生成されるマシンコードの事。オブジェクト ファイルとは、マシンコードを格納したファイル。デバッグ情報を含む事もある。そのまま実行できるものと、他のオブジェクト ファイル(例: ライブラリ)とリンクしてから完全な実行プログラムを生成する再配置可能形式のものがある。

オブジェクト ファイル ディレクティブ(Object File Directive)
オブジェクト ファイル作成時にのみ使うディレクティブ。

8進数(Octal)
0～7の数字のみを使う、8を底とした記数法。一番右の桁が1の位、次の桁が8の位、その次の桁が8^2 = 64の位を表す。

オフチップメモリ(Off-Chip Memory)
PIC18で選択できるメモリオプション。ターゲットボードのメモリを使うか、または全てのプログラムメモリをエミュレータから供給する。[Options] > [Development Mode]の順にクリックして[Memory]タブでオフチップメモリを選択する。

オペコード(Opcode)
Operational Codeの略。「ニーモニック」参照。

演算子(Operators)
定義可能な式を構成する際に使う「+」や「-」等の記号。各演算子に割り当てられた優先順位に基づいて式を評価する。

OTP (One-Time-Programmable)
One Time Programmableの略。パッケージに窓のないEPROMデバイス。EPROMを消去するには紫外線照射が必要なため、パッケージに窓のあるデバイスしか消去できない。

パスカウンタ(Pass Counter)
イベント(特定のアドレスの命令を実行する等)が発生するたびに値をデクリメントするカウンタ。パスカウンタの値がゼロになると、イベントの条件を満たす。パスカウンタはブレークロジック、トレースロジック、複合トリガダイアログの任意のシーケンシャル イベントに割り当てられる。

PC
パーソナル コンピュータまたはプログラム カウンタの略。

ホストPC (PC Host)
パーソナル コンピュータ。

永続データ(Persistent Data)
クリアも初期化もされないデータ。デバイスをリセットしてもアプリケーションがデータを保持できるようにするために使う。

ファントムバイト(Phantom Byte)
dsPICアーキテクチャで、24ビット命令ワードを32ビット命令ワードと見なして扱う場合に使う未実装バイト。dsPICのHEXファイルに見られる。

PIC MCU
Microchip社の全てのMCUファミリの総称。

PICkit 2/3
Microchip社の開発用デバイス プログラマで、Debug Expressによるデバッグ機能を備える。サポートしているデバイスの種類は、各ツールのReadmeファイルを参照。

プラグイン(Plug-ins)
MPLAB IDE/MPLAB X IDEでは、標準コンポーネントにプラグイン モジュールを追加する事で、各種ソフトウェア/ハードウェア ツールに対応する。一部のプラグインツールは、[Tools]メニューから利用できる。

ポッド(Pod)
インサーキット エミュレータまたはデバッガの筐体。丸型の場合パック(Puck)と呼ぶ事もある。あるいはプローブ(Probe)とも呼ぶが、「論理プローブ」と混同せぬよう注意が必要。

パワーオン リセット エミュレーション(Power-on-Reset Emulation)
データRAM領域にランダムな値を書き込んで、初回電源投入時のRAMの非初期化値をシミュレートするソフトウェア無作為化処理。

プラグマ(Pragma)
特定のコンパイラにとって意味を持つディレクティブ。一般に、実装で定義した情報をコンパイラに伝達するために使う。

優先順位(Precedence)
式の評価順を定義した規則。

量産プログラマ(Production Programmer)
デバイスを高速にプログラミングできるようにリソースを強化したプログラマ。各種電圧レベルでのプログラミングに対応し、プログラミング仕様に完全に準拠している。
量産環境では応用回路が組み立てラインにとどまる時間をなるべく短くする必要があるため、デバイスへの書き込み時間の短縮が特に重要である。

プロファイル(Profile)
MPLAB SIMシミュレータにおいて、実行したスティミュラスをレジスタ別に一覧表示したもの。

プログラム カウンタ(Program Counter)
現在実行中の命令のアドレスを格納した場所。

プログラム カウンタユニット(Program Counter Unit)
16ビットアセンブラ - プログラムメモリのレイアウトを概念的に表現したもの。プログラム カウンタは1命令ワードで2つインクリメントする。実行可能セクションでは、2プログラム カウンタユニットは3バイトに相当する。読み出し専用セクションでは、2プログラム カウンタユニットは2バイトに相当する。

プログラムメモリ(Program Memory)
MPLAB IDE/MPLAB X IDEの場合: デバイス内で命令が保存されるメモリ空間。また、エミュレータまたはシミュレータにダウンロードしたターゲット アプリケーションのファームウェアを格納するメモリ空間もプログラムメモリと呼ぶ。

16ビットアセンブラ/コンパイラの場合: デバイス内で命令が保存されるメモリ空間。

プロジェクト(Project)
アプリケーションのビルドに必要なファイル(例: ソースコード、リンカスクリプト ファイル)一式と、各種ビルドツールやビルドオプションとの関連付けをまとめたもの。

プロローグ(Prologue)
コンパイラで生成したコードのうち、スタック領域の割り当て、レジスタの退避、ランタイムモデルで指定したその他のマシン固有の要件を実行するコード部分。プロローグは、関数のユーザコードの前に実行する。

プロトタイプ システム(Prototype System)
ユーザのターゲット アプリケーションまたはターゲットボードの事。

Psect
GCCのセクションに相当するOCGの用語。プログラム セクション(program section)の略語。リンカが1つのまとまりとして処理するコードまたはデータのブロック。

PWM信号(PWM Signal)
パルス幅変調(Pulse Width Modulation)信号。一部のPIC MCUは周辺モジュールとしてPWMを内蔵している。

修飾子(Qualifier)
パスカウンタで使ったり、複合トリガにおける次の動作前のイベントとして使ったりするアドレスまたはアドレスレンジ。

基数(Radix)
アドレスを指定する際の記数法(16進法、10進法)の底。

RAM
Random Access Memoryの略。データメモリ。任意の順にメモリ内の情報にアクセスできる。

生データ(Raw Data)
あるセクションに関連付けられたコードまたはデータを2進数で表現したもの。

読み出し専用メモリ(Read Only Memory)
恒久的に保存されているデータへの高速アクセスが可能なメモリ ハードウェア。ただし、データの追加や変更は不可。

リアルタイム(Real Time)
インサーキット エミュレータまたはデバッガがHalt状態から解放されると、プロセッサの実行はリアルタイム モードとなり、通常のチップと同じ挙動をする。リアルタイム モードでは、エミュレータのリアルタイム トレースバッファが有効になり、選択した全てのサイクルを常時キャプチャする。また、全てのブレークロジックが有効になる。インサーキット エミュレータまたはデバッガでは、有効なブレークポイントで停止するか、またはユーザが実行を停止するまでプロセッサはリアルタイムで動作する。

シミュレータでは、ホストCPUでシミュレート可能な最大速度でマイクロ コントローラの命令を実行する事をリアルタイムと呼ぶ。

再帰呼び出し(Recursive Calls)
直接または間接的に自分自身を呼び出す関数。

再帰(Recursion)
定義した関数またはマクロがそれ自身を呼び出す事。再帰マクロを作成する際は、再帰から抜けずに無限ループとなりやすいため注意が必要。

再入可能(Re-entrant)
1つの関数を複数呼び出して同時に実行できる事。直接または間接再帰、あるいは割り込み処理中の実行によって起こる事がある。

緩和(Relaxation)
ある命令を、機能が同じでよりサイズの小さい命令に変換する事。コードサイズを抑えるために便利である。最新のMPLAB XC16には、CALL命令をRCALL命令に緩和する機能がある。この変換は、現在の命令から+/- 32k命令ワード以内にあるシンボルを呼び出す場合に行われる。

再配置可能(Relocatable)
アドレスがメモリの固定番地に割り当てられていないオブジェクト。

再配置可能セクション(Relocatable Section)
16ビットアセンブラ - アドレスが固定されていない(絶対アドレスでない)セクション。再配置可能セクションには、再配置と呼ばれるプロセスでリンカがアドレスを割り当てる。

再配置(Relocation)
リンカが絶対アドレスを再配置可能セクションに割り当てる事。再配置可能セクション内の全てのシンボルを新しいアドレスに更新する。

ROM
Read Only Memoryの略。プログラムメモリ。メモリの内容を変更できない。

Run
エミュレータをHaltから解放するコマンド。エミュレータはアプリケーション コードを実行し、I/Oに対してリアルタイムに変更、応答を行う。

ランタイムモデル(Run-time Model)
ターゲット アーキテクチャのリソースの使用を記述したもの。

ランタイム ウォッチRun-time Watch
アプリケーションの実行につれて変数の値がリアルタイムに変化する[Watch]ウィンドウ。ランタイム ウォッチの設定方法は各ツールの関連文書を参照。ランタイム ウォッチをサポートしていないツールもある。

シナリオ(Scenario)
MPLAB SIMシミュレータでスティミュラス制御を具体的に設定したもの。

項目
OCGのpsectに相当するGCCの用語。リンカが1つのまとまりとして処理するコードまたはデータのブロック。

セクション属性(Section Attribute)
GCCのセクションの特徴を表す情報(例: accessセクション)。

シーケンス ブレークポイント(Sequenced Breakpoint)
シーケンスで発生するブレークポイント。ブレークポイントのシーケンス実行順はボトムアップです。つまり、シーケンスの最後のブレークポイントが最初に発生します。

SQTP (Serialized Quick Turn Programming)
デバイス プログラマでマイクロ コントローラをプログラミングする際に、各デバイスに異なるシリアル番号を書き込めるようにする機能。この番号はエントリコード、パスワード、ID番号として使える。

シェル(Shell)
MPASMアセンブラにおいて、マクロアセンブラへの入力を行うためのプロンプト インターフェイス。MPASMアセンブラにはDOS用シェルとWindows用シェルの2種類がある。

シミュレータ(Simulator)
デバイスの動作をモデル化するソフトウェア プログラム。

シングルステップ(Single Step)
コードを1命令ずつ実行するコマンド。1命令を実行するたびに、MPLAB IDE/MPLAB X IDEがレジスタ ウィンドウ、ウォッチ変数、ステータスの表示を更新するため、命令実行を解析してデバッグできる。Cコンパイラのソースコードもシングルステップ実行できるが、その場合MPLAB IDE/MPLAB X IDEは1命令ずつ実行するのではなく、高級言語のCで記述されたコードの1行から生成される全てのアセンブリレベル命令をシングルステップで実行する。

スキュー(Skew)
命令実行に対応する情報は、異なる複数のタイミングでプロセッサバスに表れる。例えば、実行されるオペコードは直前の命令の実行時にフェッチとしてバスに表れる。ソースデータのアドレスと値、およびデスティネーション データのアドレスは、オペコードが実際に実行される時にバスに表れる。デスティネーション データの値は次の命令の実行時にバスに表れる。トレースバッファは、1インスタンスでバス上に存在する情報をキャプチャする。従って、トレースバッファの1エントリには3つの命令の実行情報が含まれる。1つの命令実行で、ある情報から次の情報までにキャプチャされるサイクル数をスキューと呼ぶ。

スキッド(Skid)
ハードウェア ブレークポイントを使ってプロセッサを停止する場合、ブレークポイント以降の命令を実行してプロセッサが停止する事がある。ブレークポイントの後に実行する命令の数をスキッドと呼ぶ。

ソースコード(Source Code)
人間が記述したコンピュータ プログラム。プログラミング言語で記述されたソースコードは、マシンコードに変換して実行するか、またはインタプリタで実行される。

ソースファイル(Source File)
ソースコードを記述したASCIIテキストファイル。

特殊機能レジスタ(SFR)
I/Oプロセッサ機能、I/Oステータス、タイマ等の各種モードや周辺モジュールを制御するレジスタ専用に使うデータメモリ(RAM)領域。

SQTP
「Serialized Quick Turn Programming」参照

ハードウェア スタック(Stack, Hardware)
PIC MCUで関数を呼び出す時に戻りアドレスを格納する場所。

ソフトウェア スタック(Stack, Software)
アプリケーションが戻りアドレス、関数パラメータ、ローカル変数を保存するのに使うメモリ。このメモリはプログラムでの命令の実行時に動的に割り当てられる。これによって、再入可能な関数の呼び出しが可能になる。

コンパイルド スタック(Stack, Compiled)
コンパイラが管理し割り当てるメモリの領域で、この領域内で変数に静的に空間を割り当てる。ターゲット デバイス上にソフトウェア スタックのメカニズムを効率的に実装できない場合、ソフトウェア スタックがコンパイルド スタックに置き換わる。このメカニズムでは、関数は再入可能ではなくなる。

スタティックRAM (SRAM) (Static RAM or SRAM)
Static Random Access Memoryの略。ターゲットボード上の読み書き可能なプログラムメモリ。リフレッシュ動作は不要。

ステータスバー(Status Bar)
MPLAB IDE/MPLAB X IDEウィンドウの一番下にあるバーで、カーソル位置、開発モードとデバイス、アクティブなツールバー等に関する情報が表示される。

ステップイントゥ(Step Into)
Single Stepと同じコマンド。Step Overとは異なり、Step IntoではCALL命令が呼び出すサブルーチン内もステップ実行する。

ステップオーバー(Step Over)
Step Overを実行すると、サブルーチン内をステップ実行せずにコードをデバッグできる。Step Overでは、`CALL`命令があると`CALL`の次の命令にブレークポイントが設定される。何らかの理由により、サブルーチンが無限ループになる等、正しくリターンしない場合、次のブレークポイントには到達しない。`CALL`命令の処理以外は、Step OverコマンドとSingle Stepコマンドは同じ。

ステップアウト(Step Out)
現在ステップ実行中のサブルーチンから抜け出すためのコマンド。このコマンドを実行すると、サブルーチンの残りのコードを全て実行し、サブルーチンの戻りアドレスで実行が停止する。

スティミュラス(Stimulus)
シミュレータへの入力、すなわち外部信号に対する応答をシミュレートするために生成するデータ。通常、テキストファイルにアクションのリストとしてこのデータを記述する。スティミュラスの種類には非同期、同期(ピン)、クロック動作、レジスタがある。

ストップウォッチ(Stopwatch)
実行サイクルを計測するためのカウンタ。

記憶域クラス(Storage Class)
指定されたオブジェクトに対応する記憶場所の持続期間を決定する。

記憶域修飾子(Storage Qualifier)
宣言されるオブジェクトの特別な属性を示す(例: const)。

シンボル(Symbol)
プログラムを構成する各種の要素を記述する汎用のメカニズム。関数名、変数名、セクション名、ファイル名、struct/enum/unionタグ名等がある。MPLAB IDE/MPLAB X IDEでは、主に変数名、関数名、アセンブリラベルをシンボルと呼ぶ。リンク実行後は、シンボルの値はメモリ内の値となる。

システム ウィンドウ コントロール(System Window Control)
ウィンドウと一部のダイアログの左上隅にあるコントロール。通常、このコントロールをクリックすると、[最小化]、[最大化]、[閉じる]等のメニュー項目がポップアップ表示される。

ターゲット(Target)
ユーザ ハードウェアの事。

ターゲット アプリケーション(Target Application)
ターゲットボードに読み込んだソフトウェア。

ターゲットボード(Target Board)
ターゲット アプリケーションを構成する回路とデバイス。

ターゲット プロセッサ(Target Processor)
ターゲット アプリケーションの基板で使われているMCU。

テンプレート(Template)
後でファイルに挿入するために作成するテキスト行。MPLABエディタでは、テンプレートはテンプレート ファイルに保存する。

ツールバー(Toolbar)
MPLAB IDE/MPLAB X IDEの機能を実行するためのボタン (アイコン)を縦または横に並べたもの。

トレース(Trace)
プログラム実行を記録するエミュレータまたはシミュレータの機能。エミュレータはプログラム実行のログをトレースバッファに記録し、これをMPLAB IDE/MPLAB X IDEのトレース ウィンドウにアップロードする。

トレースメモリ(Trace Memory)
エミュレータが内蔵するトレース用のメモリ。トレースバッファとも呼ばれる。

トレースマクロ(Trace Macro)
エミュレータ データからのトレース情報を提供するマクロ。これはソフトウェア トレースのため、トレースを利用するにはマクロをコードに追加し、コードを再コンパイルまたは再アセンブルし、ターゲット デバイスにこのコードをプログラミングする必要がある。

トリガ出力(Trigger Output)
任意のアドレスまたはアドレス範囲で生成でき、トレースとブレークポイントの設定から独立したエミュレータ出力信号の事。トリガ出力の設定数に制限はない。

トライグラフ(Trigraph)
「??」で始まる3文字のシーケンス。ISO Cで定義されており、1つの文字に置換される。

未割り当てセクション(Unassigned Section)
リンカのコマンドファイルで特定のターゲット メモリブロックに割り当てられていないセクション。リンカは、未割り当てセクションを割り当てるターゲット メモリブロックを検出する必要がある。

非初期化データ(Uninitialized Data)
初期値なしで定義されたデータ。Cでは、

```
int myVar;
```

は、非初期化済みデータセクションに格納される変数を定義する。

アップロード(Upload)
エミュレータやプログラマ等のツールからホストコンピュータへ、またはターゲットボードからエミュレータへデータを転送する事。

USB
Universal Serial Busの略。2本のシリアル伝送線でPCと外部周辺機器の通信を行う外部周辺インターフェイス規格。USB 1.0/1.1は最大12 Mbpsのデータレートをサポートしている。USB 2.0 (ハイスピードUSB)は最大480 Mbpsのデータレートをサポートしている。

ベクタ(Vector)
リセットまたは割り込みが発生した時にアプリケーションのジャンプ先となるメモリ番地。

Volatile
メモリ内の変数へのアクセス方法に影響を与えるコンパイラの最適化を抑制する変数修飾子。

警告(Warning)
警告
MPLAB IDE/MPLAB X IDEの場合: デバイス、ソフトウェア ファイル、装置に物理的な損傷を与える可能性のある状況で、ユーザに注意を促すために表示されるメッセージ。

16ビットアセンブラ/コンパイラの場合: 問題となる可能性のある状態を警告として報告するが、処理は停止されない。

ウォッチ変数(Watch Variable)
デバッグ セッション中に[Watches]ウィンドウで監視できる変数。

[Watches]ウィンドウ(Watches Window)
ウォッチ変数の一覧を表示し、ブレークポイントで毎回表示を更新するウィンドウ。

ウォッチドッグ タイマ(WDT: Watchdog Timer)
PICマイクロ コントローラに内蔵されたタイマの1つで、ユーザが設定した期間が経過するとプロセッサをリセットする。WDTの有効化/無効化、設定はコンフィグレーション ビットで行う。

ワークブック(Workbook)
MPLAB SIMシミュレータにおいて、SCLスティミュラスの生成に関する設定を保存したもの。

Microchip社のウェブサイト

Microchip社はウェブサイト(www.microchip.com)を通してオンライン サポートを提供しています。当ウェブサイトでは、お客様に役立つ情報やファイルを簡単に見つけ出せます。以下を含む各種の情報がご覧になれます。

- **製品サポート** - データシートとエラッタ、アプリケーション ノートとサンプル プログラム、設計リソース、ユーザガイドとハードウェア サポート文書、最新のソフトウェアと過去のソフトウェア
- **技術サポート** – よく寄せられる質問(FAQ)、技術サポートのご依頼、オンライン ディスカッション グループ、Microchip社のコンサルタント プログラムおよびメンバーリスト
- **ご注文とお問い合わせ** - 製品セレクタと注文ガイド、最新プレスリリース、セミナー/イベントの一覧、お問い合わせ先(営業所/販売代理店)の一覧

お客様への通知サービス

Microchip社のお客様向け変更通知サービスは、お客様にMicrochip社製品の最新情報をお届けする配信サービスです。ご興味のある製品ファミリまたは開発ツールに関する変更、更新、リビジョン、エラッタ情報をいち早くメールにてお知らせします。

http://www.microchip.com/pcnにアクセスし、登録手続きをしてください。

お客様サポート

Microchip社製品をお使いのお客様は、以下のチャンネルからサポートをご利用頂けます。

- 正規代理店
- 弊社営業所
- 技術サポート

サポートは販売代理店にお問い合わせください。各地の営業所もご利用頂けます。本書の最後のページに各国の営業所の一覧を記載しています。

技術サポートは以下のウェブページからもご利用になれます。
http://www.microchip.com/support

Microchip社のデバイスコード保護機能

Microchip社製デバイスのコード保護機能に関して以下の点にご注意ください。

- Microchip社製品は、該当するMicrochip社データシートに記載の仕様を満たしています。
- Microchip社では、通常の条件ならびに仕様に従って使用した場合、Microchip社製品のセキュリティ レベルは、現在市場に流通している同種製品の中でも最も高度であると考えています。
- しかし、コード保護機能を解除するための不正かつ違法な方法が存在する事もまた事実です。弊社の理解では、こうした手法は全てMicrochip社データシートにある動作仕様書以外の方法でMicrochip社製品を使用する事になります。このような行為は知的所有権の侵害に該当する可能性が非常に高いと言えます。
- Microchip社はコードの保全性に懸念を抱いているお客様と連携して対応策に取り組んでいきます。
- Microchip社を含む全ての半導体メーカーで、自社のコードのセキュリティを完全に保証できる企業はありません。コード保護機能とは、Microchip社が製品を「解読不能」として保証するものではありません。

コード保護機能は常に進歩しています。Microchip社では、常に製品のコード保護機能の改善に取り組んでいます。Microchip社のコード保護機能の侵害は、デジタル ミレニアム著作権法に違反します。そのような行為によってソフトウェアまたはその他の著作物に不正なアクセスを受けた場合、デジタル ミレニアム著作権法の定める所により損害賠償訴訟を起こす権利があります。

法律上の注意点

本書に記載されているデバイス アプリケーション等の情報は、ユーザの便宜のためにのみ提供されるものであり、更新によって変更となる事があります。お客様のアプリケーションが仕様を満たす事を保証する責任は、お客様にあります。

Microchip社は、明示的、暗黙的、書面、口頭、法定のいずれであるかを問わず、本書に記載されている情報に関して、状態、品質、性能、商品性、特定目的への適合性をはじめとする、いかなる類の表明も保証も行いません。Microchip社は、本書の情報およびその使用に起因する一切の責任を否認します。Microchip社の明示的な書面による承認なしに、生命維持装置あるいは生命安全用途にMicrochip社の製品を使用する事は全て購入者のリスクとし、また購入者はこれによって発生したあらゆる損害、クレーム、訴訟、費用に関して、Microchip社は擁護され、免責され、損害をうけない事に同意するものとします。暗黙的あるいは明示的を問わず、Microchip社が知的財産権を保有しているライセンスは一切譲渡されません。

商標

Microchip社の名称とロゴ、Microchipロゴ、Adaptec、AnyRate、AVR、AVRロゴ、AVR Freaks、BesTime、BitCloud、chipKIT、chipKITロゴ、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemiロゴ、MOST、MOSTロゴ、MPLAB、OptoLyzer、PackeTime、PIC、picoPower、PICSTART、PIC32ロゴ、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SSTロゴ、SuperFlash、Symmetricom、SyncServer、Tachyon、TempTrackr、TimeSource、tinyAVR、UNI/O、Vectron、XMEGAは米国およびその他の国におけるMicrochip Technology Incorporatedの登録商標です。

APT、ClockWorks、The Embedded Control Solutions Company、EtherSynch、FlashTec、Hyper Speed Control、HyperLight Load、IntelliMOS、Libero、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plusロゴ、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、Vite、WinPath、ZLは米国におけるMicrochip Technology Incorporatedの登録商標です。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BlueSky、BodyCom、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、INICnet、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNetロゴ、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certifiedロゴ、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、ViewSpan、WiperLock、Wireless DNA、ZENAは米国とその他の国におけるMicrochip Technology Incorporatedの商標です。

SQTPは米国におけるMicrochip Technology Incorporatedのサービス マークです。

Adaptecロゴ、Frequency on Demand、Silicon Storage Technology、Symmcomはその他の国におけるMicrochip Technology Incorporatedの登録商標です。

GestICは、その他の国におけるMicrochip Technology Germany II GmbH & Co. KG (Microchip Technology Inc.の子会社)の登録商標です。

その他の商標は各社に帰属します。

© 2019, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-5264-5

品質管理システム

Microchip社の品質管理システムについては、http://www.microchip.com/qualityをご覧ください。

各国の営業所とサービス

北米	アジア太平洋	アジア太平洋	欧州
本社 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術サポート: http://www.microchip.com/support URL: http://www.microchip.com **アトランタ** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **オースティン、TX** Tel: 512-257-3370 **ボストン** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **シカゴ** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **ダラス** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **デトロイト** Novi, MI Tel: 248-848-4000 **ヒューストン、TX** Tel: 281-894-5983 **インディアナポリス** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **ロサンゼルス** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **ローリー、NC** Tel: 919-844-7510 **ニューヨーク、NY** Tel: 631-435-6000 **サンノゼ、CA** Tel: 408-735-9110 Tel: 408-436-4270 **カナダ - トロント** Tel: 905-695-1980 Fax: 905-695-2078	**オーストラリア - シドニー** Tel: 61-2-9868-6733 **中国 - 北京** Tel: 86-10-8569-7000 **中国 - 成都** Tel: 86-28-8665-5511 **中国 - 重慶** Tel: 86-23 -8980-9588 **中国 - 東莞** Tel: 86-769 -8702-9880 **中国 - 広州** Tel: 86-20 -8755-8029 **中国 - 杭州** Tel: 86-571 -8792-8115 **中国 - 香港SAR** Tel: 852-2943-5100 **中国 - 南京** Tel: 86-25 -8473-2460 **中国 - 青島** Tel: 86-532 -8502-7355 **中国 - 上海** Tel: 86-21 -3326-8000 **中国 - 瀋陽** Tel: 86-24 -2334-2829 **中国 - 深圳** Tel: 86-755 -8864-2200 **中国 - 蘇州** Tel: 86-186 -6233-1526 **中国 - 武漢** Tel: 86-27 -5980-5300 **中国 - 西安** Tel: 86-29 -8833-7252 **中国 - 厦門** Tel: 86-592-2388138 **中国 - 珠海** Tel: 86-756-3210040	**インド - バンガロール** Tel: 91-80 -3090-4444 **インド - ニューデリー** Tel: 91-11 -4160-8631 **インド - プネ** Tel: 91-20 -4121-0141 **日本 - 大阪** Tel: 81-6 -6152-7160 **日本 - 東京** Tel: 81-3-6880-3770 **韓国 - 大邱** Tel: 82-53 -744-4301 **韓国 - ソウル** Tel: 82-2 -554-7200 **マレーシア - クアラルンプール** Tel: 60-3 -7651-7906 **マレーシア - ペナン** Tel: 60-4 -227-8870 **フィリピン - マニラ** Tel: 63-2 -634-9065 **シンガポール** Tel: 65-6334-8870 **台湾 - 新竹** Tel: 886-3 -577-8366 **台湾 - 高雄** Tel: 886-7 -213-7830 **台湾 - 台北** Tel: 886-2 -2508-8600 **タイ - バンコク** Tel: 66-2 -694-1351 **ベトナム - ホーチミン** Tel: 84-28 -5448-2100	**オーストリア - ヴェルス** Tel: 43-7242 -2244-39 Fax: 43-7242 -2244-393 **デンマーク - コペンハーゲン** Tel: 45-4450-2828 Fax: 45-4485-2829 **フィンランド - エスポー** Tel: 358-9 -4520-820 **フランス - パリ** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **ドイツ - ガーヒンク** Tel: 49-8931-9700 **ドイツ - ハーン** Tel: 49-2129-3766400 **ドイツ - ハイルブロン** Tel: 49-7131-72400 **ドイツ - カールスルーエ** Tel: 49-721-625370 **ドイツ - ミュンヘン** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **ドイツ - ローゼンハイム** Tel: 49-8031 -354-560 **イスラエル - ラーナナ** Tel: 972-9 -744-7705 **イタリア - ミラノ** Tel: 39-0331-742611 Fax: 39-0331-466781 **イタリア - パドヴァ** Tel: 39-049-7625286 **オランダ - ドリューネン** Tel: 31-416-690399 Fax: 31-416-690340 **ノルウェー - トロンハイム** Tel: 47-72884388 **ポーランド - ワルシャワ** Tel: 48-22-3325737 **ルーマニア - ブカレスト** Tel: 40-21-407-87-50 **スペイン - マドリッド** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **スウェーデン - ヨーテボリ** Tel: 46-31-704-60-40 **スウェーデン - ストックホルム** Tel: 46-8 -5090-4654 **イギリス - ウォーキンガム** Tel: 44-118 -921-5800 Fax: 44-118 -921-5820