MPLAB® X IDE
ユーザガイド

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。

マイクロチップ社製デバイスのコード保護機能に関して次の点にご注意ください。

- マイクロチップ社製品は、該当するマイクロチップ社データシートに記載の仕様を満たしています。
- マイクロチップ社では、通常の条件ならびに仕様に従って使用した場合、マイクロチップ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、マイクロチップ社データシートにある動作仕様書以外の方法でマイクロチップ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。
- マイクロチップ社は、コードの保全性に懸念を抱くお客様と連携し、対応策に取り組んでいきます。
- マイクロチップ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、マイクロチップ社が製品を「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。マイクロチップ社では、常に製品のコード保護機能の改善に取り組んでいます。マイクロチップ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は、デジタル ミレニアム著作権法の定めるところにより損害賠償訴訟を起こす権利があります。

商標

ISBN: 978-1-62076-134-2

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2009

マイクロチップ社では、Chandler および Tempe (アリゾナ州)、Gresham (オレゴン州) の本部、設計部およびウェハー製造工場そしてカリフォルニア州とイドのデザインセンターが ISO/TS-16949:2009 認証を取得しています。マイクロチップ社の品質システム プロセスおよび手順は、PIC® MCU および dsPIC® DSC、KEELOQ® コード ホッピング デバイス、シリアル EEPROM、マイクロペリフェラル、不揮発性メモリ、アナログ製品に採用されています。さらに、開発システムの設計と製造に関するマイクロチップ社の品質システムは ISO 9001:2000 認証を取得しています。

目次

Chapter 4. 基本作業

Chapter 5. 追加の作業

Chapter 6. 高度な作業

Chapter 7. トラブルシュート

Chapter 8. MPLAB X IDE と MPLAB IDE v8 の相違点

序章

はじめに

序章では、MPLAB® X IDE を使い始める前に知っておくと便利な下記の一般的事項について説明します。

- 本書の構成
- 表記規則
- 推奨参考資料

本書の構成

本書では MPLAB X IDE の使い方について説明します。本書の構成は以下の通りです。

- **Chapter 1.「MPLAB® X IDE とは」** – MPLAB X IDE の概要とヘルプの使い方
- **Chapter 2.「使用前の準備」** – ハードウェア ツール用 USB ドライバと、コードのコンパイル / アセンブルに使う言語ツールパッケージのインストール方法
- **Chapter 3.「チュートリアル」** – MPLAB X IDE を使うための各種機能の手順を追った説明
- **Chapter 4.「基本作業」** – MPLAB X IDE の基本機能 (前章「チュートリアル」で使った機能の詳細な説明)
- **Chapter 5.「追加の作業」** – MPLAB X IDE の追加機能 (MPLAB IDE v8 プロジェクトのインポート、ストップウォッチ等)
- **Chapter 6.「高度な作業」** – MPLAB X IDE の高度な機能 (複数プロジェクトおよび複数プロジェクト コンフィグレーションの使用等)
- **Chapter 7.「トラブルシュート」** – 問題への対処方法
- **Chapter 8.「MPLAB X IDE と MPLAB IDE v8 の相違点」** – MPLAB IDE v8 との主な相違点 (機能、メニュー、ツール サポート等)
- **Chapter 9.「デスクトップの詳細」** – MPLAB X IDE デスクトップ アイテム (メニュー、ツールバー、ステータスバー等)
- **Chapter 10.「ウィンドウとダイアログ」** – NetBeans™ ウィンドウおよびダイアログと、MPLAB X IDE 専用ウィンドウおよびダイアログ
- **Chapter 11.「プロジェクト ファイルおよびフォルダ」** – フォルダ構造とプロジェクト ファイルの保存場所
- **Chapter 12.「コンフィグレーション設定の要約」** – 各種の言語ツールを使って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 ([Configurations Settings] 設定ウィンドウは、デバッグ用に一時的にコンフィグレーション ビットを設定するだけであるため、MPLAB X IDE ではこの手順が必要です)

表記規則

本書では以下の表記規則を適用します。

本書の表記規則

内容	意味	例
Arial、MS ゴシック フォント :		
二重かぎカッコ : 『 』	参考資料	『MPLAB® IDE ユーザガイド』
太字	テキストの強調	... は**唯一**のコンパイラです ...
角カッコ : [ ]	ウィンドウ名	[Output] ウィンドウ
	ダイアログ名	[Settings] ダイアログ
	メニューの選択肢	[Enable Programmer] を選択
かぎカッコ : 「 」	ウィンドウまたはダイアログ内のフィールド名、メニュー項目名	「Save project before build」
角カッコで囲んだテキストと右山カッコ (>)、下線付き	メニューの選択方法 (テキスト内に記載の場合)	[File]>[Save]
テキストと右山カッコ (>)、下線なし	メニューの選択方法 (表セル内に記載の場合)	File>Save
角カッコで囲んだ太字のテキスト	ダイアログのボタン	**[OK]** をクリックする
	タブ	**[Power]** タブをクリックする
山カッコ (< >) で囲んだテキスト	キーボードのキー	<Enter>、<F1> を押す
Courier New フォント		
標準の Courier New フォント	サンプル ソースコード	`#define START`
	ファイル名	`autoexec.bat`
	ファイルパス	`c:\mcc18\h`
	キーワード	`_asm, _endasm, static`
	コマンドライン オプション	`-Opa+, -Opa-`
	ビット値	`0, 1`
	定数	`0xFF,'A'`
斜体	変数の引数	*`file`*`.o` (*`file`* は有効な任意のファイル名)
角カッコ : []	オプションの引数	`mpasmwin [options] file [options]`
中カッコとパイプ文字 : {|}	いずれかの引数を選択する場合 (OR 選択)	`errorlevel {0|1}`
...	繰り返されるテキスト	`var_name [, var_name...]`
	コード内でユーザが定義する部分	`void main (void) { ... }`

推奨参考資料

本書『MPLAB X IDE のユーザガイド』以外の参考資料として、以下に記載したマイクロチップ社の文書をお薦めします。

MPLAB IDE の Readme ファイル

`Readme for MPLAB IDE.htm` ファイルには、MPLAB X IDE の使用に関する最新情報を記載しています。このファイルは、MPLAB X IDE のインストール先ディレクトリの「Readmes」サブディレクトリ内にあります。Readme ファイルには、本書に記載できなかった最新情報と既知の問題を記載しています。

Readme ファイル

その他のツールの使用に関する最新情報は、MPLAB X IDE のインストール先ディレクトリの「Readmes」サブディレクトリ内にある各ツールの Readme ファイルを参照してください。Readme ファイルには、本書に記載できなかった最新情報と既知の問題を記載しています。

オンラインヘルプ ファイル

MPLAB X IDE、MPLAB エディタ、MPLAB SIM シミュレータではチュートリアル、機能説明、参照資料を含む包括的なヘルプファイルを利用できます。

デバイス データシートとファミリ リファレンス マニュアル

全ての PIC® MCU および dsPIC® DSC デバイスのデータシートとファミリ リファレンス マニュアルの最新版は、マイクロチップ社ウェブサイトでご覧になれます。

NOTE:

Chapter 1. MPLAB® X IDE とは

1.1 組み込みシステムの概要

MPLAB X IDE は、マイクロチップ社製マイクロコントローラおよびデジタルシグナル コントローラ向けのアプリケーション開発用ソフトウェア プログラムです。このソフトウェアは統合開発環境 (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。組み込みシステムの設計に既に詳しい読者は、この章を読まずに次章へ進んでもかまいません。ここでは、組み込みシステムの開発と MPLAB X IDE の使い方について簡単に紹介します。

1.1.1 「組み込みシステム」について

一般的に組み込みシステムとは、マイクロチップ社の PIC MCU や dsPIC デジタルシグナル コントローラ (DSC) 等の小型マイクロコントローラの機能を利用したシステムを指します。このようなマイクロコントローラは、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。

1.1.2 組み込みコントローラと PC の違い

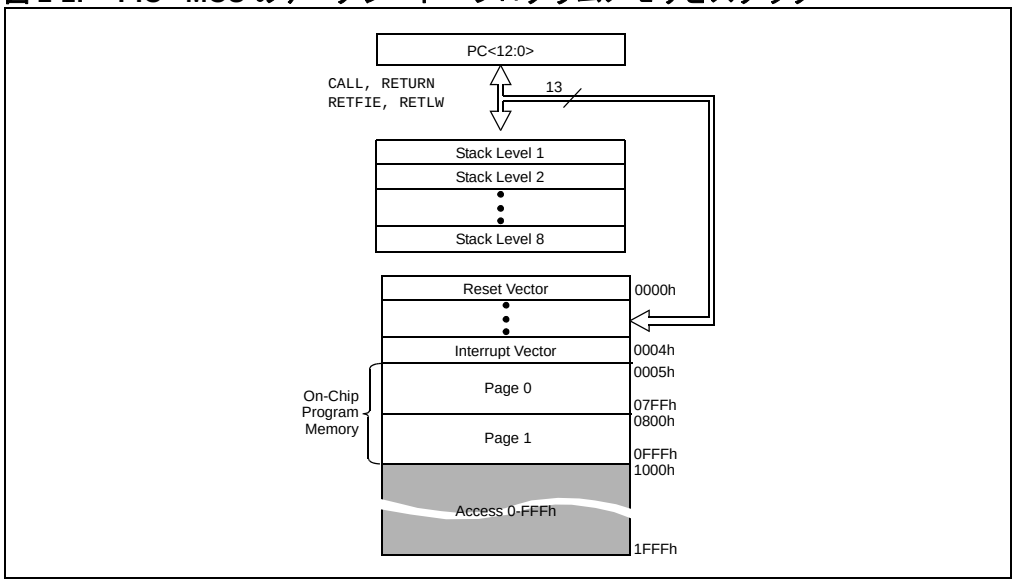
組み込みコントローラは、1 つまたは複数の限定されたタスクだけを実行するという点で、PC とは大きく異なります。PC は不特定のプログラムを実行でき、各種の外部デバイスへ接続できます。組み込みコントローラは 1 つのプログラムだけを実行するため、そのタスクの実行に必要な処理能力とハードウェアだけを実装するだけで済み、結果として価格を低く抑える事ができます。対して PC は、比較的高価な汎用中央演算装置 (CPU) を核とし、多数の外部デバイス (メモリ、ディスク ドライブ、ビデオ コントローラ、ネットワーク インターフェイス回路など) を備える必要があります。組み込みシステムは、演算装置として低コストのマイクロコントローラ ユニット (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) の内部部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、装置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、コスト効率が悪く、9 V の電池 1 個だけで何年間も無人稼動させる事はできません。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、多くの場合安価なマイクロコントローラが使われます。

1.1.3 マイクロコントローラのコンポーネント

PIC MCU は、プログラムを実行するためのファームウェア (コード化された命令) を格納するプログラムメモリを内蔵しています (図 1-1)。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) を指定するために、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



マイクロコントローラは、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 (図 1-3 参照)。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格納用に使います。マイクロコントローラには、EEPROM メモリを追加で内蔵したものもあります。プログラムメモリと同様に、データメモリの使用方法と動作の詳細も、デバイス データシートに記載しています。

図 1-2: PIC® MCU のデータシート – ファイルレジスタ

	File Address		File Address		File Address		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
TXREG	19h	SPBRG	99h	CM1CON0	119h		199h
RCREG	1Ah	SPBRGH	9Ah	CM2CON0	11Ah		19Ah
	1Bh	BAUDCTL	9Bh	CM2CON1	11Bh		19Bh
PWM1CON	1Ch		9Ch		11Ch		19Ch
ECCPAS	1Dh		9Dh		11Dh	PSTRCON	19Dh
ADRESH	1Eh	ADRESL	9Eh	ANSEL	11Eh	SRCON	19Eh
ADCON0	1Fh	ADCON1	9Fh	ANSELH	11Fh		19Fh
General Purpose Register 96 Bytes	20h – 7Fh	General Purpose Register 80 Bytes	A0h – EFh	General Purpose Register 80 Bytes	120h – 16Fh		1A0h
		accesses 70h-7Fh	F0h – FFh	accesses 70h-7Fh	170h – 17Fh	accesses 70h-7Fh	1F0h – 1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

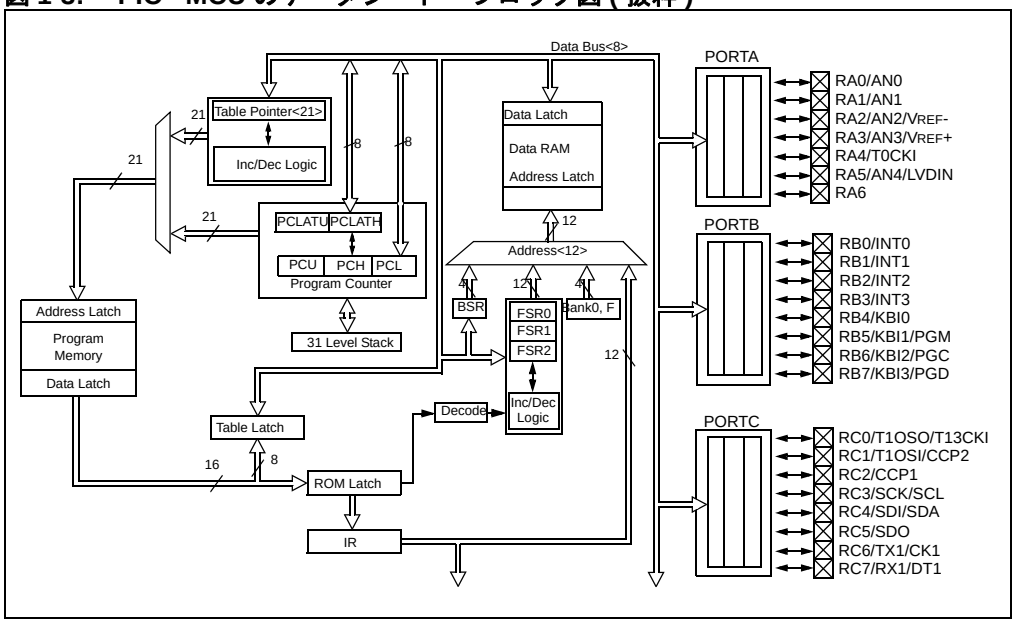
未実装のデータメモリ領域、「0」として読み出し

Note 1: 物理レジスタではありません。

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。

メモリ以外にも、マイクロコントローラは同一チップ上に各種の周辺デバイス回路を内蔵しています。周辺デバイスには入出力 (I/O ポート) を含みます。I/O ポートはマイクロコントローラの出力として使用可能なピンに割り当てられ、HIGH/LOW に駆動する事により、ランプの点滅やスピーカの駆動等、配線を通して各種の信号を送信できます。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。入力として使う事により、外部スイッチやセンサからの入力に対する応答や、外部デバイスとの通信が可能になります。

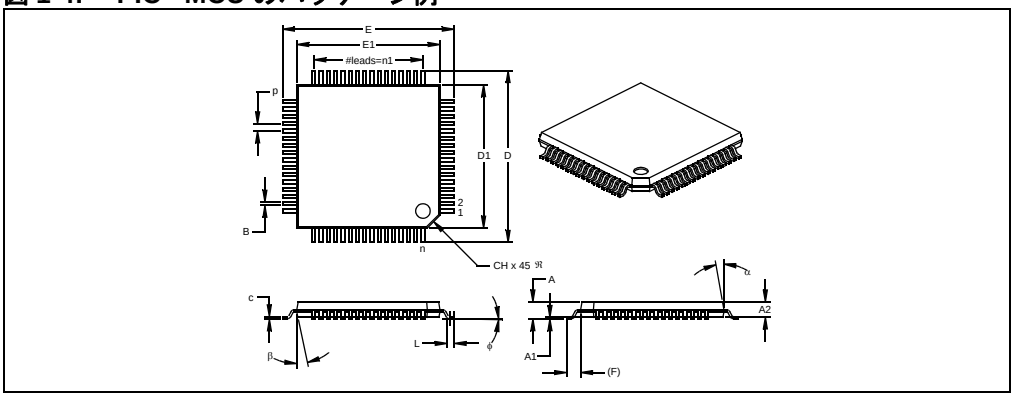
図 1-3: PIC® MCU のデータシート – ブロック図 (抜粋)



システムを設計する場合、まず、アプリケーションに適合する周辺モジュールを選択する必要があります。A/D コンバータ (ADC) を使うと、マイクロコントローラ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、配線を通して別のマイクロコントローラ、ローカル ネットワーク、インターネットとストリーム通信が行えます。PIC MCU 上の「タイマ」周辺モジュールを使うと、信号イベントの正確な計測、通信信号の生成と受信、正確な波形の生成が可能となる他に、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、マイクロコントローラ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て、マイクロコントローラ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。

PIC MCU の選択は、アプリケーション プログラム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、マイクロコントローラの消費電力とフォームファクタ (回路に実装可能なサイズとパッケージ形態) が挙げられます。

図 1-4: PIC® MCU のパッケージ例



1.1.4 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ド作成、編集、デバッグと、マイクロコントローラへのプログラムの書き込みを支援するソフトウェア システムです。MPLAB X IDE は、組み込みシステム アプリケーションの設計および実装に必要な全てのコンポーネントを備えています。

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典型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 PIC MCU または dsPIC DSC デバイスを選択し、関連するハードウェア回路を設計します。さらに、ハードウェアを制御する周辺モジュールとピンを決めた後に、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。

図 1-5: PIC® MCU のデータシート – タイミング (抜粋)

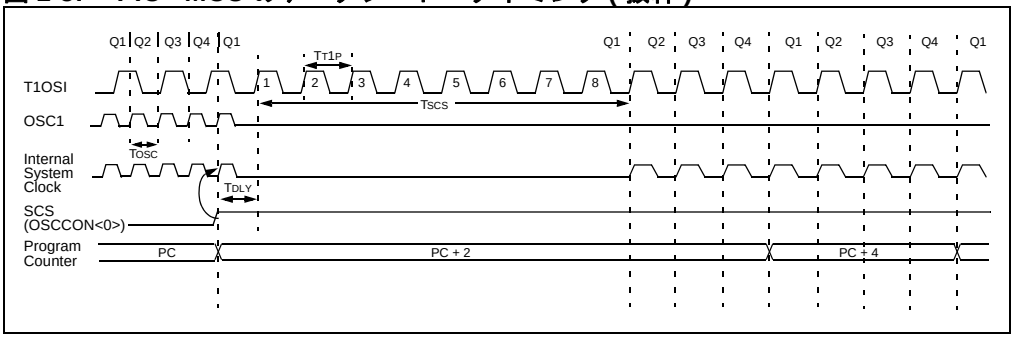
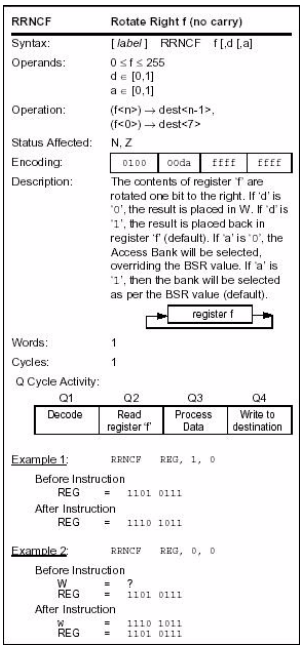


図 1-6:　PIC® MCU のデータシート – 命令 (抜粋)



RRNCF　Rotate Right f (no carry)

Syntax: [*label*]　RRNCF　f [,d [,a]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f<n>) → dest<n-1>,
(f<0>) → dest<7>

Status Affected: N, Z

Encoding:

0100	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

register f

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1:　RRNCF　REG, 1, 0

Before Instruction
REG = 1101 0111
After Instruction
REG = 1110 1011

Example 2:　RRNCF　REG, 0, 0

Before Instruction
W = ?
REG = 1101 0111
After Instruction
W = 1110 1011
REG = 1101 0111

2. アセンブラおよび / またはコンパイラと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。最終的には、このマシンコードをファームウェアとしてマイクロコントローラに書き込みます。
3. 作成したコードをマイクロコントローラに書き込む前に、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、正しく動作させるためにプログラムの「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や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、変数の値を変更してルーチンの動作を確認する「what if」チェックを行えます。
4. 最後に、マシンコードをマイクロコントローラに「書き込んで」、最終的にアプリケーション内で正しく動作する事を確認します。

上記の手順はいずれも非常に複雑な作業を伴います。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 とそのコンポーネントを使う事により、煩わしい操作の習得に時間を費やさずに、本来の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。プログラミング用エディタは、各種の言語ツールに対応し、精度の高いコード作成を支援します。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 (実行速度またはバイナリサイズ優先) の厳密さと、変数やプログラム データのメモリ格納領域を設定できます。マイクロコントローラ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、エラー発生行が示されるので、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。コードを修正した後に、アプリケーションをビルドし直して、エラーが発生しない事を確認します。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 - 修正のループを多数回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 PIC MCU と dsPIC DSC に対応するデバッガと、無償のソフトウェア シミュレータを、コンポーネントとして備えています。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、マイクロコントローラ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行えます。

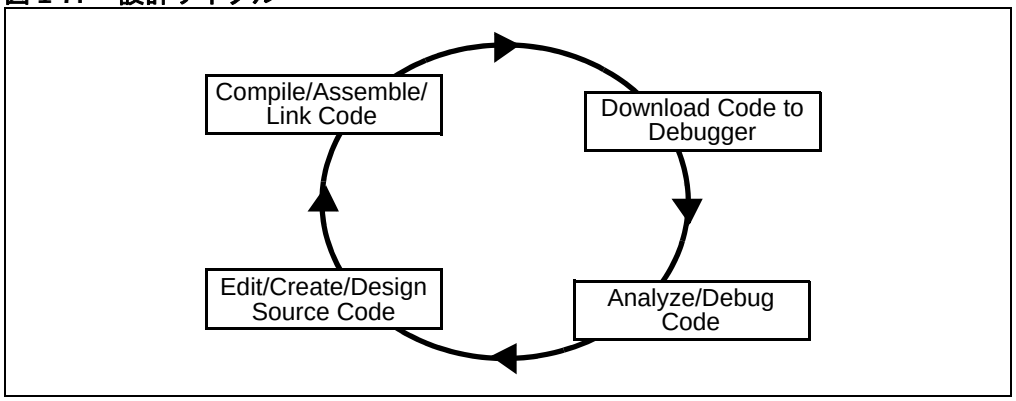
プロトタイプのハードウェアをアプリケーションに組み込める段階になると、インサーキット エミュレータやインサーキット デバッガ等のハードウェア デバッガを使えます。これらのデバッグツールは、フラッシュ プログラムメモリを内蔵したデバイスの多くが備える特殊な回路を使って、実際のアプリケーション上でコードをリアルタイムに実行します。これらのツールを使うと、プログラムの実行を停止 / 再開して、ターゲット マイクロコントローラのプログラムメモリとデータメモリの内容を観察できるため、アプリケーションにマイクロコントローラを組み込んだ状態でコードをテストできます。

アプリケーションが正常に動作したら、マイクロチップ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、マイクロコントローラにプログラムを書き込みます。これらのプログラマは、完成したコードが予期した通りに動作する事を検証します。MPLAB X IDE は、ほとんどの PIC MCU と全ての dsPIC DSC をサポートします。

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。完全に機能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。このような開発において、統合開発環境は、多数のツールを繰り返し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、ツールや操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、本来のアプリケーション開発により集中できます。

図 1-7: 設計サイクル

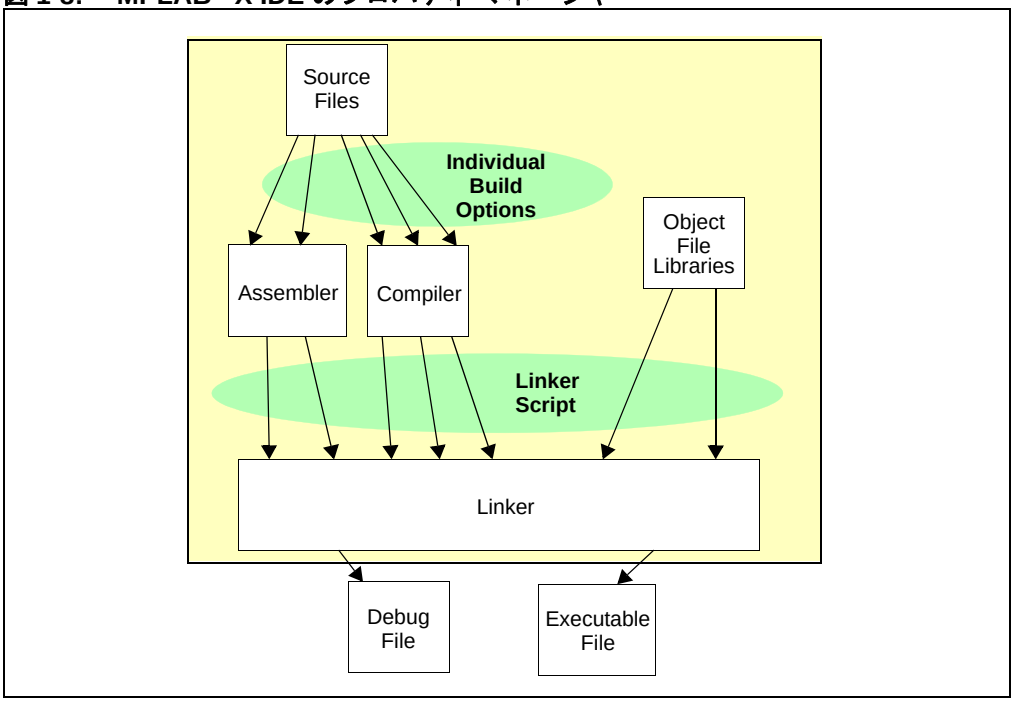


MPLAB X IDE は、共通のグラフィカル ユーザインターフェイスを介して全てのツールを (通常は自動的に) 連動させる「ラッパー」として機能します。例えば、ソースコードを記述し、これを実行可能なマシンコードに変換し、マイクロコントローラにプログラミングし、その動作を確認するといった一連の作業を共通の環境内で行えます。開発プロセスでは、コードを記述するためのエディタ、ファイルと設定を管理するためのプロジェクト マネージャ、ソースコードをマシンコードに変換するためのコンパイラやアセンブラ、マイクロコントローラの接続やマイクロコントローラの動作をシミュレートするための各種ハードウェア / ソフトウェア等、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 1-8: MPLAB® X IDE のプロパティ マネージャ



ソースファイルは、アセンブラや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、このプレースホルダに基づいて、全てのモジュールから実行可能な 1 つのマシンコード 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な機能を備えています。コードを作成した後、この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も表示します。エディタでブレークポイント (コードの実行を一時停止 (ブレーク) する位置) を設定してプログラムを実行すると、変数名の上にマウスポインタを置くだけで変数の値が表示されます。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 & ドロップする事により、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。

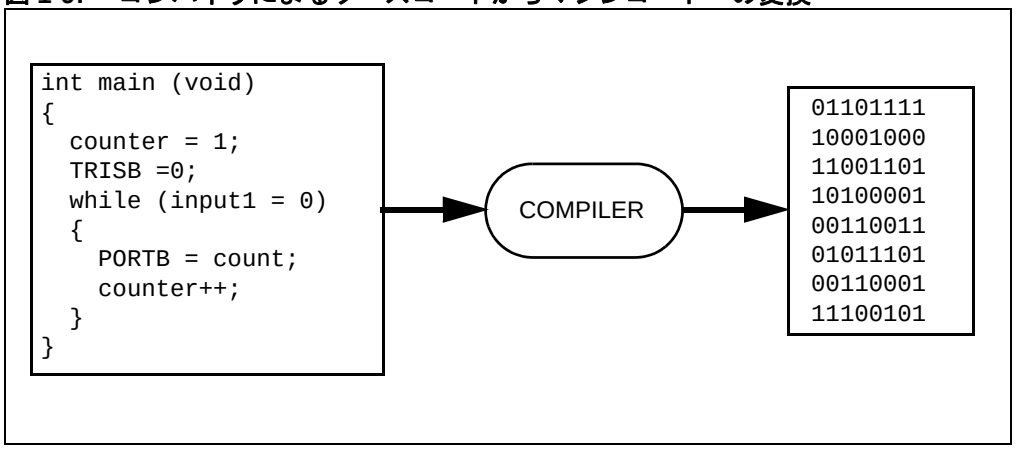
1.4 言語ツール

「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が、組み込みシステムでは「クロスアセンブラ」や「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、一般的なコンパイラとは異なり、その PC ではなく別のマイクロプロセッサ (またはマイクロコントローラ) 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マイクロコントローラ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ための技量が求められます。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、1KB 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 もの記憶容量を使うものもあります。

図 1-9: 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

アプリケーション開発では、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、マイクロコントローラの動作を 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。

1.5.1 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用方法を習得すれば、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 を使ってマイクロコントローラの動作をシミュレートするため、通常は実際のマイクロコントローラよりも低速で動作します。

1.5.2 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、マイクロコントローラの内部メモリに単純に書き込みます。これにより、アプリケーションに組み込んだマイクロコントローラ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極めて稀です。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、マイクロコントローラ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。プログラマでも、コードを変更してマイクロコントローラをプログラミングし直す事はできますが、そのたびにマイクロコントローラを抜き差しして再テストしなければなりません。このような方法で複雑なコードを扱うと、時間と工数がかかるだけでなく、ハードウェアの問題を正確に理解する事は困難です。

このような場合にはハードウェア 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マイクロコントローラを使うインサーキット エミュレータまたはインサーキット 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。

1.5.3 統合開発環境

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、デバッグは緊急の問題になります。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なければなりませんが、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。各種のツールを共通の環境で使う事ができれば、このような作業の「サイクル」にかかる時間を短縮できます。致命的なバグを全て取り除かなければならない最終ステップでは、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価格のインサーキット 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。

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実装した後でもプログラミングできます。このような場合、In-Circuit Serial Programming™ (ICSP™) プログラミング機能を使って、製造時に書き込んだファームウェアを後から書き直す事ができます。インサーキット デバッグをサポートしているデバイスであれば、完成品をインサーキット デバッガに接続して、製造後の品質検査やファームウェアの改良開発も行えます。

1.7 MPLAB X IDE のコンポーネント

MPLAB X IDE は下記を備えます。

- 機能を完備した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。
- 各種のアセンブラ / 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の開発用に使います。
- 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、最新のデバッガが備える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 MCU と dsPIC DSC デバイスに対応します。シミュレータの実行ファイルは、各種デバイス別に用意されています。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。

下記の 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いただけます。

- **コンパイラ言語ツール**

マイクロチップ社の MPLAB C コンパイラは、PIC MCU と dsPIC DSC 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。MPLAB C コンパイラ以外に、microEngineering Labs、CCS、SDCC が提供するコンパイラを 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、即座にテストおよび検証できます。

- **プログラマ**

ターゲット デバイスへのコードのプログラミングには、MPLAB PM3 プログラマ、PICkit™ 2、PICkit 3、MPLAB ICD 3 インサーキット デバッガ、MPLAB REAL ICE™ インサーキット エミュレータを使えます。MPLAB X IDE は、コードとデータだけでなく、ターゲット マイクロコントローラまたはデジタルシグナル コントローラ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完全に制御します。

- **インサーキット デバッガ / エミュレータ**

PICkit™ 2、PICkit 3、MPLAB ICD 3 インサーキット デバッガ、MPLAB REAL ICE™ インサーキット 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 マイクロコントローラ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、レジスタと変数の観察を可能とします。エミュレータでは、その他のデバッグ機能 (トレース等) 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、各種のプラグインを利用できます。例えば、DMCI (Data Monitor and Control Interface) を使うと、コード内の変数を表示および制御し、それらの値をリアルタイムに変更できます。また、DMCI を使うと、出力データをグラフィカルに表示する事もできます。

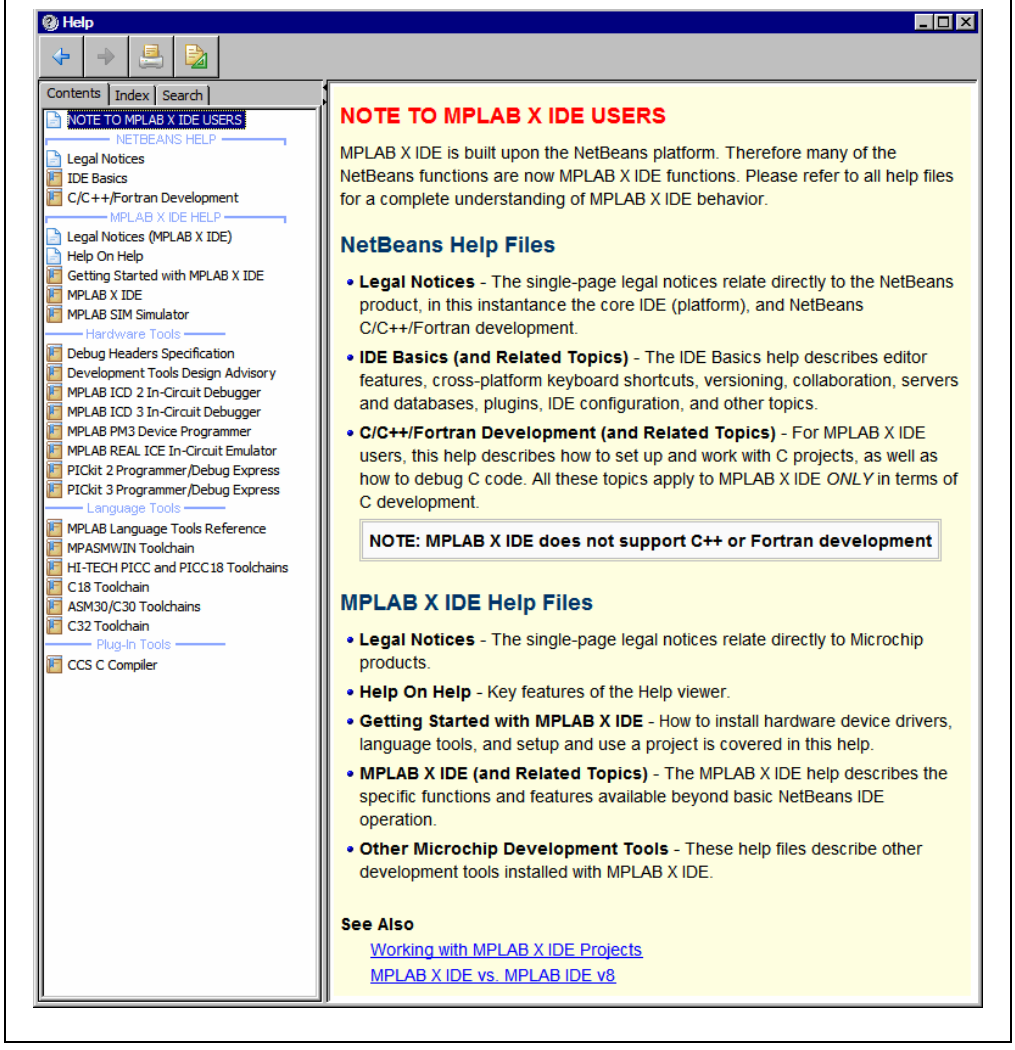
1.8 MPLAB X IDE オンラインヘルプ

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。このため、多くの NetBeans 機能を MPLAB X IDE の機能として利用できます。

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 (図 1-10 参照)。NetBeans に関する情報は、[Help] ウィンドウの左の [Contents] タブ内で、「NetBeans Help」の下に表示される各種のオンラインヘルプ ファイルを参照してください。MPLAB X IDE 開発ツールに関する情報は、[Help] ウィンドウの左の [Contents] タブ内で、「MPLAB X IDE Help」の下に表示される各種のオンラインヘルプ ファイルを参照してください。

MPLAB X IDE と MPLAB IDE v8 の相違点については、**Chapter 8.「MPLAB X IDE と MPLAB IDE v8 の相違点」**を参照してください。

図 1-10: 最新の MPLAB X IDE ヘルプ (NetBeans 関連項目を含む)



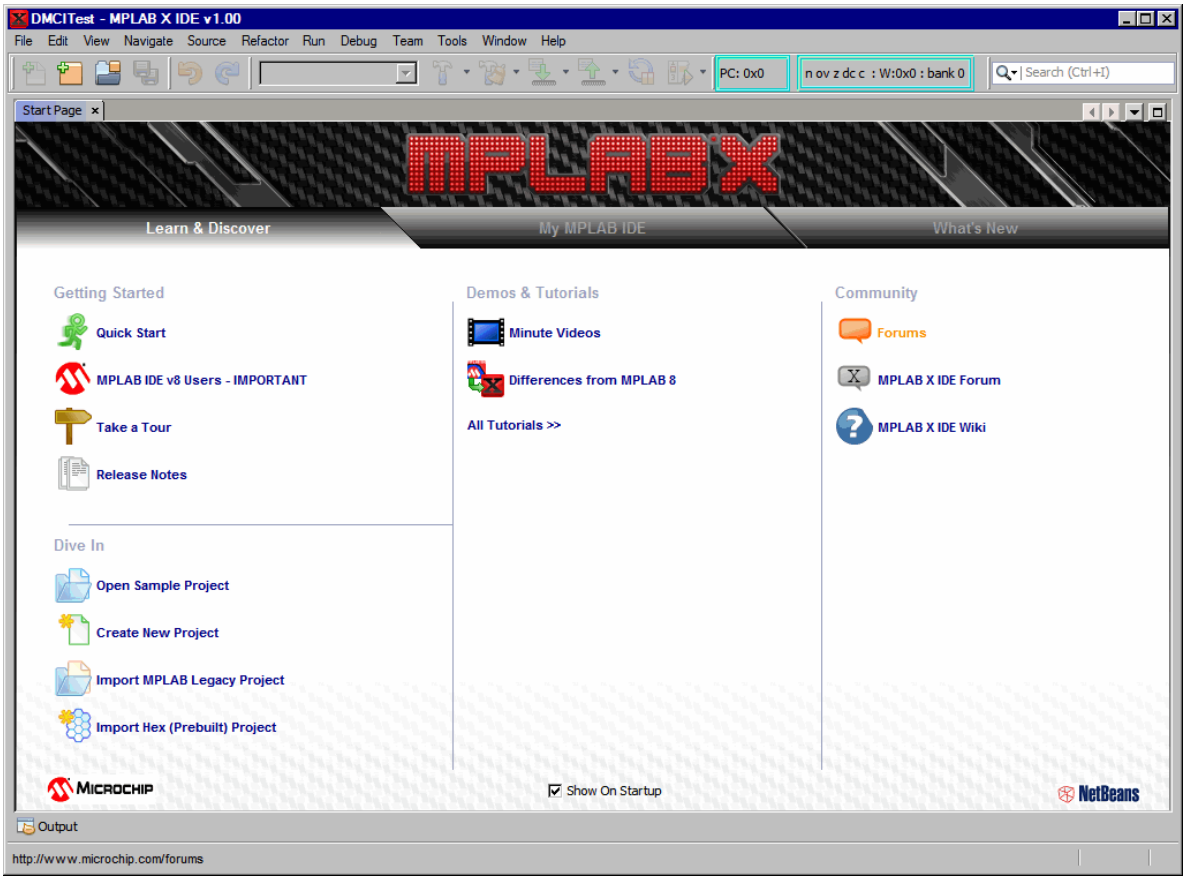
1.9 その他の MPLAB X IDE 関連文書

ヘルプ以外の文書とビデオおよびフォーラムへのリンクは [Start Page] から利用できます (図 1-11)。

MPLAB IDE v8 関連の文書も参考になります (例 : 既知のハードウェア問題を記載したハードウェア ツールのリリースノート等)。これらの文書については、下記のマイクロチップ社ウェブサイトを参照してください :

http://www.microchip.com/mplab

図 1-11: MPLAB X IDE の [START PAGE]



1.10 ウェブサイト

マイクロチップ社は、自社が運営するウェブサイト (www.microchip.com) を通してオンライン サポートを提供しています。このウェブサイトからファイルや情報を簡単に入手できます。詳細は本書の「**サポート**」を参照してください。

1.11 MPLAB X IDE の更新

MPLAB X IDE はユーザの皆様にお使いいただくだけでなく、マイクロチップ社内でも新機能を採用した新型マイクロコントローラ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マイクロチップ社内での使用経験を基に開発しています。新設計のマイクロコントローラ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等) がない限り、その時点での新バージョンへの更新はお薦めしません。新しいプロジェクトの開始時に更新する事をお薦めします。

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となります。

MPLAB X IDE と関連コンポーネントの変更通知サービスをご希望のお客様は、myMICROCHIP Personalized Notification Service(http://www.microchip.com/pcn) の開発ツールのセクションにご登録ください。詳細は本書の「**サポート**」を参照してください。

NOTE:

Chapter 2. 使用前の準備

MPLAB X IDE を使い始める前に、下記の作業が必要です。

- JRE と MPLAB X IDE のインストール
- 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
- ターゲットへの接続 (ハードウェア ツールの場合)
- 言語ツールのインストール
- IDE の起動
- IDE で複数のインスタンスを使う方法

2.1 JRE と MPLAB X IDE のインストール

本章を読み進める前に、Java ランタイム環境 (JRE) と MPLAB X IDE (NetBeans プラットフォームに基づく) を、下記の手順でインストールする必要があります。

1. [Start Page] の [Release Notes] から「Readme for MPLAB X IDE」にアクセスして、ご使用になる MPLAB X IDE のバージョンに対して JRE が適合している事を確認します。
2. 本章の以降の内容に従って作業を進めます。必要に応じて本書を印刷してください (印刷方法は「Help On Help」を参照してください)。
3. MPLAB X IDE を終了し、必要な手順を実施した後に、再起動します。

2.2 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

ツールを正しく動作させるために、USB ドライバをインストールする必要があります。

2.2.1 Mac または Linux OS 用の USB ドライバのインストール

Mac または Linux コンピュータに MPLAB X IDE をインストールする場合、インストーラは自動的に USB ドライバもインストールします。特別な操作は不要です。

2.2.2 Windows® 2000/XP/Vista/7 OS 用の USB ドライバのインストール

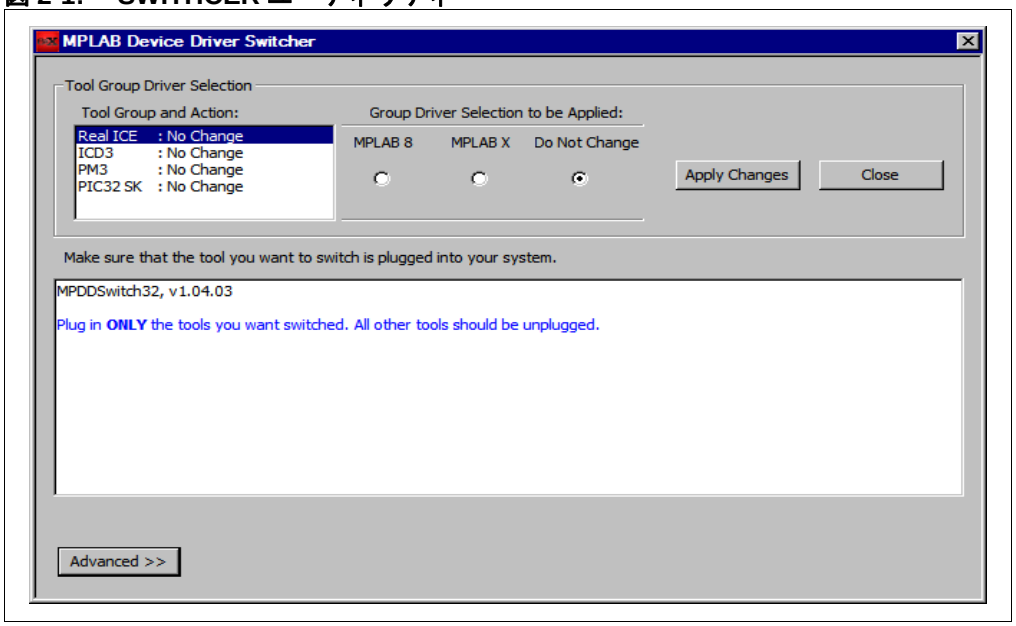
MPLAB X IDE を Windows PC にインストールする場合、以下の手順で USB ドライバをインストールする必要があります。これらの手順は下記のツールを使う場合に必要です。

- MPLAB REAL ICE インサーキット エミュレータ
- MPLAB ICD 3 インサーキット デバッガ
- MPLAB PM3 デバイス プログラマ
- PIC32 スタータキット (Beta 7.10 以前の 32 ビット オペレーティング システムにのみ適用)

PICkit 2 と PICkit 3 および上記以外の MPLAB スタータキットに対しては、以下の操作は不要です。

USB ドライバのインストールと更新には、「MPDDSwitch」という名の GUI アプリケーションを使えます。このアプリケーションは、デスクトップ アイコン [MPLAB Device Driver Switcher] から起動できます。

図 2-1:　SWITHCER ユーティリティ



上記のアプリケーションが動作しない場合、同じディレクトリにあるバッチファイルを使ってドライバをインストールできます。以下に記載する手順に従って、インストール方法を選択してください。

2.2.2.1　管理者モード (WINDOWS 7 OS)

以下の手順では、Switcher 実行ファイルの使用を推奨します。これを Windows 7 で実行するには、管理者モードでログインする必要があります。

「Device Driver Switcher」GUI アプリケーションを管理者として実行するには、実行ファイル (`MPDDSwitch32.exe` または `MPDDSwitch64.exe`) を右クリックして [Run as Administrator] を選択します。

ドライバの切り換えには、まず、この GUI アプリケーションを使う事を推奨します。このアプリケーションで問題が生じた場合にのみ、コマンドライン アプリケーションを使ってください。

コマンドライン アプリケーション (`mchpdds32.exe` または `mchpdds64.exe`) を実行するには、Windows の [スタート] メニューから [スタート]>[全てのプログラム]>[アクセサリ] を選択し、メニュー項目 [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。これにより管理者用コマンドプロンプト画面が開きます。この後に、`ReadMe32.txt` または `ReadMe64.txt` ファイルに記載されている命令を入力してドライバを切り換える事ができます。

ヘルパー バッチファイルを使ってドライバを切り換える場合にも、管理者モードでコマンド プロンプトを開いてから、適切な構文を入力してバッチファイルを実行します。

2.2.2.2　MPLAB IDE V8.XX がシステムにインストールされている場合

4. 必要なツールを PC の USB コネクタに接続します。
5. PC の [デバイス マネージャ] ウィンドウを開きます。例として、Windows XP PC の場合、[マイ コンピュータ] アイコンを右クリックし、[プロパティ] を選択します。[システムのプロパティ] ダイアログ内の **[ハードウェア]** タブを開き、**[デバイス マネージャ]** ボタンをクリックします。
6. 「Microchip Tools」セクションを展開して、接続したツールに現在割り当てられているドライバを表示します。ドライバは、「Microchip *Tool Name*」の書式で表示されます。
7. MPLAB X IDE のインストール先フォルダの中の「Swithcer」フォルダを開きます。このフォルダの既定値パスは下記の通りです。
   a) 32 ビット OS の場合 : `C:\Program Files\Microchip\MPLABX\Switcher`
   b) 64 ビット OS の場合 : `C:\Program Files (x86)\Microchip\MPLABX\Switcher`

8. 「Switcher」フォルダの中のオペレーティング　システムに対応したフォルダ (「32Bit」または「64Bit」) を開きます。
9. その中の MPDDSwitch.exe ファイルを起動します。
10. MPLAB IDE v8 または MPLAB X IDE を既定値とは異なるディレクトリにインストールしている場合、**[詳細]** をクリックして、ドライバファイルの格納場所を指定します。
11. USB ドライバのインストールまたは切り換え方法
    a) 「Tool Group and Action」の下で、ドライバを切り換えるツールをクリックして選択します。
    b) ラジオボタンで「MPLAB 8」または「MPLAB X」のいずれかを選択します。
    c) **[Apply All]** をクリックします。切り換えの進捗状況が大きなテキスト ウィンドウに表示されます。切り換えには若干の時間を要します。
12. GUI によるドライバのインストールに失敗した場合、ReadMe32.txt または ReadMe64.txt ファイルの説明に従ってバッチファイルを実行して、ドライバをインストールしてください。
13. プログラムまたはバッチの実行が完了したら、[デバイス マネージャ] ウィンドウにドライバの名前が表示される事を確認します。ドライバ名は「Microchip WinUSB Device」と表示されるはずです。

MPLAB X IDE 用ドライバをインストールした後は、ドライバを MPLAB IDE v8.xx と MPLAB X IDE 用の間で任意に切り換える事ができます。

2.2.2.3 MPLAB IDE V8.XX がシステムにインストールされていない場合

v8 がインストールされていない場合、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] ウィザードが起動します。

2.2.2.3.1 MPLAB X IDE の自動インストール (推奨)

IDE のインストール中にドライバがプリインストールされます。これらはウィザードを使って自動的にインストールできます。ドライバを手動でインストールする事もできます (次項参照)。

2.2.2.3.2 MPLAB X IDE の手動インストール

不適正なドライバのインストールを防ぐために、ウィンドウザードに従ってドライバをインストールしてください。ドライバの格納場所を示すよう求められた場合、「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 :

```
C:\Program Files\Microchip\MPLAB X IDE\Switcher\32Bit\winusb\x86\
  MCHPWinUSBDevice.inf
```

または

```
C:\Program Files\Microchip\MPLAB X IDE\Switcher\64Bit\winusb\amd64\
  MCHPWinUSBDevice.inf
```

2.2.2.4 ツールの通信の問題

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問題が生じる場合、PCのUSBポートに直接ツールを接続する必要があるかもしれません。これはWinUSB ドライバの既知の問題です。

ツールとの通信で問題が生じた場合、システムにインストールされている WinUSB のバージョンが古い可能性があります。PC のシステムフォルダ (32 ビット OS の場合 : C:\Windows\system32、64 ビット OS の場合 : C:\Windows\SysWOW64) を開き、WinUSB を見つけて名前を変更した後に、ドライバをインストールし直してください。

2.3 ターゲットへの接続 (ハードウェア ツールの場合)

インサーキット デバッガおよびエミュレータを使う場合、下記を参照してハードウェアツールをターゲットに接続してください。

- 開発ツールの設計注意書
- ヘッダの仕様書 (ヘッダを使う場合)
- ツールの文書

デバッグ機能を持たない一般的プログラマを使う場合、接続に関する情報は、そのプログラマの文書を参照してください。

マイクロチップ社のデモボード、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。

2.4 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 (MPASM ツールチェーンと ASM30 ツールチェーン) も一緒にインストールされます。

MPLAB X IDE では、各種の C コンパイラツール パッケージ (コンパイラ、アセンブラ、リンカ等) を使えます。マイクロチップ社ウェブサイト (http://www.microchip.com) からは、無償のコンパイラ (Lite、評価用) と機能を完備したコード最適化コンパイラ (Standard、Pro) を入手できます。

ご使用になるデバイスをサポートしているコンパイラツール パッケージを選択する必要があります。

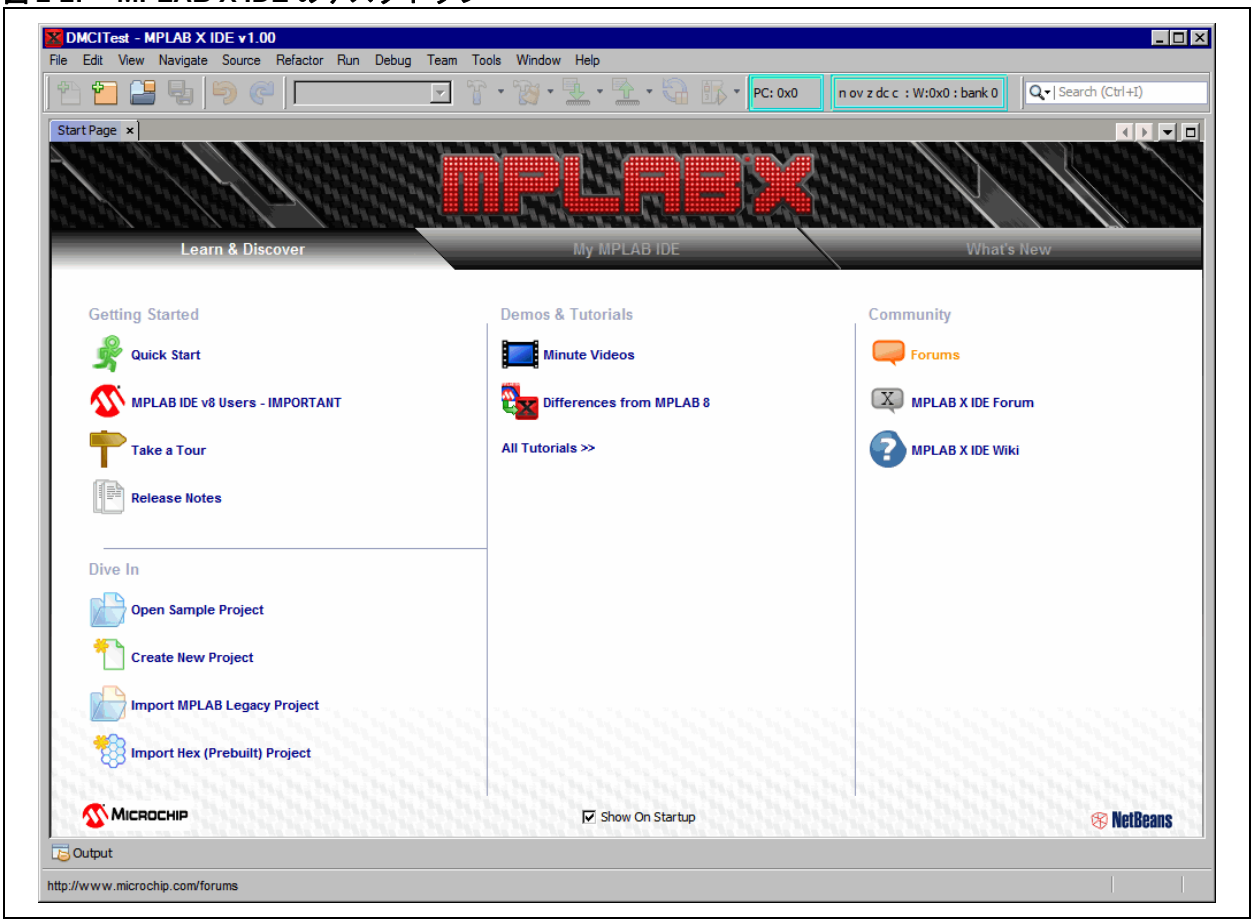
選択したコンパイラをインストールするには、ダウンロード中に表示される指示に従ってください。ダウンロード後は、ダウンロードした ZIP ファイル内の指示に従ってください。

2.5 IDE の起動

[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。

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。

図 2-2: MPLAB X IDE のデスクトップ



[**Start Page**] は、各種のリンクを納めた 3 つのタブで構成されます。各タブの内容を次の表に示します。

表 2-1: [LEARN AND DISCOVER] タブ

Getting Started	
Quick Start	プロジェクトを作成およびセットアップして、すぐに使えるようにします。
MPLAB® IDE v8 Users – IMPORTANT	使用するハードウェア ツール向けのUSBWin デバイスドライバをインストールします。
Take a Tour	MPLAB X IDE の操作の概要を見る事ができます。
MPLAB® X IDE Limitations	このバージョンのサポート / 非サポート情報を見る事ができます。
Dive In	
Open Sample Project	実行可能なサンプル プロジェクトを開きます。
Create New Project	新しい MPLAB X IDE プロジェクトを作成します。初めてプロジェクトを作成する場合、先に「Quick Start」をご覧になる事をお薦めします。
Import Legacy Project	既存のMPLAB IDE v8 プロジェクトをインポートします。インポートしたプロジェクトで作業を始める前に、「Quick Start」をご覧になる事をお薦めします。
New (Hex) Prebuilt Project	別のツール用にプリビルドしたプロジェクトからHEX ファイルをインポートします。
Demos and Tutorials	
Minute Videos	MPLAB X IDE の操作方法を紹介する短いビデオをご覧になれます。
Difference from MPLAB v8	MPLAB X IDE と MPLAB IDE v8 の相違点をご覧になれます。
All Tutorials	利用可能な全てのチュートリアルを表示します。
Community	
Forums	Microchip 社のフォーラム ウェブページを開きます。
MPLAB X IDE Forum	MPLAB X IDE フォーラムに登録します。
MPLAB X IDE Wiki	MPLAB X IDE 開発者向けヘルプセンターを開きます。

表 2-2: [MY MPLAB X IDE] タブ

Recent Projects	
MyProject.c:	最近開いたプロジェクトの一覧を表示します。
Extend MPLAB	
Selecting Simple or Full Featured Menus	MPLAB X IDE を初めて起動した時は、一部の項目を省略したメニューが表示されますが、全ての項目を含むメニューを選択する事もできます。
Install More Plug-Ins	プラグイン用のダイアログを開きます。
Notes and Newsletters	
ANxxxx, TBxxxx	お薦めのアプリケーション ノートと技術概要を表示します。
microSOLUTIONS E-newsletter	お薦めのニュースレターを表示します。
All App Notes/ Newsletters	全てのアプリケーション ノートとニュースレターを表示します。
References and Featured Links	
Data Sheets, etc.	リンクをクリックしてデータシート等の参考文書を閲覧できます。

表 2-3: [WHAT'S NEW] タブ

Data Sheets and Errata	
Data Sheet, Silicon Errata	お薦めのデータシートとエラッタの一覧を表示します。[All Data Sheets] または [All Errata] をクリックすると、全てのデータシートまたはエラッタを含む一覧を表示します。
Reference Manuals and Programming Spec	
Family Reference Manual, Device Programming Spec	お薦めのリファレンス マニュアルとプログラミング仕様の一覧を表示します。[All Reference Manuals] または [All Programming Specs] をクリックすると、これらの全ての文書を含む一覧を表示します。
Recently Released Software	
Source code	マイクロチップ社のデバイスを使った開発をサポートする最新のソフトウェアの一覧を表示します。[All Recently Released Software] をクリックすると、これらの全てのソフトウェアを含む一覧を表示します。
Product and Corporate News	
New stuff, new news	お薦めのマイクロチップ社製品とニュースを表示します。[All News] をクリックすると、これらの全ての文書を含む一覧を表示します。

2.6 IDE で複数のインスタンスを使う方法

MPLAB X IDE は、各インスタンスに対して、独自のユーザディレクトリを持つ事を要求します。従って、あるインスタンスに対する設定や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」オプションでディレクトリを指定して IDE を起動する必要があります。

2.6.1 Windows OS

「--userdir」オプションを使ってショートカットを作成します。以下に例を示します。

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、インストールした MPLAB X IDE の実行ファイルを指定します (下記は既定値)。
   ```
   C:\Program Files\Microchip\MPLABX\mplab_ide\bin\
    mplab_ide.exe
   ```
3. 行の末尾に下記を入力します。
   ```
   --userdir "C:\Documents and Settings\MyFiles\
     ApplicationData\.mplab_ide\dev\beta7Instance2"
   ```
4. **[OK]** をクリックします。

2.6.2 Linux OS

インストールしたバージョンを、パラメータを何も指定せずに実行 (デスクトップアイコンをクリックして実行) した場合、ユーザ ディレクトリとして「$(HOME)/.mplab_ide」を使います。ユーザ ディレクトリを変更するには、シェル スクリプト「$InstallationDir/mplab_ide/bin/mplab_ide」に引数「--userid *anydir*」を指定して実行します。下記は、MPLAB X IDE を 2 つの異なるインスタンスで実行する場合の例です。

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

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。

2.6.3 Mac OS

シェルウィンドウを開き、下記のコマンド行をタイプ入力して、インストールした MPLAB X IDE (例 : Beta 7.12) を別のユーザ ディレクトリで実行します。

```
$/bin/sh /Applications/microchip/mplabx/712/mplab_ide.app/Contents/
 Resources/mplab_ide/bin/mplab_ide --userdir "${HOME}/Library/
 Application Support/mplab_ide/dev/beta7.12”
```

Chapter 3. チュートリアル

チュートリアルは、例を示しながら MPLAB X IDE の使い方をガイドします。

アルゴリズムとソフトウェアのセットアップ

- チュートリアルで使うツール
- インストールとセットアップ

プロジェクトの作成とセットアップ

- 新規プロジェクトの作成
- プロジェクト作成後のデスクトップ画面
- 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設定
- 言語ツールのパスの指定
- 既存ファイルをプロジェクトに追加する
- エディタの使い方
- コンフィグレーション ビット

コードの実行とデバッグ

- プロジェクトのビルド
- コードの実行
- コードのデバッグ実行
- ブレークポイントによるプログラム実行の制御
- コードのステップ実行
- シンボル値の変化の観察
- 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
- デバイスのプログラミング

3.1 チュートリアルで使うツール

チュートリアルでは下表の製品を使います。

ツール	ウェブページ	注文番号
MPLAB® X IDE	http://www.microchip.com/mplabx	N/A
PIC32 MCU 向け MPLAB® C コンパイラ *	http://www.microchip.com/c32	SW006015 (標準バージョン)
MPLAB® REAL ICE™ インサーキット エミュレータ	http://www.microchip.com/realice	DV244005
Explorer 16 開発ボード	http://www.microchip.com/explorer16	DM240001
PIC32MX360F512L PIM	http://www.microchipdirect.com/productsearch.aspx?Keywords=MA320001	MA320001
* このコンパイラの評価用 (無償) バージョンは MPLAB X IDE に付属しています。チュートリアルでは、この評価用バージョンを使っています。もちろん、標準バージョンもチュートリアルでご使用になれます。		

3.2 インストールとセットアップ

Chapter 2.「使用前の準備」に従って MPLAB X IDE をインストールし、エミュレータをセットアップし (USB ドライバをインストールしてターゲットに接続)、32 ビット言語ツールをインストールしてください。それらを済ませた後に MPLAB X IDE を起動して、チュートリアルを始めます。

3.3 新規プロジェクトの作成

MPLAB X IDE はプロジェクトを基本とするため、アプリケーション用のプロジェクトをセットアップする必要があります。

新規プロジェクトは下記のいずれかの方法で作成できます。

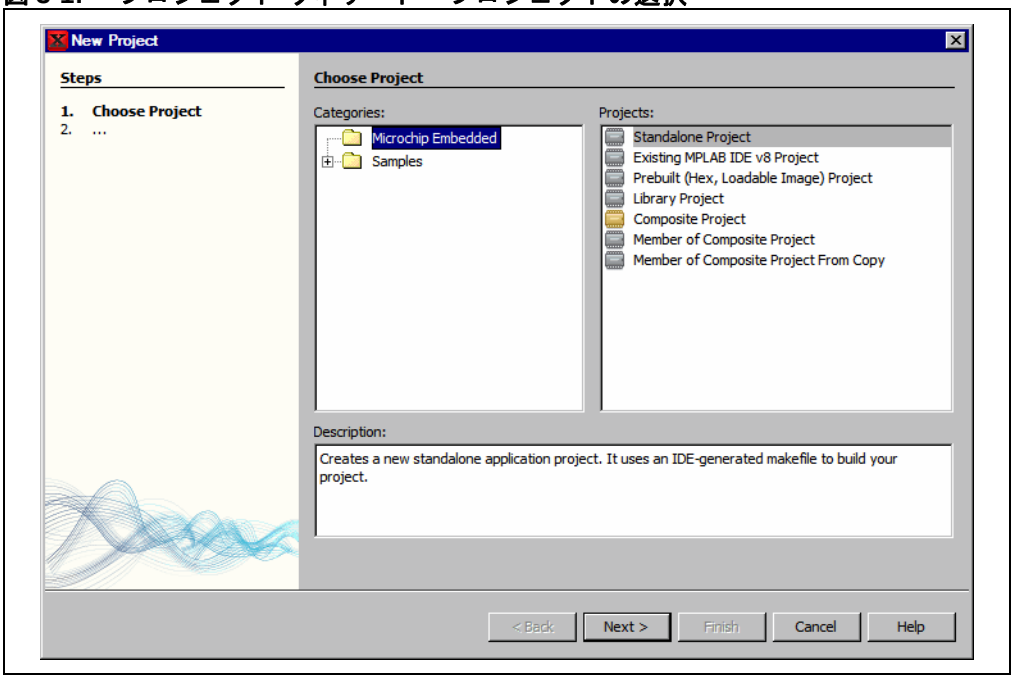
- **[Start Page]** の **[Learn & Discover]** タブで、「Dive In」の下の「Create New Project」を選択する
- [File]>[New Project] を選択する (または Ctrl+Shift+N)

[New Project] ウィザードが起動し、新規プロジェクトのセットアップ手順をガイドします。

Step 1 では、プロジェクト カテゴリを選ぶよう指示されます。これは NetBeans のダイアログです。マイクロチップ社製品を使うには「Microchip Embedded」を選択する必要があります。次にプロジェクトのタイプを選択します。本チュートリアルでは「Stand-alone Project」を選びます。

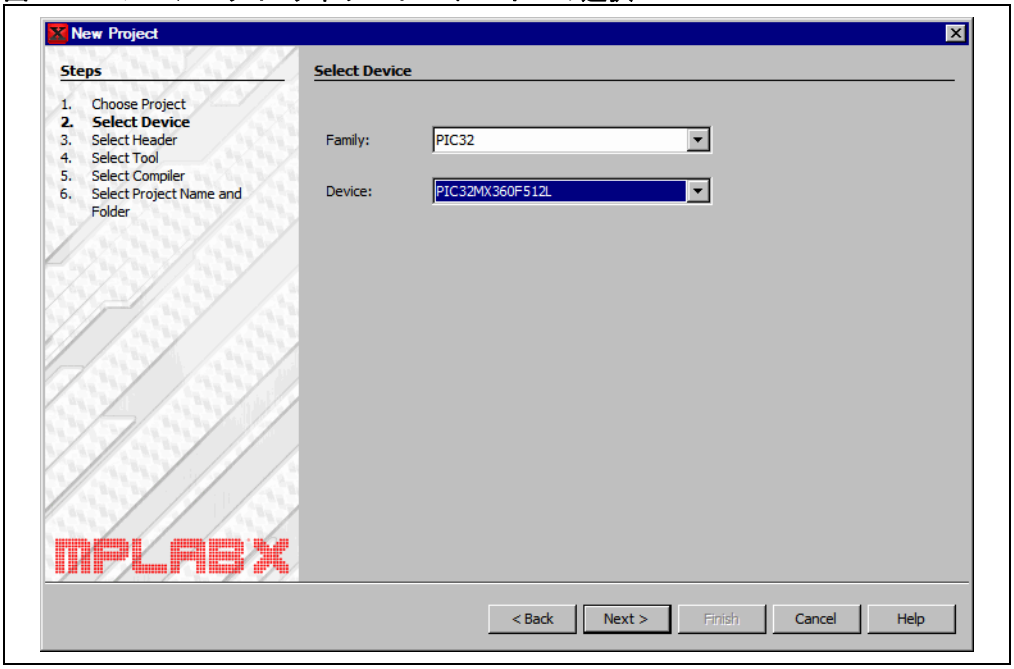
[Next>] をクリックして次のダイアログへ進みます。

図 3-1:　プロジェクト ウィザード – プロジェクトの選択



Step 2 は MPLAB X IDE のダイアログです。このため Step 1 のダイアログとは外観が異なります。ここでは、使用するデバイス (PIC32MX360F512L) を選択してから **[Next>]** をクリックします。

図 3-2:　プロジェクト ウィザード – デバイスの選択



Step 3 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。PIC32MX360F512L デバイスはヘッダを備えないため、MPLAB X IDE は自動的にこのステップをスキップします。

Step 4 ではツールを選択します。選択したデバイスをサポートしているツールは、ツール名の左側の丸マークの色で識別できます。これは MPLAB IDE v8 の [Device Selection] ダイアログと同じです。マークの色分けは下記の通りです。

- 緑 – 完全サポート
- 黄 – ベータ版サポート
- 赤 – 未サポート

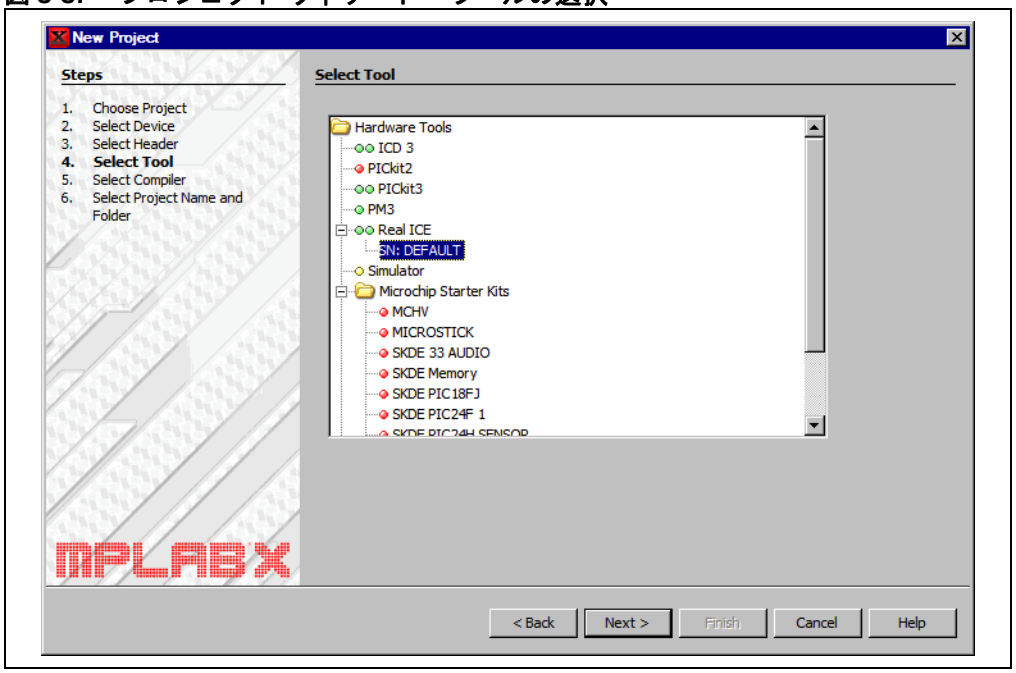
色を識別できない場合、マウスカーソルをマークの上に置くと、サポートに関する情報を表示できます。

2 つあるマークの左側のマークはデバッガのサポートを示し、右側のマークはプログラマのサポートを示します。

ハードウェア ツールを PC に接続している場合、それらのツールの下にシリアル番号 (SN) が表示されます。これにより、接続した複数のハードウェア ツールから必要なツールを識別できます。

ツールを選択してから **[Next>]** をクリックします。

図 3-3: プロジェクト ウィザード – ツールの選択

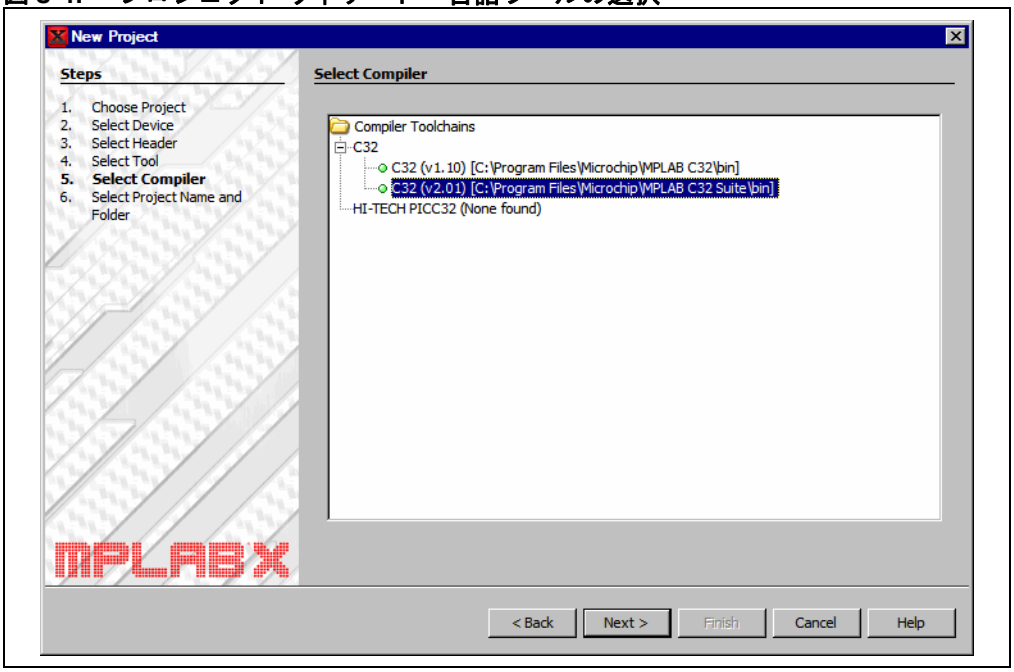


Step 5 では、言語ツール (C コンパイラまたはアセンブラ) を選択します。ここでも、コンパイラ名の左側のマークの色で、選択したデバイスに対するサポートレベルを識別できます。マウスカーソルをマークの上に置くと、サポートに関する説明を表示できます。

インストールされている各ツール名の右側には、バージョンとインストール先のパスも表示されます。これにより、インストールされている複数の言語ツールから必要なツールを選択できます。

ツールを選択してから **[Next>]** をクリックします。

図 3-4:　プロジェクト ウィザード – 言語ツールの選択



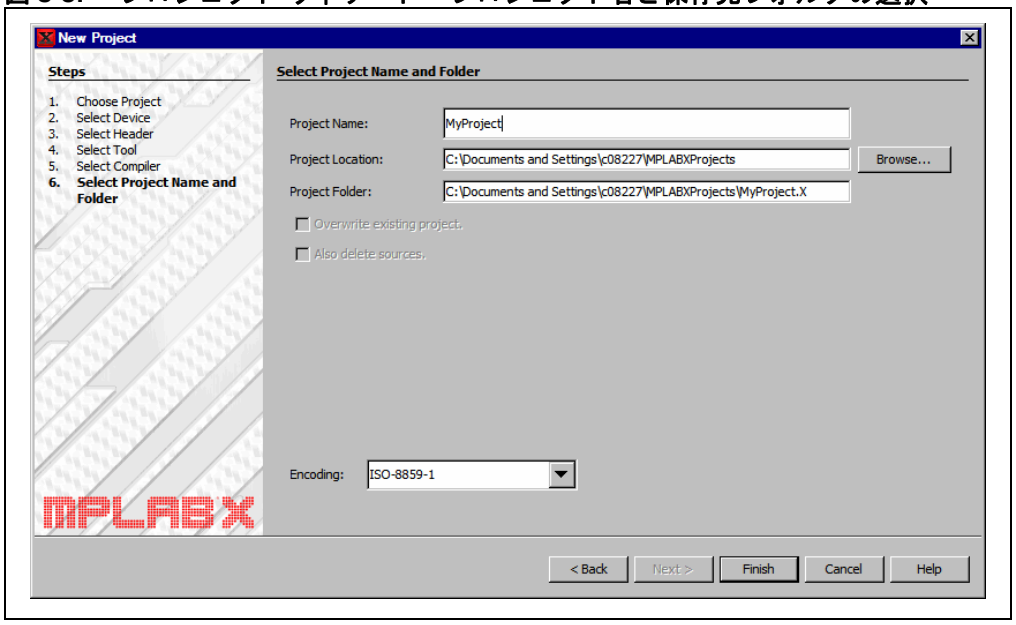
Step 6 では、プロジェクト名とプロジェクトの保存場所を選択します。プロジェクト名として「MyXProject」を入力します。既定値では、プロジェクトは下記のパスに保存されます。

- Windows XP – C:\Documents and Settings\UserName\MPLABXProject
- Windows 7 – C:\Users\UserName\MPLABXProjects
- Linux – /home/UserName/MPLABXProjects
- Mac – /Users/UserName/MPLABXProjects

「Project Location」の右方の **[Browse...]** ボタンを使って上記以外の保存先を指定する事もできます。

上記を済ませた後に、**[Finish]** をクリックすると、新規プロジェクトの作成が完了します。

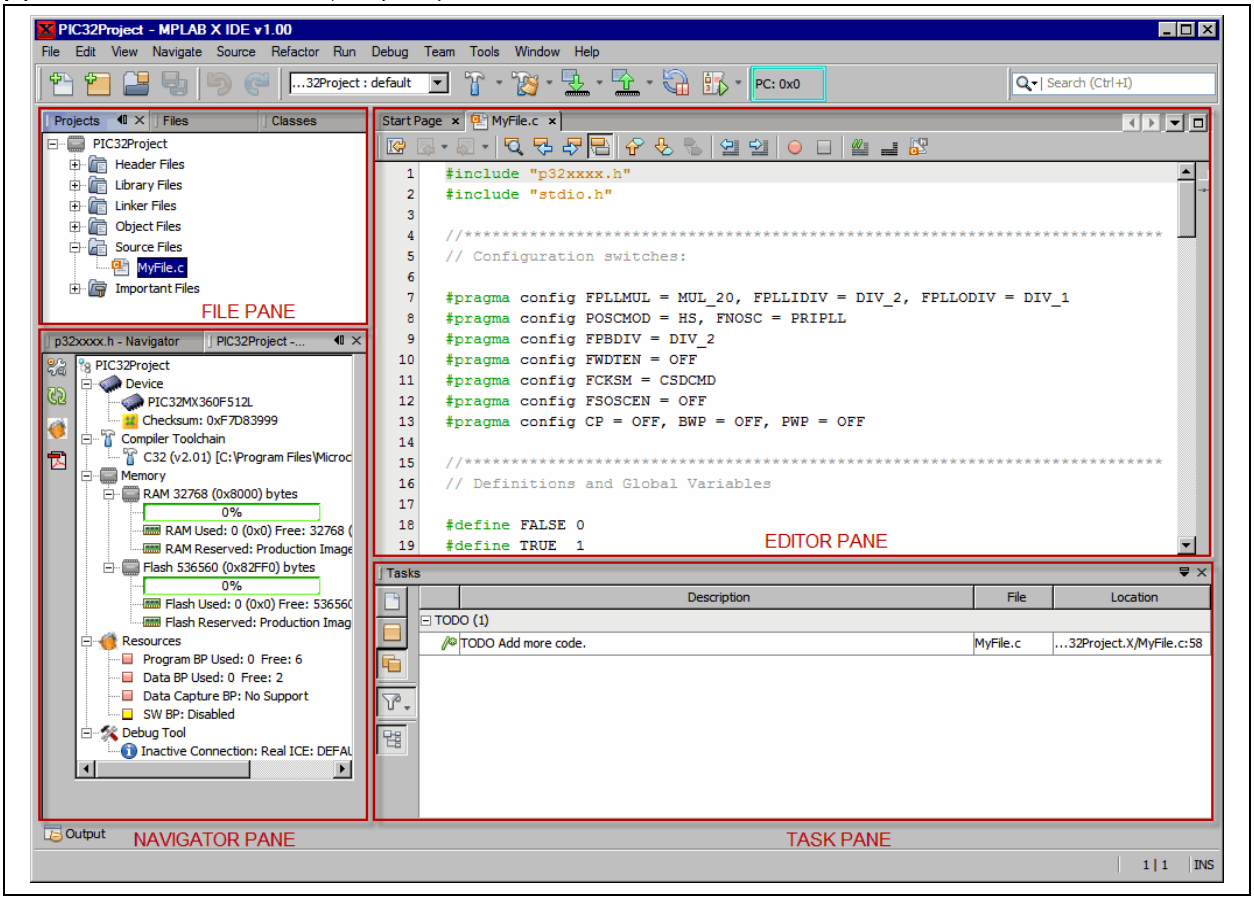
図 3-5:　プロジェクト ウィザード – プロジェクト名と保存先フォルダの選択



3.4 プロジェクト作成後のデスクトップ画面

プロジェクトの作成が完了すると、IDE 内に各種のウィンドウ枠が開きます。

図 3-6: MPLAB X IDE のデスクトップ



- [File] 枠 – ファイルに関連する 4 つのタブを備えます。[Projects] タブはプロジェクト ツリー、[Files] タブはプロジェクト ファイル、[Classes] タブはコード内の全てのクラス、[Services] タブはコードの開発に使える全てのサービスを表示します。
- [Navigator]枠 – 選択されているファイルまたはプロジェクトに関する情報を表示します。プロジェクトが選択されている場合はプロジェクトの詳細を示すプロジェクト環境を表示し、ファイルが選択されている場合はシンボルと変数を表示します。
- [Editor]枠 – 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] も表示できます。
- [Task] 枠 – 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。

[File] 枠内で、いずれかの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] 枠内の [Start Page] タブの隣のタブに表示されます。タブを閉じるには、タブ名の右横の「x」をクリックします。

[File] 枠の [Projects] タブ内でプロジェクト名を右クリックすると、ポップアップ (コンテクスト) メニューが開きます。プロジェクトのサブフォルダに対しても同様の操作が可能です。

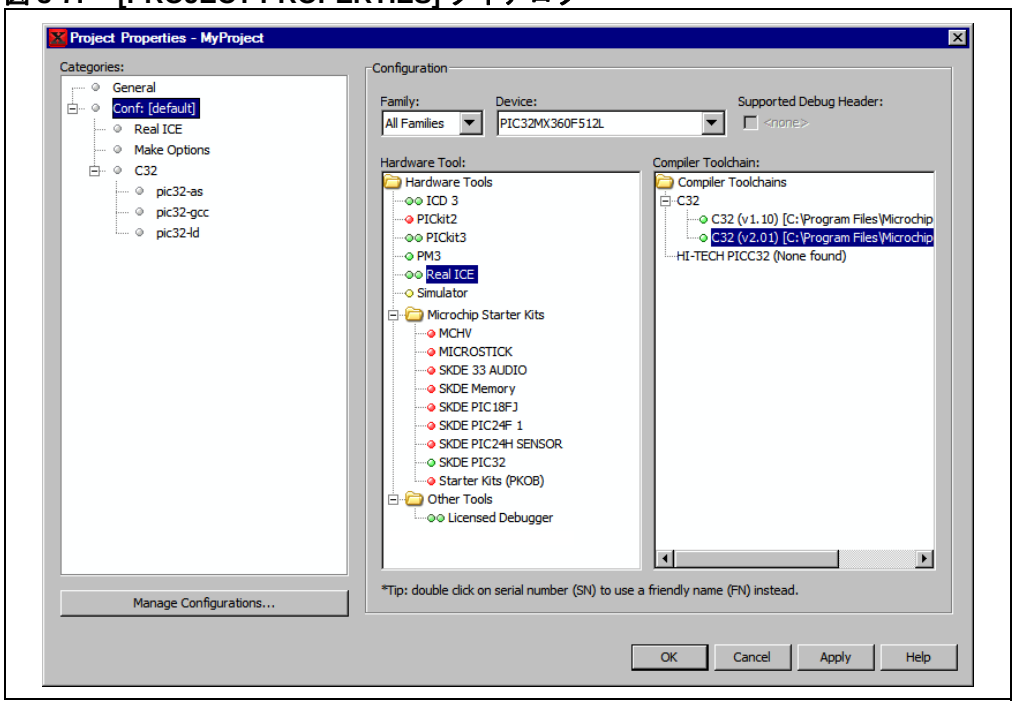
3.5 プロジェクト プロパティの表示と変更

プロジェクトを作成した後に、[Project Properties] ダイアログにプロジェクト プロパティを表示して編集できます。このダイアログは下記のいずれかの方法で開く事ができます。

- [File] 枠の [Projects] タブ内でプロジェクト名を右クリックし、「Properties」を選択する
- [File] 枠の [Projects] タブ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] を選択する

開いたダイアログ内で、「Conf:[default]」カテゴリをクリックすると、標準的なプロジェクト コンフィグレーション (プロジェクト デバイス、関連するデバッガ / プログラマ ツール、言語ツール等) が表示されます。前章までに説明したセットアップ手順に誤りがなければ、チュートリアルでこれらのコンフィグレーションを変更する必要はありません。**[Apply]** をクリックして、このダイアログの内容を適用します。

図 3-7: [PROJECT PROPERTIES] ダイアログ



3.6 デバッガ、プログラマ、言語ツールのオプション設定

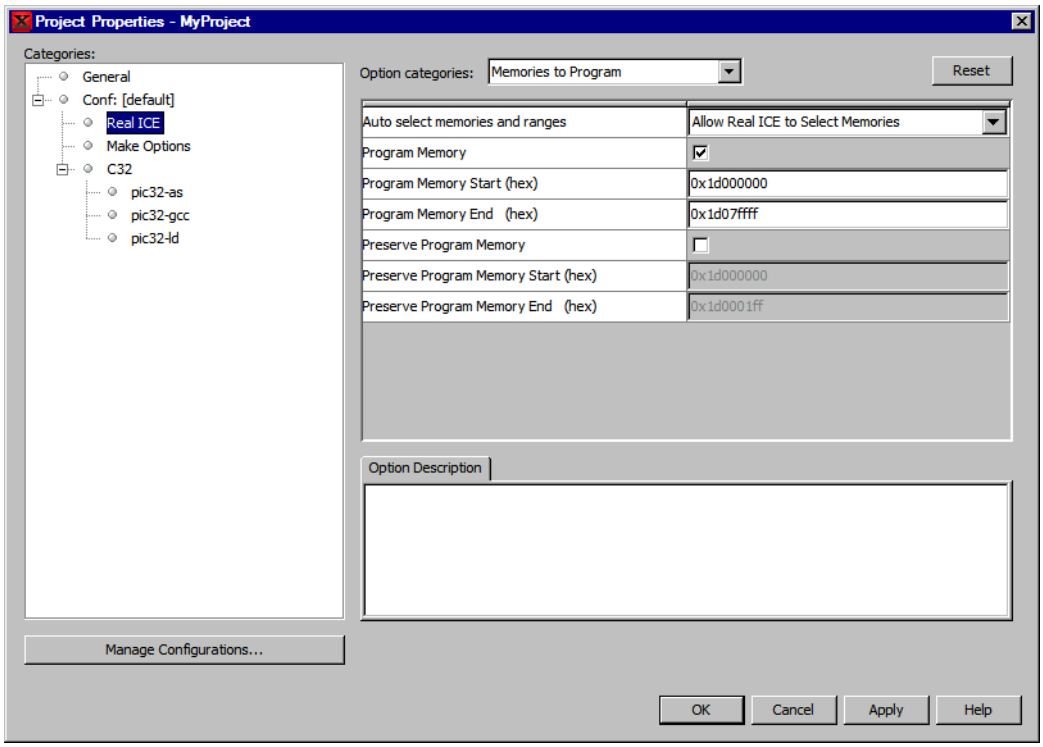
[Project Properties] ダイアログでは、ツールのオプションも設定します。

デバッガ / プログラマ ツール オプションをセットアップまたは変更する方法

- 「REAL ICE」をクリックして、関連するセットアップ オプションを表示します。これらのオプションの意味については、エミュレータの文書を参照してください。

チュートリアルでは何も変更しないでください。

図 3-8: ツールのセットアップ ページ

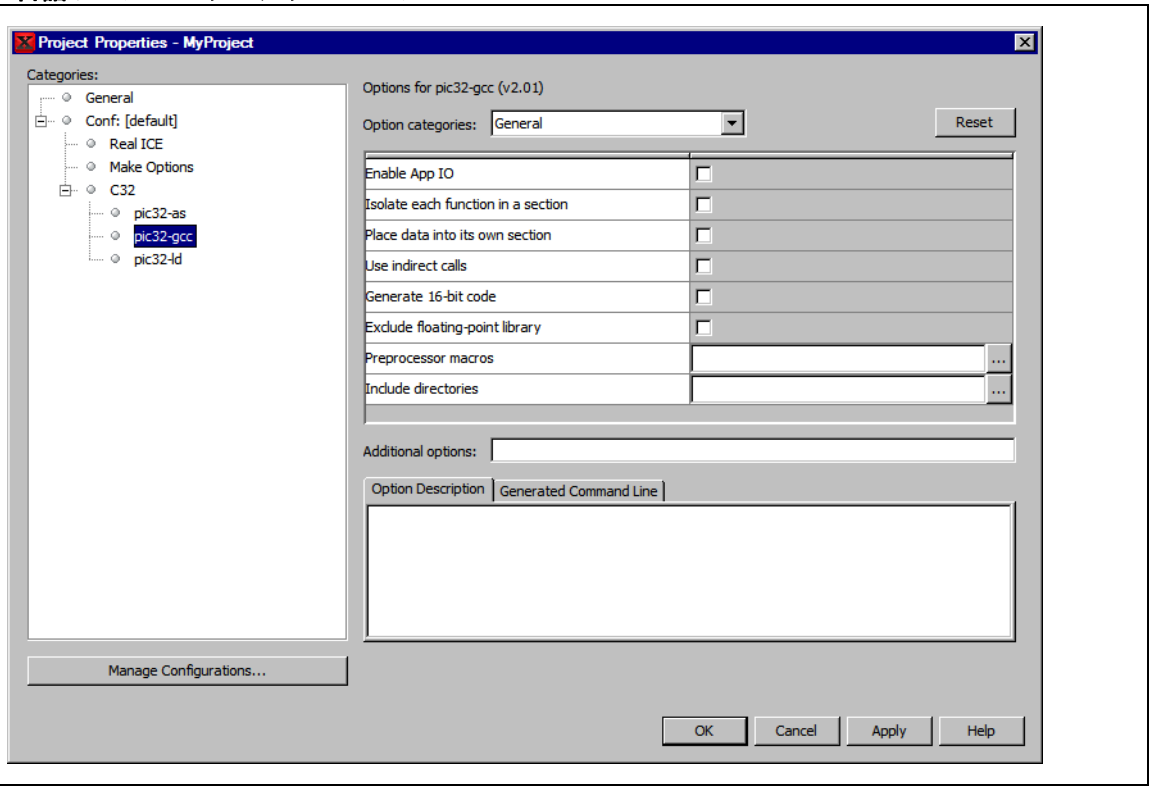


言語ツール オプションをセットアップまたは変更する方法

- 使用する言語ツールをクリックして、関連するセットアップ オプションを表示します。これらのオプションの意味については、言語ツールの文書を参照してください。

チュートリアルでは何も変更しないでください。

図 3-9:　言語ツールのセットアップ ページ



3.7 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または変更する方法

- **Mac OS の場合 :**
  メイン メニューバーから [mplab_ide]>[Preferences]>[Embedded]>[Build Tools] を選択してビルドツールにアクセスします。
- **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します。

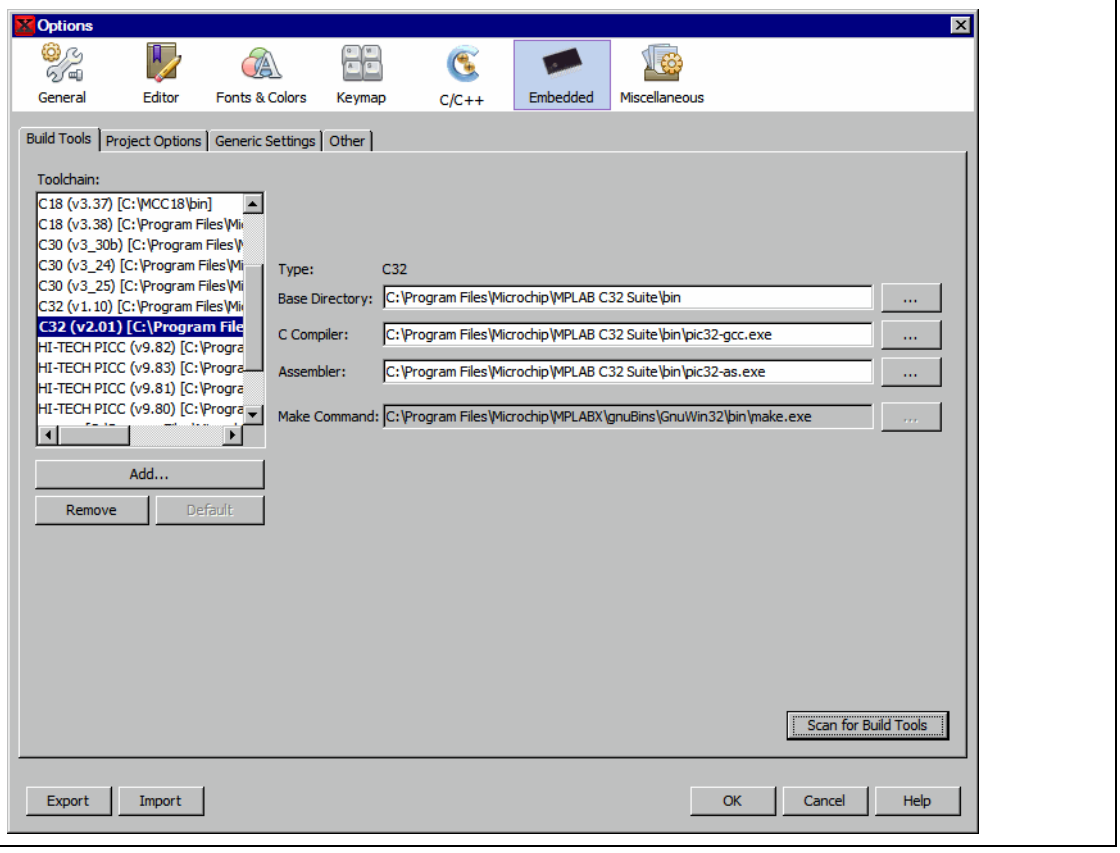
ウィンドウには、インストールされている全てのツールチェーンが自動的に表示されます。インストール済みのツールが表示されない場合、下記を試してください。

- **[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格納したディレクトリ (ベース ディレクトリ) 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ディレクトリ内の「bin」サブディレクトリ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。

チュートリアルでは、C32 ツールチェーンが選択されている事を確認してください。

図 3-10: 言語ツール (コンパイラ) のパス



3.8 既存ファイルをプロジェクトに追加する

本チュートリアルで使うサンプルコードは、下記の方法で入手できます。

- マイクロチップ社ウェブサイト内の Explorer 16 開発ボードのウェブページに (http://www.microchip.com/explorer16) アクセスします。
- 「PIC32 Explorer 16 LED Example Application」をクリックして、サンプルコードを格納した ZIP ファイルをダウンロードします。
- ダウンロードした ZIP ファイルを解凍します。
- ファイル `led_message.c` をプロジェクト ディレクトリ「`MyXProject.X`」に移動します。

下記のいずれかの方法により、既存ファイルをプロジェクトに追加します。

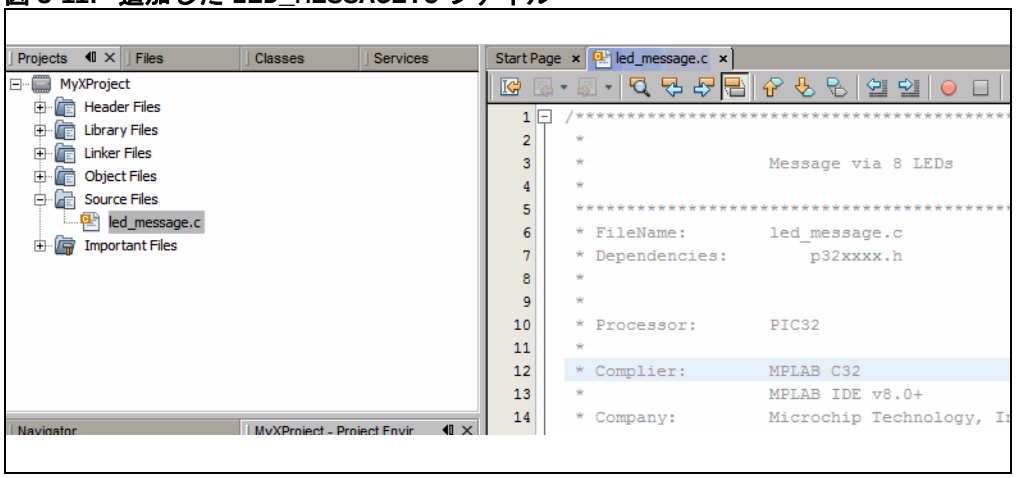
- [Projects] タブ内のプロジェクトを右クリックし、「Add Existing Item」を選択する
- [Project/File] ウィンドウ内の論理フォルダ (例 : Source Files) を右クリックし、「Add Existing Item」を選択する

ファイルを追加する際に、パスの指定方法を下記から選択できます。

- 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 (チュートリアルでは、これを選択)。
- Absolute – ファイルのパスを絶対パスで指定します。

このファイルは、[File] 枠内のプロジェクトの下に表示されるとともに、[Editor] 枠にこのファイルと同名のタブが開きます。

図 3-11: 追加した LED_MESSAGE.C ファイル



led_message.c のコード：

```
/********************************************************************
 *
 *                  Message via 8 LEDs
 *
 ********************************************************************
 * FileName:        led_message.c
 * Dependencies:p32xxxx.h
 *
 *
 * Processor:       PIC32
 *
 * Complier:        MPLAB C32
 *                  MPLAB IDE v8.0+
 * Company:         Microchip Technology, Inc.
 *
 * Software License Agreement
 *
 * The software supplied herewith by Microchip Technology Incorporated
 * (the “Company”) for its PIC32 Microcontroller is intended
 * and supplied to you, the Company’s customer, for use solely and
 * exclusively on Microchip PIC32 Microcontroller products.
 * The software is owned by the Company and/or its supplier, and is
 * protected under applicable copyright laws.All rights are reserved.
 * Any use in violation of the foregoing restrictions may subject the
 * user to criminal sanctions under applicable laws, as well as to
 * civil liability for the breach of the terms and conditions of this
 * license.
 *
 * THIS SOFTWARE IS PROVIDED IN AN “AS IS” CONDITION.NO WARRANTIES,
 * WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED
 * TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 * PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.THE COMPANY SHALL NOT,
 * IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
 * CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 *
 * $Id: led_message.c 5898 2007-10-23 19:39:48Z rajbhartin $
 *
 ********************************************************************/

/*
** Message in a bottle
**
** Explorer16 version (long delays)
**
** Thanks to Lucio DiJasio for letting us use this example.
**
** Run this example on Explorer-16 board with PIC32MX PIM.
** Hold the board vertically from the PICtail connector size
** and wave the board back-and-forth to see message "HELLO" on LEDs
*/

#include <p32xxxx.h>

// Config settings
// POSCMOD = HS, FNOSC = PRIPLL, FWDTEN = OFF
// PLLIDIV = DIV_2, PLLMUL = MUL_16
// PBDIV = 8 (default)
// Main clock = 8MHz /2 * 16    = 64MHz
// Peripheral clock = 64MHz /8  =  8MHz
```

```
// Configuration Bit settings
// SYSCLK = 64 MHz (8MHz Crystal/ FPLLIDIV * FPLLMUL / FPLLODIV)
// PBCLK = 8 MHz
// Primary Osc w/PLL (XT+,HS+,EC+PLL)
// WDT OFF
// Other options are don't care
//
#pragma config FPLLMUL = MUL_16, FPLLIDIV = DIV_2, FPLLODIV = DIV_1,
FWDTEN = OFF
#pragma config POSCMOD = HS, FNOSC = PRIPLL, FPBDIV = DIV_8



// 1. define timing constant
#define SHORT_DELAY (50*8)
#define LONG_DELAY(400*8)

// 2. declare and initialize an array with the message bitmap
char bitmap[30] = {
    0xff,// H
    0x08,
    0x08,
    0xff,
    0,
    0,
    0xff,// E
    0x89,
    0x89,
    0x81,
    0,
    0,
    0xff,// L
    0x80,
    0x80,
    0x80,
    0,
    0,
    0xff,// L
    0x80,
    0x80,
    0x80,
    0,
    0,
    0x7e,// O
    0x81,
    0x81,
    0x7e,
    0,
    0
    };

// 3. the main program
main()
{
    // disable JTAG port
    DDPCONbits.JTAGEN = 0;

    // 3.1 variable declarations
    int i;          // i will serve as the index
```

```
    // 3.2 initialization
    TRISA = 0;      // all PORTA as output
    T1CON = 0x8030; // TMR1 on, prescale 1:256 PB

    // 3.3 the main loop
    while( 1)
    {
        // 3.3.1 display loop, hand moving to the right
        for( i=0; i<30; i++)
        {// 3.3.1.1 update the LEDs
            PORTA = bitmap[i];

            // 3.3.1.2 short pause
            TMR1 = 0;
            while ( TMR1 < SHORT_DELAY)
            {
            }
        } // for i

        // 3.3.2 long pause, hand moving back to the left
        PORTA = 0;      // turn LEDs off
        TMR1 = 0;
        while ( TMR1 < LONG_DELAY)
        {
        }
    } // main loop
} // main
```

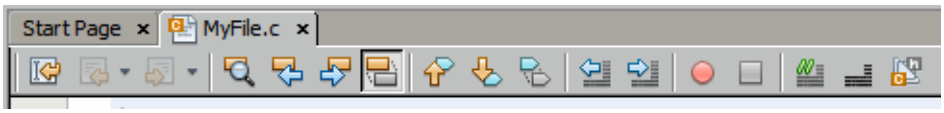
3.9 エディタの使い方

このサンプルコードを編集する必要はありませんが、実際のアプリケーションのコードの編集には NetBeans エディタを使います。このエディタに関する一般情報は、NetBeans ヘルプトピック ([IDE Basics] >[Basic File Features]) を参照してください。エディタに関連する C コンパイラ情報は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[C/C++/Fortran Project Basics]>[Navigating and Editing C/C++/Fortran Source Files]) を参照してください。エディタ機能の一覧は、**6.3「NetBeans™ エディタ」**に記載しています。

エディタの各種機能は下記から利用できます。

- [Edit] メニュー (**9.2.2「[Edit] メニュー」**参照)
- 各ファイルのエディタ ウィンドウの上にあるエディタ用ツールバー
- 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

図 3-12: エディタ用ツールバー



3.10 コンフィグレーション ビット

このサンプルコードでは、コンフィグレーション ビットを設定済みです。コードでは必ずコンフィグレーション ビットを設定する必要があります。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概要は、**Chapter 12.「コンフィグレーション設定の要約」**に記載しています。

コンフィグレーション ビットの設定は、デバッグ実行中に [Configuration Bits] ウィンドウ ([Window]>[PIC Memory Views]>[Configuration Bits]) を使って一時的に変更できます。

3.11 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、実行および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しておきたい場合もあります。

プロジェクトのビルド方法

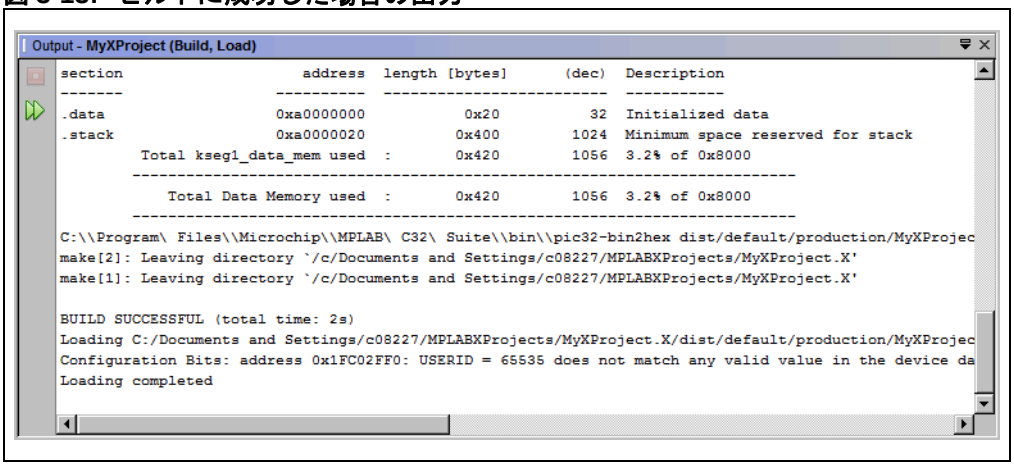
- [Projects] タブ内のプロジェクト名の上で右クリックして、「Build」を選択する (あるいは「Clean and Build」を選択して中間的に生成されたファイルを削除してからビルドする事も可能)
- ツールバーの [Build Project] または [Clean and Build Project] アイコンをクリックする

[Build] アイコン

[Clean and Build] アイコン

ビルドの進捗状況は、[Output] ウィンドウ (デスクトップの右下隅) に表示されます。本チュートリアルのコードは問題なくビルドされるはずです。

図 3-13: ビルドに成功した場合の出力



チェックサム情報の表示方法

- ビルド後のチェックサムは [Dashboard] ウィンドウに表示されます。このウィンドウが開いていない場合、[Window]>[Dashboard] を選択して開きます。

3.12 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。プログラムを実行するには、[Make and Program Device Project] アイコンをクリックするか [Run]>[Run Project] を選択します。

[Make and Program Device Project] アイコン

プログラムを実行すると、デモボード上のランプが点滅します。ボードを左右に振ると「Hello」という文字が浮かび上がります。

実行の進捗状況も [Output] ウィンドウに表示されます。

[Hold in Reset] ボタンを使うと、デバイスをリセットと実行の間で交互に切り換える事ができます。

[Hold in Reset] アイコン

必要に応じて、[Run Project] アイコンをツールバーに追加する事ができます ([View]>[Toolbars]>[Customize])。

[Run] アイコン

3.13 コードのデバッグ実行

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、アプリケーションを開発する場合、コードをデバッグする必要があります。

チュートリアルのコードをデバッグ実行するには、[Debug Project] アイコンをクリックするか、[Debug]>[Debug Project] または [Debug]>[Step Into] を選択して、デバッグ セッションを始めます。

[Debug Run] アイコン

デバッグ実行の進捗状況も [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を一時停止する方法

- [Pause] アイコンをクリックするか、[Debug]>[Pause] を選択すると、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か、[Debug]>[Continue] を選択すると、一時停止していた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] アイコンをクリックするか、[Debug]>[Finish Debugger Session] を選択すると、プログラムの実行が終了します。

C コード プロジェクトのデバッグに関する詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Debugging C/C++/Fortran Applications with gdb]) を参照してください。

通常の実行とデバッグ実行の違いは、次の **3.14「ブレークポイントによるプログラム実行の制御」**から始まるデバッグ機能の説明を読むとよくわかります。

3.14 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。

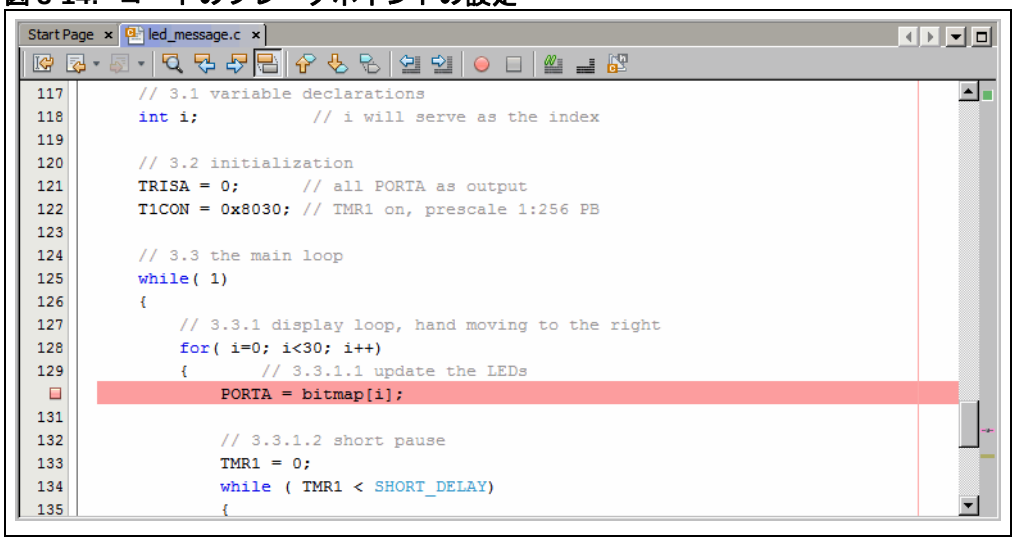
例として、コードの下記の行にブレークポイントを設定します。

```
PORTA = bitmap[i];
```

下記のいずれかの方法により、コードの 1 行にブレークポイントを設定できます。

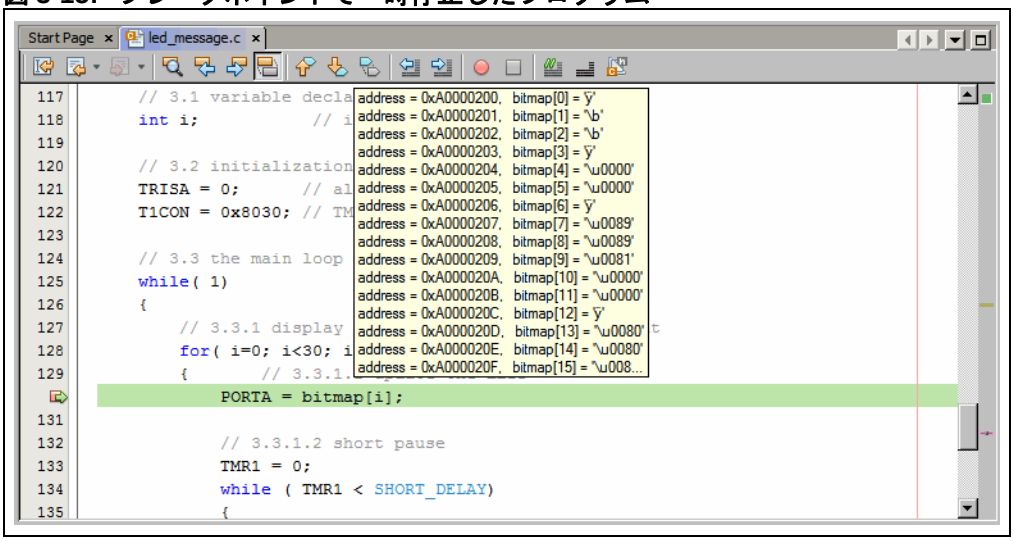
- ソースエディタ内の該当する行の左側の余白をクリックする
- Ctrl-F8 を押す

図 3-14: コードのブレークポイントの設定



ブレークポイントを設定した後に、チュートリアルのプログラムを再度デバッグ実行すると、プログラムはブレークポイントの位置で一時停止します。`bitmap[]` 変数の上にマウスカーソルを置くと、その値が表示されます。

図 3-15: ブレークポイントで一時停止したプログラム



下記のいずれかの方法により、設定したブレークポイントをクリアできます。

- ブレークポイントの設定と同じ操作を繰り返す
- [Debug]>[Toggle Breakpoint] を選択する

ブレークポイントの詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Debugging C/C++/Fortran Applications with gdb]>[Setting C/C++/Fortran Breakpoints]) を参照してください。

3.15 コードのステップ実行

[Debug] メニューおよび [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の直後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、下記の方法で行えます。

- Step Over – プログラムの 1 ソース行を実行します。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 1 ソース行を実行します。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 1 ソース行を実行します。関数をステップ実行中である場合、関数の残りを全て実行した後に、呼び出し元のルーチンに戻って停止します。
- 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。
- Animate – 1 ステップずつ画面に表示しているレジスタの値を更新しながらプログラムを実行します。Animate は通常の実行よりも実行速度が遅くなりますが、[Special Function Register] ウィンドウまたは [Watch] ウィンドウでレジスタ値の変化を観察できます。

ステップ実行の詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Debugging C/C++/Fortran Applications with gdb]>[C/C++/Fortran Debugging Sessions]>[Stepping Through Your C/C++/Fortran Program]) を参照してください。

3.16 シンボル値の変化の観察

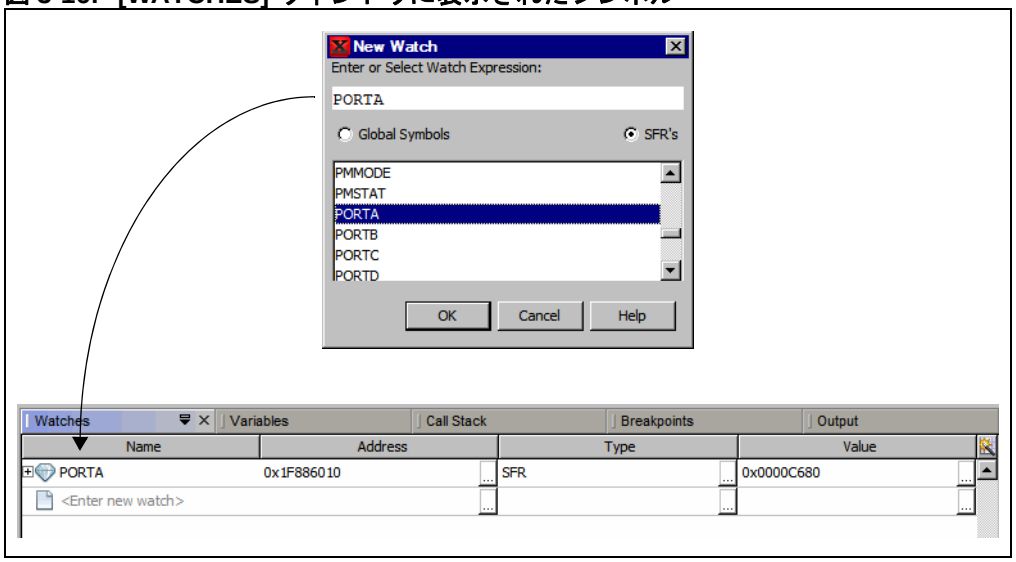
選択したシンボルの値の変化を [Watches] ウィンドウで観察します。これらの値がプログラム実行中に期待通りに変化するかどうかを確認する事により、コードのデバッグに役立てる事ができます。

ウォッチの作成方法

1. [Debug]>[New Watch] を選択します。
2. ウォッチ式 (この例では「PORTA」) 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。

別の方法として、エディタ内でシンボル名または SFR を選択し、これを [Watches] ウィンドウにドラッグ & ドロップしてウォッチを作成する事もできます。あるいは、エディタ内でシンボル名または SFR を選択し、右クリックのコンテクスト メニューから「New Watch」オプションを選択して作成する事もできます。

図 3-16: [WATCHES] ウィンドウに表示されたシンボル



シンボル値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. [Watches] タブをクリックしてウィンドウを開くと、シンボル値が表示されます (変化した値は赤字で示されます)。

ウォッチの詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Debugging C/C++/Fortran Applications with gdb]>[Viewing C/C++/Fortran Program Information]>[Creating a C/C++/Fortran Watch]) を参照してください。

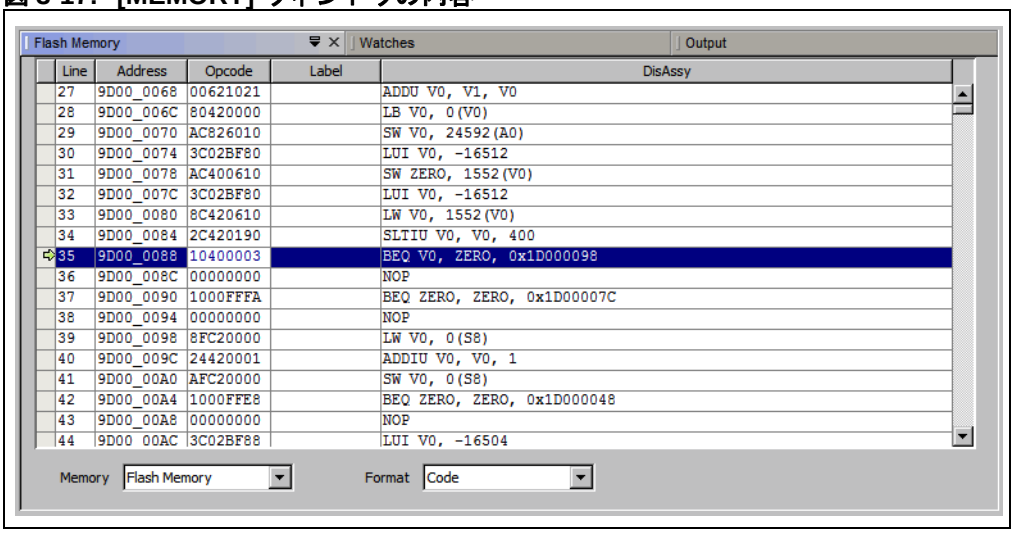
3.17 デバイスメモリ (コンフィグレーション ビットを含む) の表示

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、各種の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。[Memory] ウィンドウでは、メモリのタイプとメモリのフォーマットをドロップダウン ボックスで選択します。

まずフラッシュメモリを表示します。

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

図 3-17: [MEMORY] ウィンドウの内容



下記の方法により、[Memory] ウィンドウの表示を変更できます。

- [Window]>[PIC Memory Views] を選択してリストを開き、そこで別のウィンドウを選択する
- ウィンドウ上でドロップダウン [Memory] メニューを使う

[Memory] ウィンドウのオプションを設定する方法

[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) を含むポップアップ メニューが開きます。このメニューの詳細は、**10.3.4.1「[Memory] ウィンドウのメニュー」**を参照してください。

3.18 デバイスのプログラミング

コードをデバッグした後に、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。これには下記の 2 つの方法があります。

- **[Run]** をクリックする : プロジェクトをビルドし (必要な場合のみ)、デバイスをプログラミングします。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。
- **[Make and Program Device]** をクリックする : プロジェクトをビルドし (必要な場合のみ)、デバイスをプログラミングします。プログラミングが完了しても、プログラムの実行は始まりません。アプリケーションを実行する前に、ターゲットボードからハードウェア ツールを切断する必要があります。

その他のプログラミング関連機能 :

- **Hold in Reset**: デバイスをリセットと実行の間で交互に切り換えます。
- **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。

図 3-18: プログラミング関連のアイコン

[Run] アイコン

[Hold in Reset] アイコン

[Make and Program Device] アイコン

[Read Device Memory] アイコン

Chapter 4. 基本作業

本章には、下記の「MPLAB X IDE プロジェクトの作業手順」に要約した一連の作業を実施するための段階的なガイドを記載しています。

4.1 MPLAB X IDE プロジェクトの作業手順

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。

準備

1. MPLAB X IDE をインストールし、ハードウェア ツールをセットアップし (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)、選択したデバイス向けの言語ツールをインストールします。それらを済ませた後に MPLAB X IDE を起動します。

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新しいプロジェクトを作成します。デスクトップ画面の表示が切り換わります。
2. [Project Properties] ダイアログでプロジェクトのプロパティを確認または編集します。同じダイアログでデバッガ、プログラマ、言語ツールのオプションも設定します。
3. [Tools Options] ダイアログで、言語ツールのパスと、追加のツールオプションを設定します。
4. プロジェクト ファイルを新規作成してプロジェクトに追加します。または、既存のファイルをプロジェクトに追加します。追加したファイルは [File] 枠に表示され、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. エディタの機能については「エディタの使い方」を参照してください。
6. ライブラリやその他のファイルをプロジェクトに追加します。
7. 各ファイルのプロパティ (ビルドに含めるかどうか) を設定します。
8. ビルド前後のステップと、ビルド完了時の代替 HEX ファイルの読み込みに関するビルド 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。

コードの実行

1. [Run] メニューを使ってコードを実行します。
2. [Debug] メニューを使ってコードをデバッグ実行します。

コードのデバッグ

1. ライブラリやその他のファイルをプロジェクトに追加します。ブレークポイントも設定します (エディタ内で 1 行を選択して設定するか、[Breakpoint] ウィンドウを使って設定)。
2. コードをステップ実行します。
3. [Watches] および [Variables] ウィンドウでシンボルおよび変数値の変化を観察します。
4. デバイスメモリ (コンフィグレーション ビットを含む) を表示/変更します。メモリのタイプはデバイスによって異なります。
5. 関数コールを調べるためにコールスタックを表示します。

デバイスのプログラミング

1. ツールバーボタンを使ってデバイスをプログラミングします。

4.2 新規プロジェクトの作成

MPLAB X IDE はプロジェクトを基本とするため、アプリケーション向けにプロジェクトをセットアップする必要があります。

プロジェクトは下記のいずれかの方法で作成できます。

- [Start Page] の「Learn & Discover」タブで、「Dive In」の下の「Create New Project」を選択する
- [File]>[New Project] を選択する (または Ctrl+Shift+N)

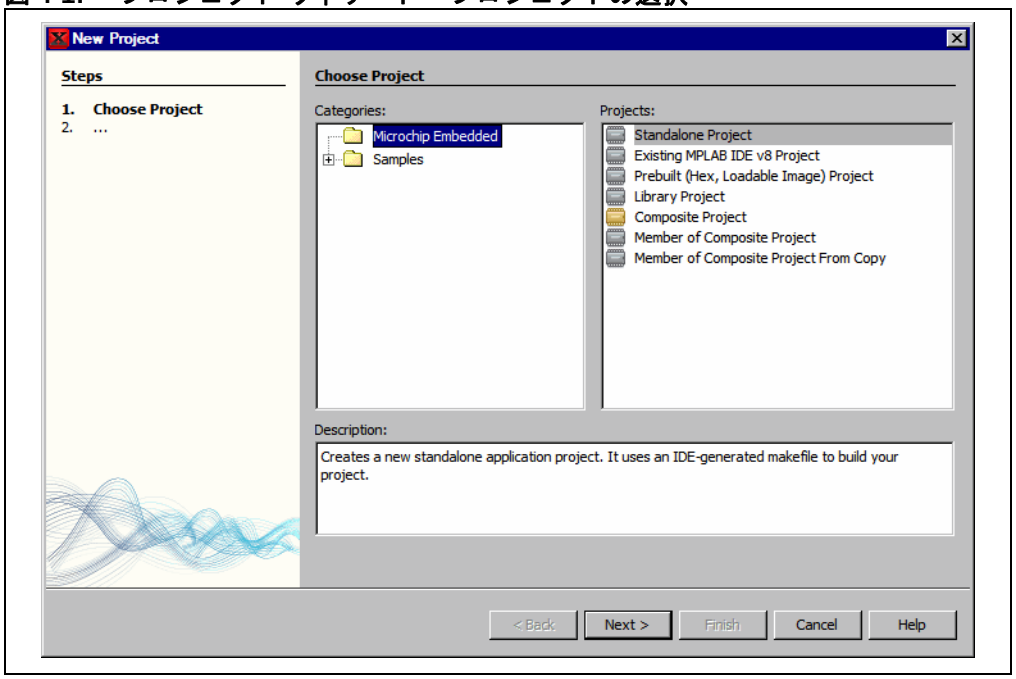
[New Project] ウィザードが起動し、新規プロジェクトのセットアップ手順をガイドします。

Step 1 では、プロジェクト カテゴリを選ぶよう指示されます。これは NetBeans のダイアログです。マイクロチップ社製品を使うには、「Microchip Embedded」を選択する必要があります。次にプロジェクトのタイプを選択します。

- Stand-alone Project – 新しいプロジェクト (C 言語および / またはアセンブリコード) を作成します。
- 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。
- Prebuilt (Hex, Loadable Image) Project – 既存のプロジェクト イメージを MPLAB X IDE に読み込みます。
- 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 言語および / またはアセンブリコード) を作成します。
- Composite Projects – アプリケーション開発用に、複数のプロジェクトを 1 つに複合したプロジェクトを作成します。

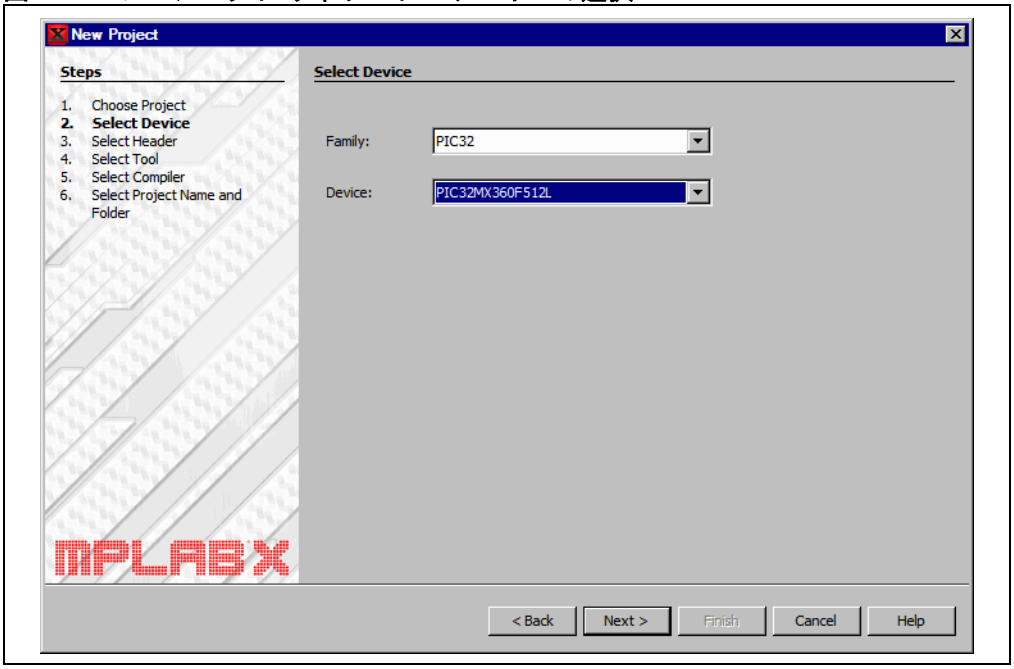
上記を選択した後に、**[Next>]** をクリックして次のダイアログへ進みます。

図 4-1: プロジェクト ウィザード – プロジェクトの選択



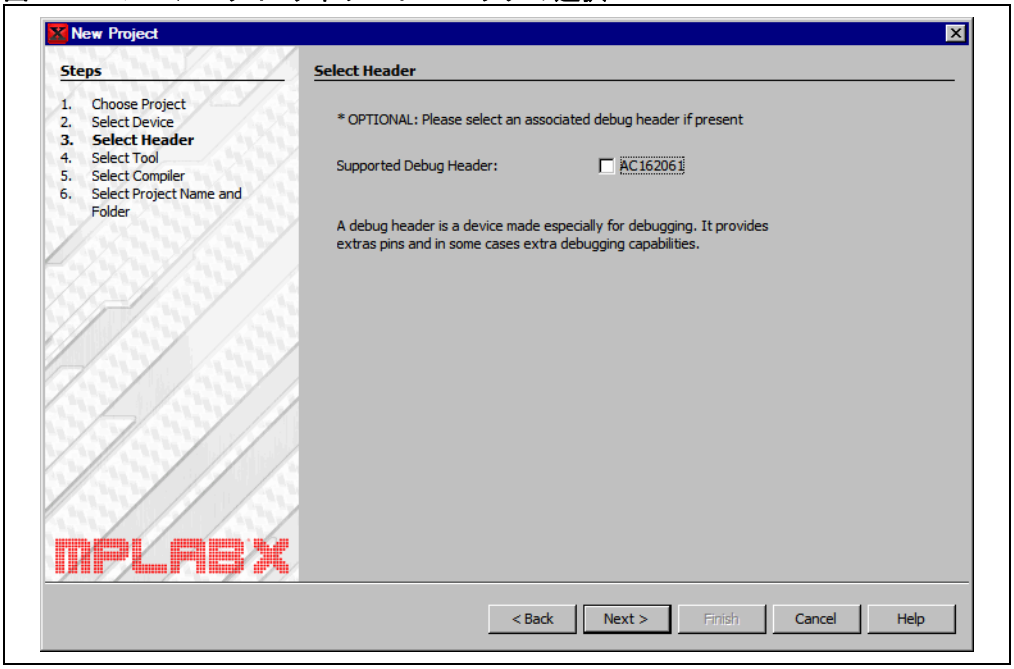
Step 2 は MPLAB X IDE のダイアログです。このため Step 1 のダイアログとは外観が異なります。アプリケーションで使う予定のデバイスを [Device] ドロップダウン リストから選択します。選択肢を絞り込むために、最初にデバイスファミリを選択します。選択したら **[Next>]** をクリックします。

図 4-2:　プロジェクト ウィザード – デバイスの選択



Step 3 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、使用するデバイスがデバッグ回路を内蔵しているかどうかは、ヘッダ仕様 (DS51292 またはオンラインヘルプ) で確認してください。上記を確認してからヘッダを使うかどうかを選択します。選択したら **[Next>]** をクリックします。

図 4-3:　プロジェクト ウィザード – ヘッダの選択



Step 4 ではツールを選択します。選択したデバイスをサポートしているツールは、ツール名の左側の丸マークの色で識別できます。これは MPLAB IDE v8 の [Device Selection] ダイアログと同じです。マークの色分けは下記の通りです。

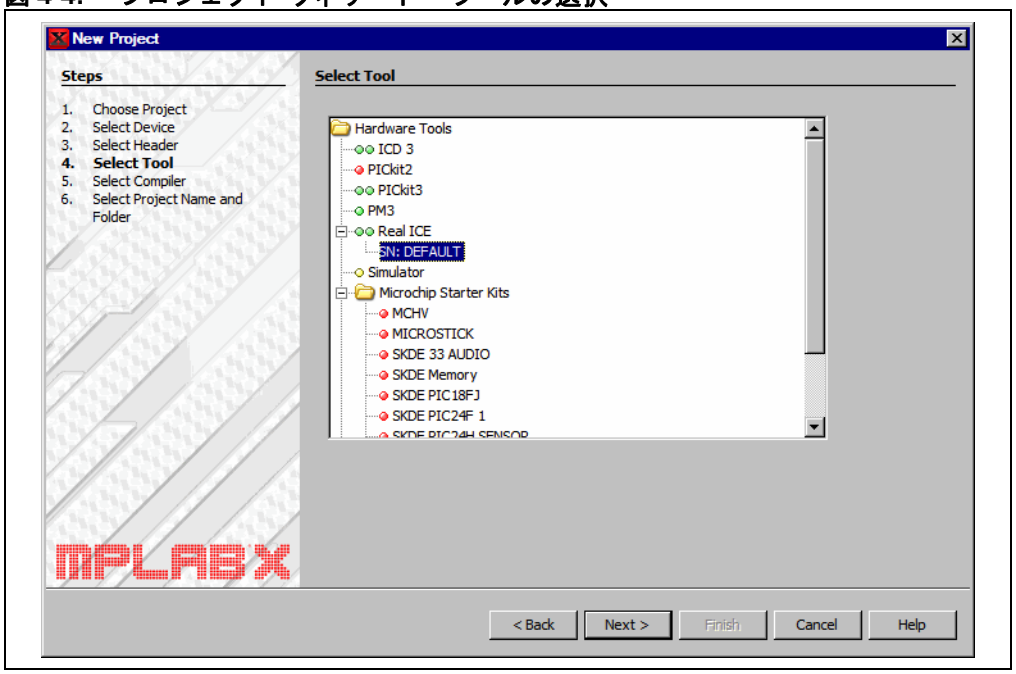
- 緑 – 完全サポート
- 黄 – ベータ版サポート
- 赤 – 未サポート

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。

ハードウェア ツールを PC に接続している場合、それらのツールの下にシリアル番号 (SN) が表示されます。これにより、接続した複数のハードウェア ツールから必要なツールを選択できます。

ツールを選択してから **[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

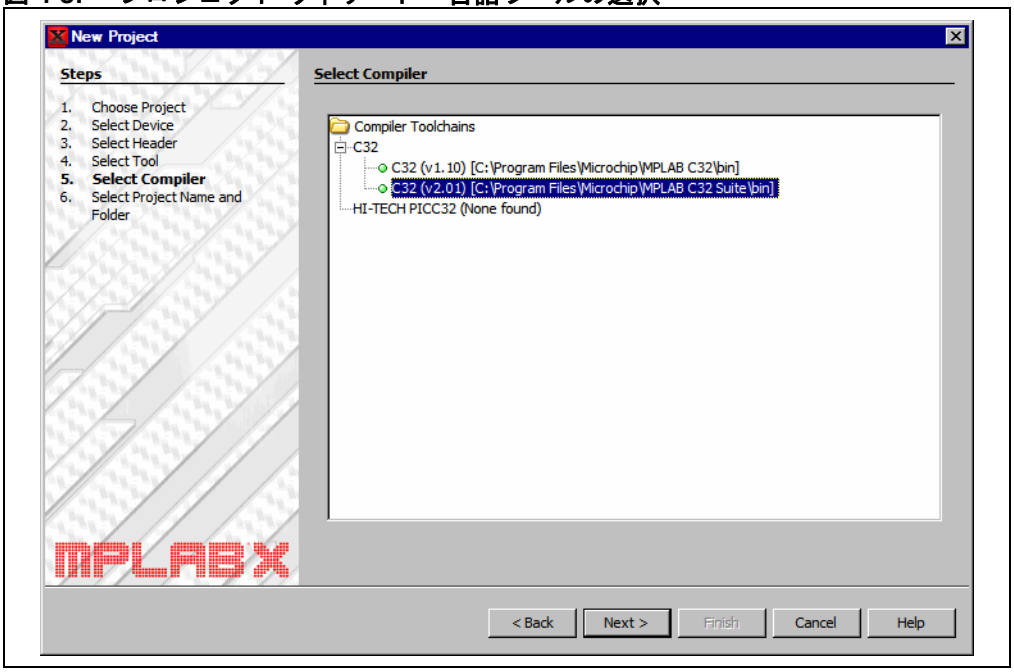


Step 5 では、言語ツール (C コンパイラまたはアセンブラ) を選択します。ここでも、コンパイラ名の左側のマークの色で、デバイスに対するサポートレベルを識別できます。マウスカーソルをマークの上に置くと、サポートに関する説明を表示できます。

[New Project] ウィザードは、Step 2 で選択したデバイスに適した言語ツールだけを表示します。インストールされている各ツール名の右側には、バージョンとインストール先のパスも表示されます。これにより、インストールされている複数の言語ツールから必要なツールを選択できます。

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 – 言語ツールの選択



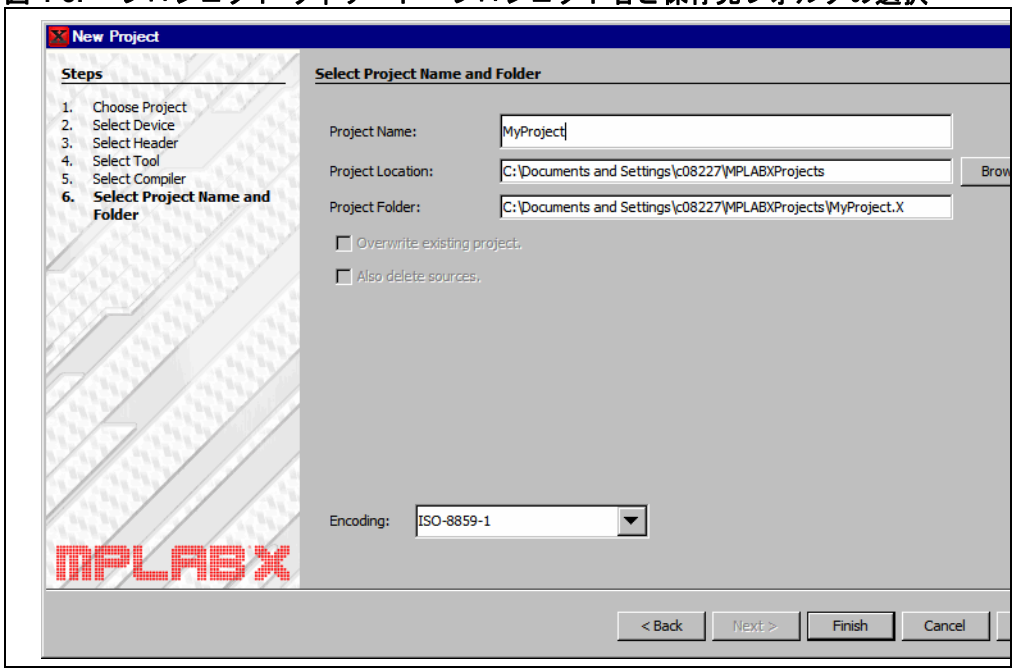
Step 6 では、プロジェクト名とプロジェクトの保存場所を選択します。プロジェクト名を入力してからフォルダを選択します。必要に応じて新しいフォルダを作成する事もできます。既定値では、プロジェクトは下記のパスに保存されます。

- Windows XP – `C:\Documents and Settings\UserName\MPLABXProject`
- Windows 7 – `C:\Users\UserName\MPLABXProjects`
- Linux – `/home/UserName/MPLABXProjects`
- Mac – `/Users/UserName/MPLABXProjects`

既定値以外のパスを選択する事もできます。

上記を済ませた後に、**[Finish]** をクリックすると、新規プロジェクトの作成が完了します。

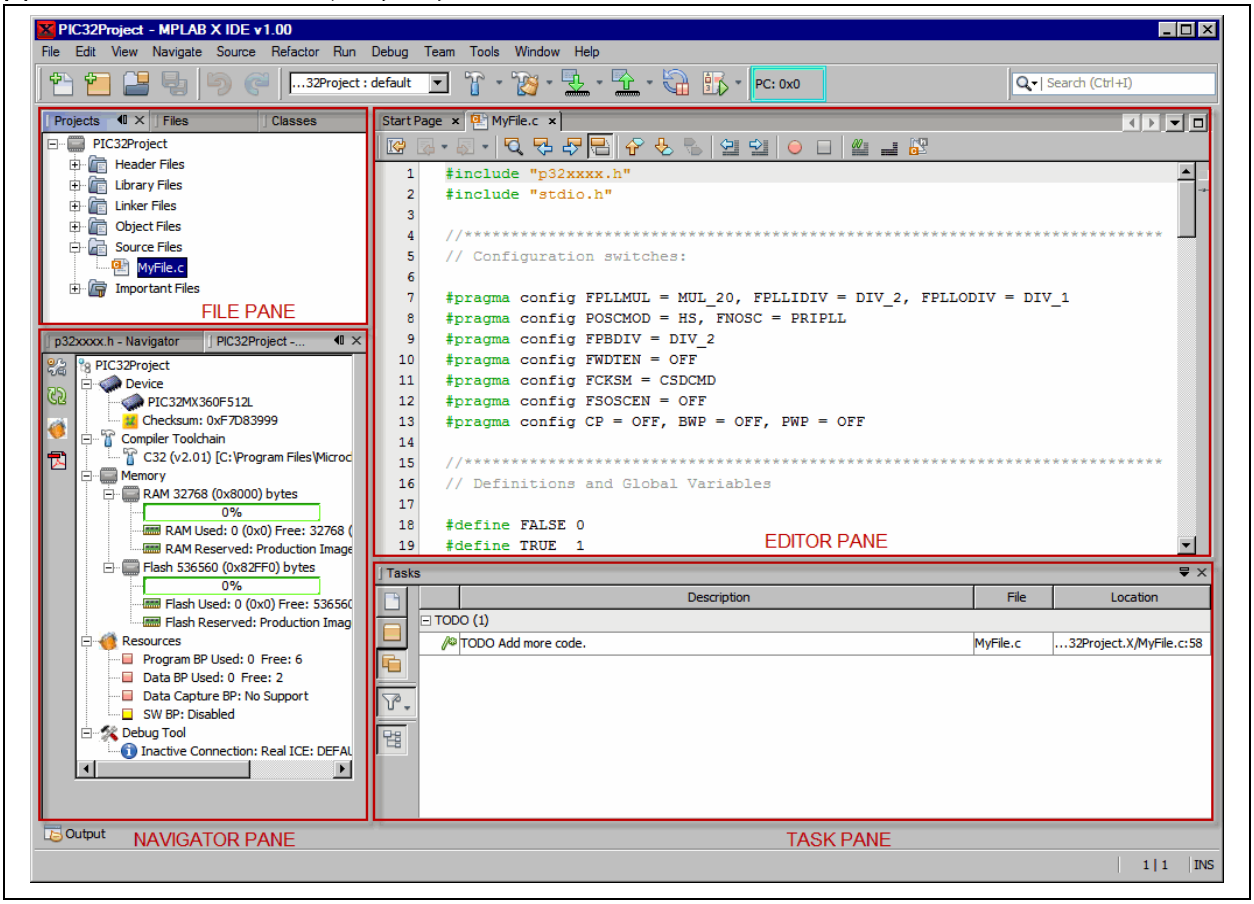
図 4-6:　プロジェクト ウィザード – プロジェクト名と保存先フォルダの選択



4.3 プロジェクト作成後のデスクトップ画面

プロジェクトの作成が完了すると、IDE 内に各種のウィンドウ枠が開きます。

図 4-7: MPLAB X IDE のデスクトップ



- [File] 枠 – ファイルに関連する 4 つのタブを含みます。[Projects] タブはプロジェクト ツリー、[Files] タブはプロジェクト ファイル、[Classes] タブはコード内の全てのクラス、[Services] タブはコードの開発に使える全てのサービスを表示します。
- [Navigator] 枠 – [File] 枠で選択されている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] 枠 – 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] も表示できます。
- [Task] 枠 – 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。

[File] 枠内で、いずれかの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] 枠内の [Start Page] タブの隣のタブに表示されます。タブを閉じるには、タブ名の右横の「x」をクリックします。

[File] 枠の [Projects] タブ内でプロジェクト名を右クリックすると、ポップアップ (コンテクスト) メニューが開きます。プロジェクトのサブフォルダに対しても同様の操作が可能です。

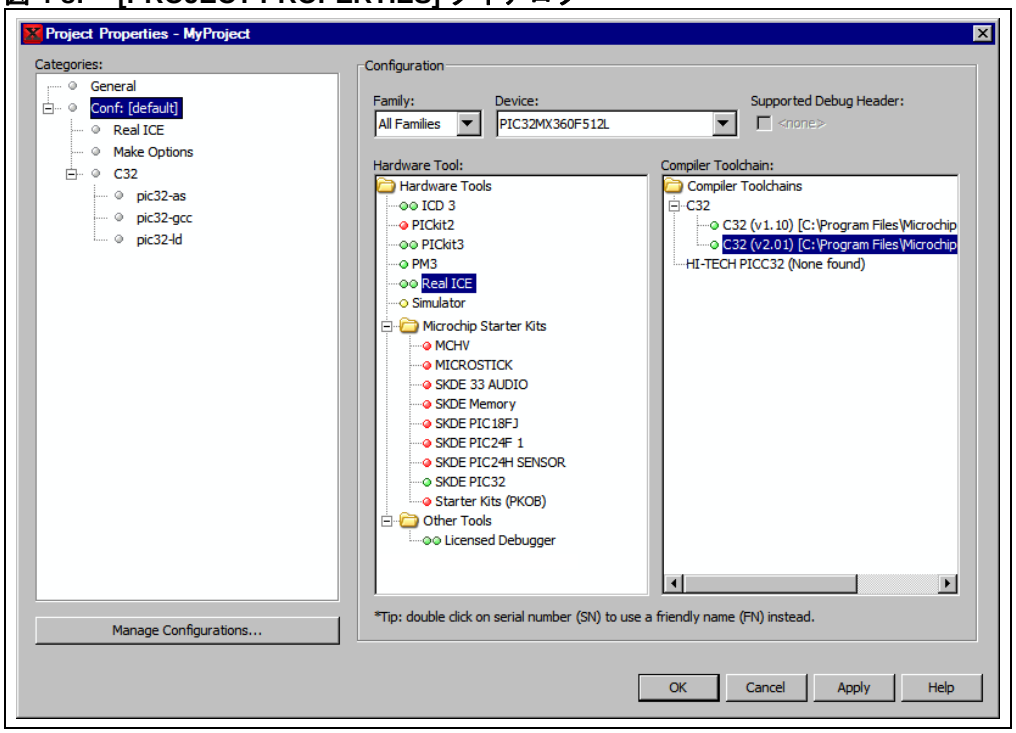
4.4 プロジェクト プロパティの表示と変更

プロジェクトを作成した後に、[Project Properties] ダイアログにプロジェクト プロパティを表示して編集できます。このダイアログは下記のいずれかの方法で開く事ができます。

- [File] 枠の [Projects] タブ内でプロジェクト名を右クリックし、「Properties」を選択する
- [File] 枠の [Projects] タブ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] を選択する

開いたダイアログで、「Conf:[default]」カテゴリをクリックすると、標準的なプロジェクト コンフィグレーション (プロジェクト デバイス、関連するデバッガ / プログラマ ツール、言語ツール等) が表示されます。これらの項目を変更する必要がある場合、選択を変更してから **[OK]** をクリックします。

図 4-8: [PROJECT PROPERTIES] ダイアログ



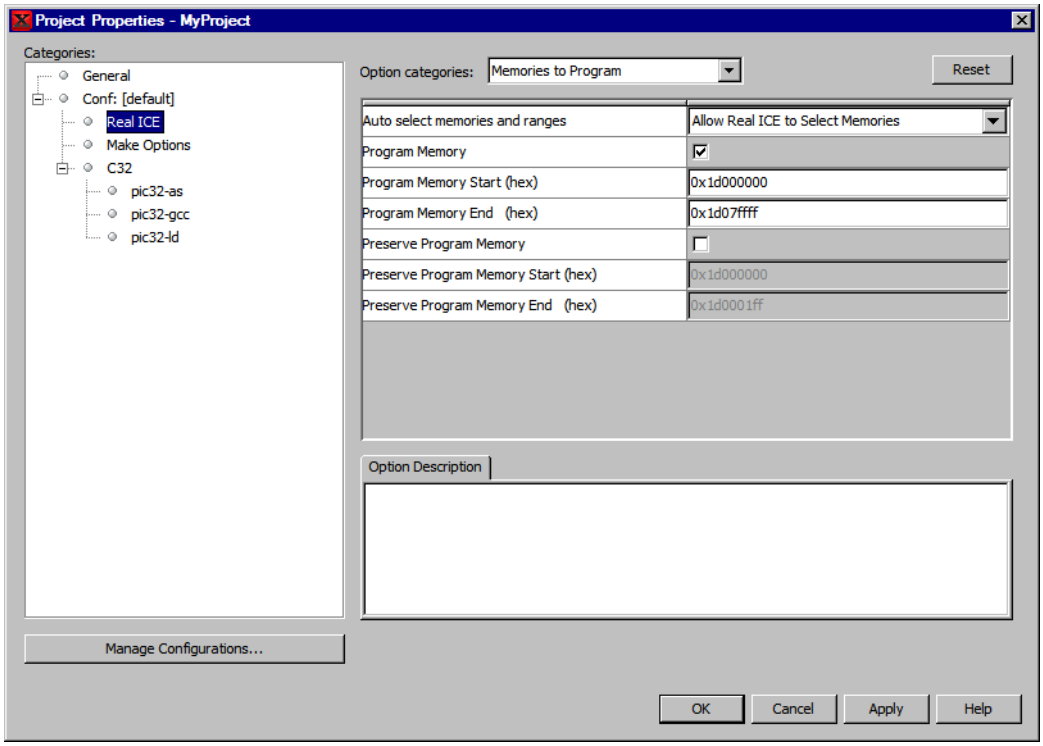
4.5 デバッガ、プログラマ、言語ツールのオプション設定

[Project Properties] ダイアログでは、ツールのオプションも設定します。

デバッガ / プログラマ ツール オプションをセットアップまたは変更する方法

- ハードウェア ツール (またはシミュレータ) をクリックして、関連するセットアップ オプションを表示します。これらのオプションの意味については、各ツールの関連文書を参照してください。

図 4-9: ツールのセットアップ ページ

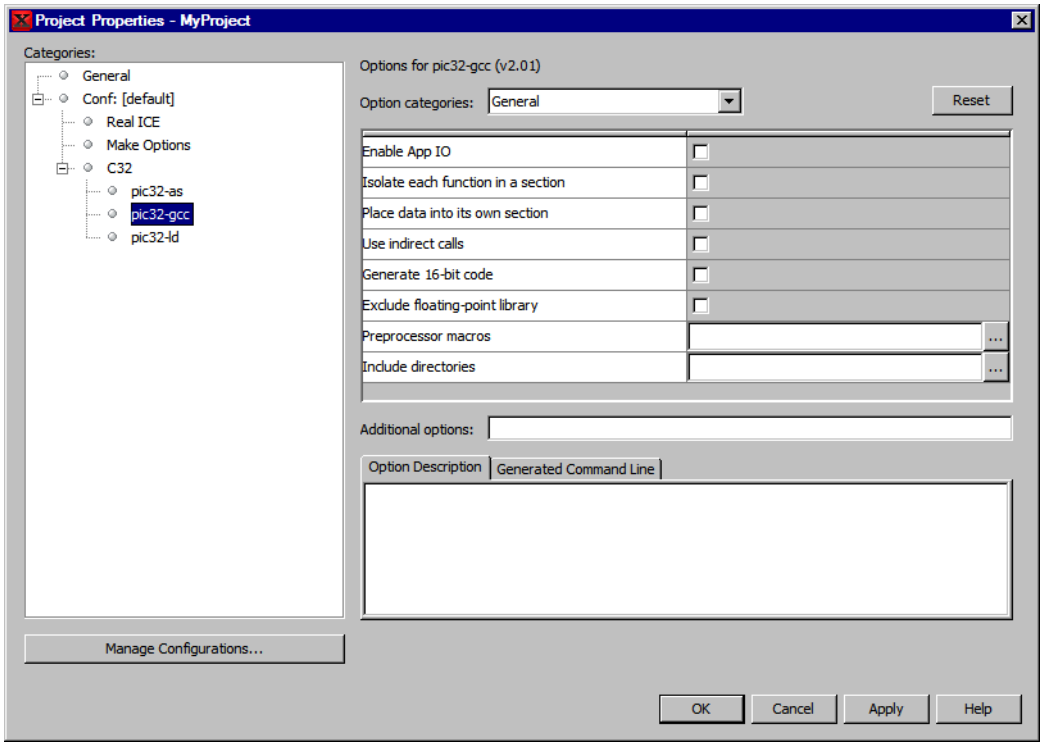


言語ツール オプションをセットアップまたは変更する方法

- 言語ツールをクリックして、関連するセットアップ オプションを表示します。これらのオプションの意味については、言語ツールの文書を参照してください。

その他のヘルプについては、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Working with C/C++/Fortran Projects]>[Setting Project Properties]) を参照してください。

図 4-10: 言語ツールのセットアップ ページ



4.6 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または変更する方法

- **Mac OS の場合 :**
  メイン メニューバーから [mplab_ide]>[Preferences]>[Embedded]>[Build Tools] を選択してビルドツールにアクセスします。
- **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します。

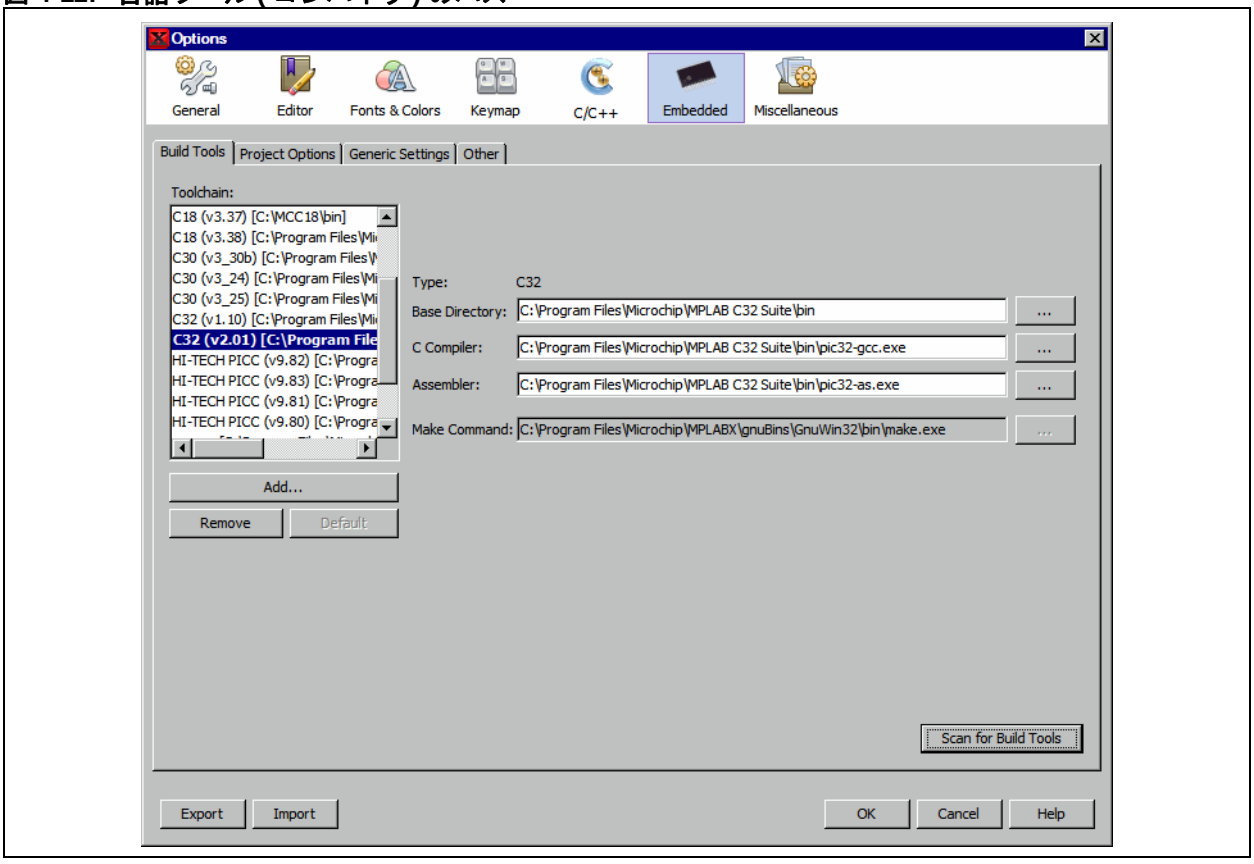
ウィンドウには、インストールされている全てのツールチェーンが自動的に表示されます。インストール済みのツールが表示されない場合、下記を試してください。

- **[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格納したディレクトリ (ベース ディレクトリ) 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ディレクトリ内の「bin」サブディレクトリ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、新しいパスをタイプ入力するか、**[...]** ボタンを使って選択します。

図 4-11: 言語ツール (コンパイラ) のパス



4.7 その他のツールオプションの設定

ビルド用パス以外のオプションも設定できます。[Options] ウィンドウの下記のタブから「Embedded」カテゴリを選択します。

- Project Options – プロジェクト関連のオプション (make オプション、ファイルパスオプション (自動 / 相対 / 絶対) 等) を設定します (**10.3.5.2「[Project Options] タブ」**参照)。
- Generic Settings – ログファイル等のプロジェクト機能を設定します(**10.3.5.3「[Generic Settings] タブ」**参照)。
- Other – C 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 (**10.3.5.4「[Other] タブ」**参照)。

4.8 新規プロジェクト ファイルの作成

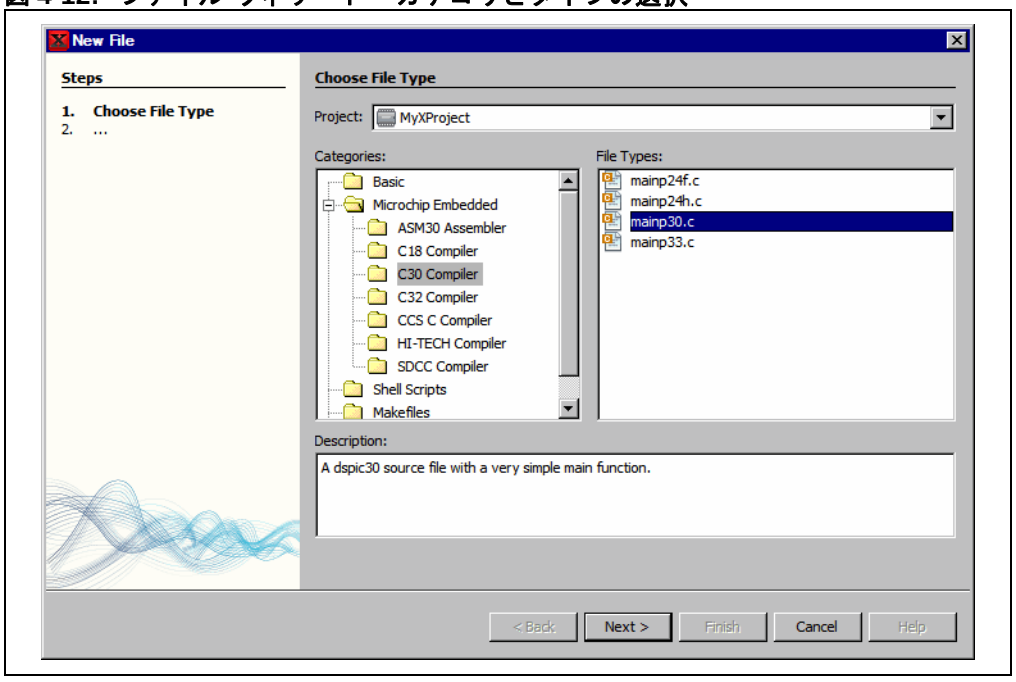
新しいプロジェクトファイルは、下記のいずれかの方法で作成できます。

- [File]>[New Project] を選択する (または Ctrl+N)
- [Project/File] ウィンドウ内のプロジェクトを右クリックし、[New]>[Other] を選択する
- [Project/File] ウィンドウ内の論理フォルダ (例 : Source Files) を右クリックし、[New]>[Other] を選択する

[New File] ウィザードが起動し、新規ファイルのセットアップ手順をガイドします。

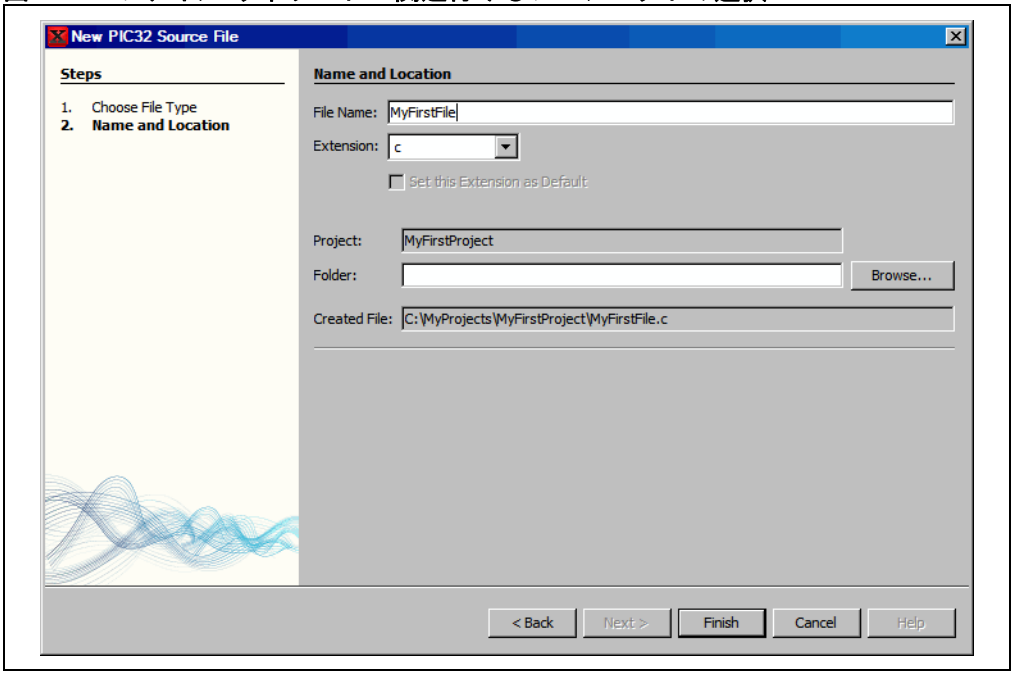
- **Step 1. Choose File Type:** 「Microchip Embedded」を展開して、その中のファイルカテゴリを選択します。次にファイルタイプを選択します。

図 4-12: ファイル ウィザード – カテゴリとタイプの選択



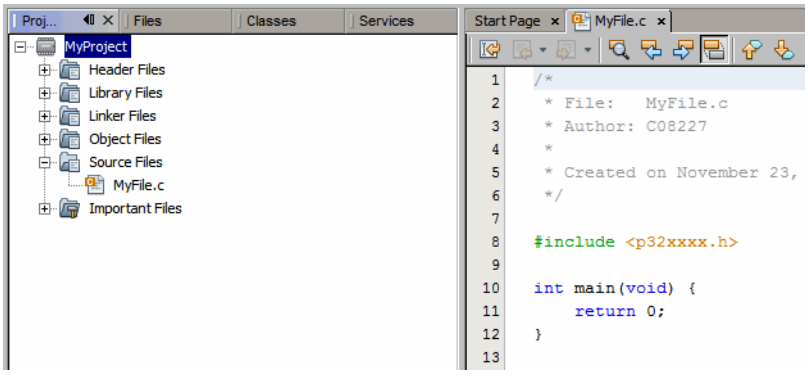
- **Step 2. Name and Location:** ファイルに名前を付け、そのファイルをプロジェクトフォルダ内に配置します。

図 4-13: ファイル ウィザード – 関連付けるプロジェクトの選択



このファイルは、[File] 枠内のプロジェクトの下に表示されるとともに、[Editor] 枠内にこのファイルと同名のタブが開きます。このタブでは、ファイルの内容を編集できます。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 (色分けはファイルのタイプによって異なります)。

図 4-14: 新規ファイル – MYFILE.C



4.9 既存ファイルのプロジェクトへの追加

下記のいずれかの方法により、既存ファイルをプロジェクトに追加できます。

- [Projects] タブ内のプロジェクトを右クリックし、「Add Existing Item」を選択する
- [Project/File] ウィンドウ内の論理フォルダ (例 : Source Files) を右クリックし、「Add Existing Item」を選択する

4.9.1 プロジェクト フォルダ内のファイル

ファイルを追加する際に、パスの指定方法を下記から選択できます。

- 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。

このファイルは、[File] 枠内のプロジェクトの下に表示されるとともに、[Editor] 枠内にこのファイルと同名のタブが開きます。

4.9.2 プロジェクト フォルダ外のファイル

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、ファイルの格納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、下記を行います。

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ブラウザ機能を使って選択し、**[Select]** をクリックします。
5. **[Apply]** または **[OK]** をクリックしてから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。従って、上記の手順を実行して v8 フォルダのパスをプロジェクト内で指定する必要があります。

4.10 エディタの使い方

新規コードの作成または既存コードの編集には、NetBeans エディタを使います。このエディタに関する一般情報は、NetBeans ヘルプトピック ([IDE Basics]>[Basic File Features]) を参照してください。エディタに関連する C コンパイラ情報は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[C/C++/Fortran Project Basics]>[Navigating and Editing C/C++/Fortran Source Files]) を参照してください。エディタ機能の一覧は、**6.3「NetBeans™ エディタ」**に記載しています。

エディタの各種機能は下記から利用できます。

- [Edit] メニュー (**9.2.2「[Edit] メニュー」**参照)
- 各ファイルのエディタ ウィンドウの上にあるエディタ用ツールバー

図 4-15: エディタ用ツールバー

- 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

エディタ プロパティの設定方法

1. [Tools]>[Options] 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セットアップします。
3. **[Fonts and Colors]** ボタンをクリックします。タブをクリックして色オプションを設定します。

ファイルをナビゲートする方法とコードをフォーマットする方法

1. [Navigate] メニューから項目を選択し、ファイルの内外を検索して移動します。
2. [Source] メニューから項目を選択して、コードのフォーマット、コメント化、オートコンプリートを実行します。

4.11 ライブラリ等のファイルのプロジェクトへの追加

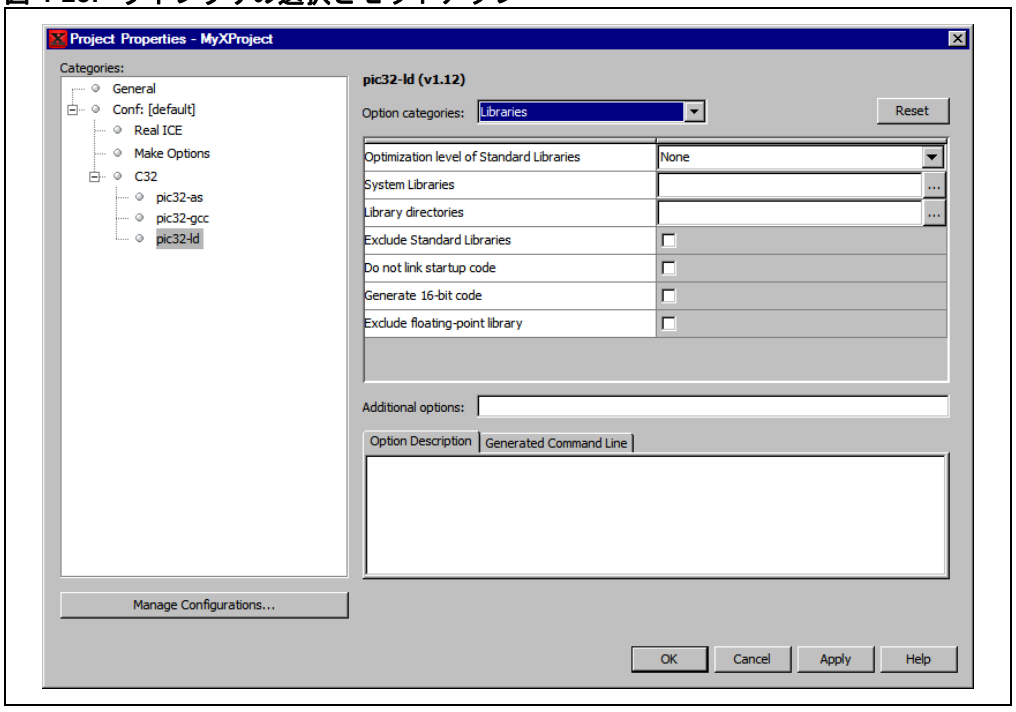
[Project Properties] ダイアログでは、リンカが使うライブラリ ファイルを参照できます。このダイアログの開き方は **4.4「プロジェクト プロパティの表示と変更」**参照してください。

左側のウィンドウ枠でリンカの名前をクリックしてから、右側のウィンドウ枠の「Option categories」で「Libraries」を選択します (図 4-16 参照)。

このプロジェクトに必要なライブラリを作成する他のプロジェクトを作成する事もできます (依存性の確立)。これに関する詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development>C/C++/Fortran Project Basics]>[Building C/C++/Fortran Projects]>[Creating Dependencies Between C/C++/Fortran Projects]) を参照してください。

既存のライブラリ ファイルをプロジェクトに追加するには、[Project] ウィンドウ内でプロジェクトまたは論理フォルダを右クリックして、「Add Existing Item」を選択します。ファイルの参照方法 (auto/relative/absolute) を選択してから <Enter> キーを押します。

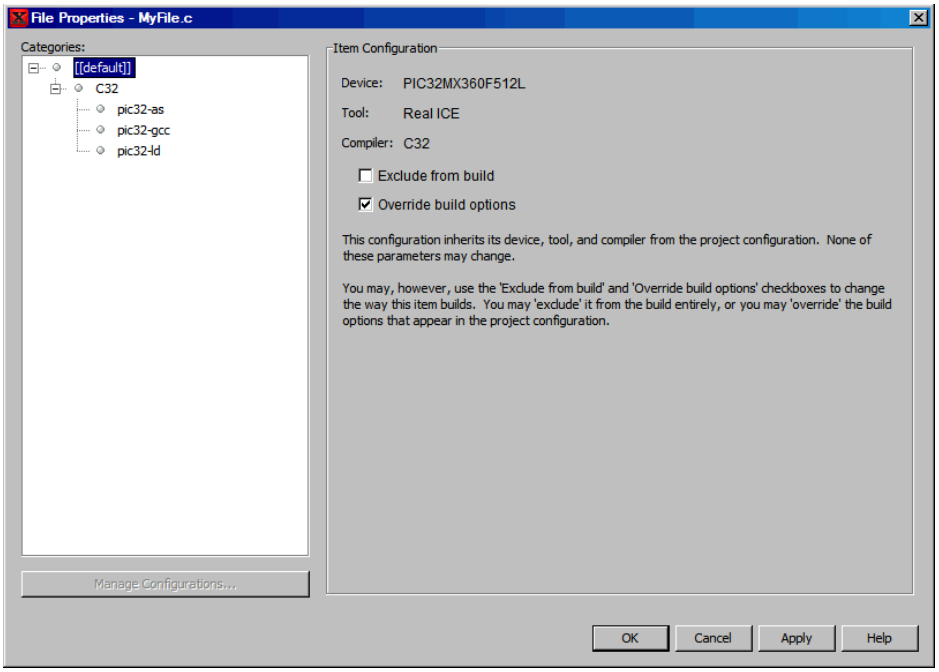
図 4-16: ライブラリの選択とセットアップ



4.12 ファイル プロパティの設定

[File Properties] ダイアログを使うと、各プロジェクト 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、[Project] 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」チェックボックスを有効にすると、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」チェックボックスを有効にすると、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を上書きできます。「Override build options」チェックボックスを有効にしてから **[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 4-17: ファイル プロパティ



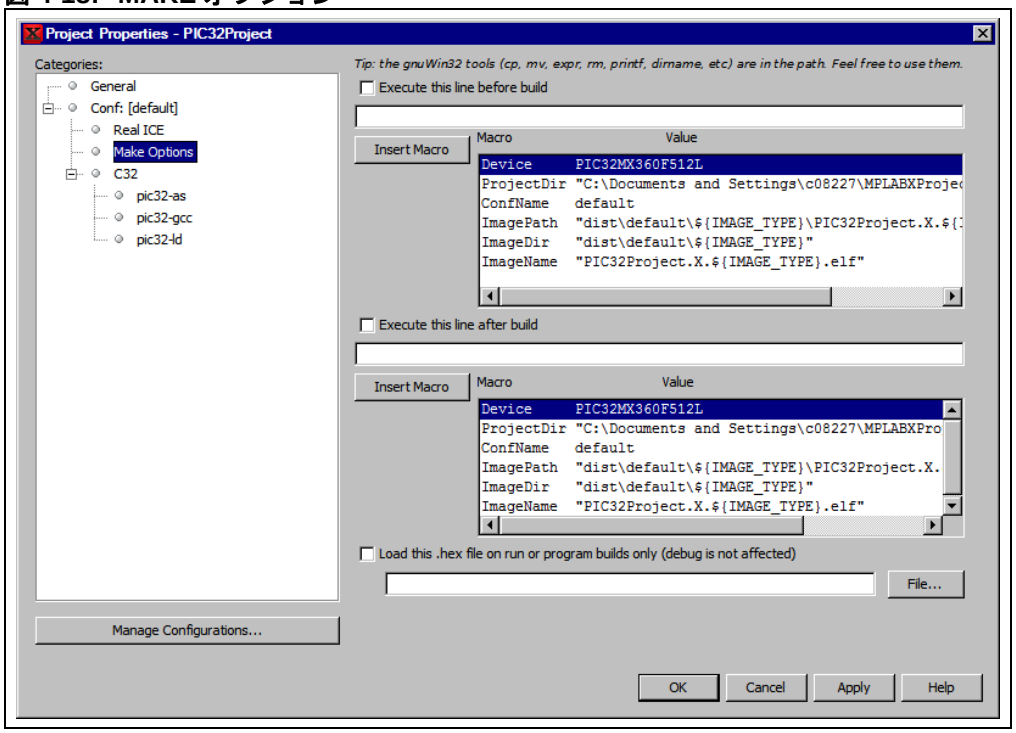
4.13 ビルド プロパティの設定

プロジェクトをビルドする前に、必要に応じて下記のビルド プロパティを設定できます。

表 4-1: ビルドの追加オプション

項目	内容
Execute this line before the build	プリビルド /make ステップです。リストからマクロを選択し、**[Insert Macro]** をクリックします。使用するコンパイラによっては、その他のオプションも選択できる場合があります。
Execute this line after the build	ポストビルド /make ステップです。リストからマクロを選択し、**[Insert Macro]** をクリックします。使用するコンパイラによっては、その他のオプションも選択できる場合があります。
Load this hex file on run or program builds only (debug not affected)	ビルド完了時に代替 HEX ファイルを読み込みます。

図 4-18: MAKE オプション



4.14 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、実行および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しておきたい場合もあります。

プロジェクトのビルド方法

- [Projects] タブ内のプロジェクト名の上で右クリックして [Build] を選択する、または、「Clean and Build」を選択して中間的に生成された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

[Build] アイコン

[Clean and Build] アイコン

ビルドの進捗状況は、[Output] ウィンドウ (デスクトップの右下隅) に表示されます。

チェックサム情報の表示方法

- [Dashboard] ウィンドウ (**5.10「ダッシュボードの表示」**参照) を開いて、ビルド後のチェックサムを確認します。

プロジェクトのビルドに関する詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[C/C++/Fortran Project Basics]>[Building C/C++/Fortran Projects]) を参照してください。

4.15 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。

アプリケーションコードを実行する方法

1. [Project] ウィンドウで、プロジェクトを選択するか、あるいは、そのプロジェクトをメインプロジェクトに指定します (プロジェクトを右クリックして「Set as main」を選択)。
2. プログラムを実行するには、[Make and Program Device Project] アイコンをクリックするか [Run]>[Run Project] を選択します。

[Make and Program Device Project] アイコン

コード実行の進捗状況も [Output] ウィンドウに表示されます。

コード実行時の動作

make 処理がビルドを必要と判断した場合、「Run Project」をクリックするとビルドが始まります。

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。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」をクリックします。

[Hold in Reset] アイコン

MPLAB X IDE は、コード実行時またはデバッグ実行時にだけハードウェア ツールに接続します。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] 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」チェックボックスを有効にします。

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単純にコードを実行します。

コード実行の進捗状況も [Output] ウィンドウに表示されます。

4.16 コードのデバッグ実行

コードが正常に動作しない場合、デバッグが必要です。デバッグモードでコードを実行する事を「デバッグ実行」と呼びます。

アプリケーションコードをデバッグする方法

1. [Project] ウィンドウで、プロジェクトを選択するか、あるいは、そのプロジェクトをメインプロジェクトに指定します（プロジェクトを右クリックして「Set as main」を選択）。
2. [Debug Project] アイコンをクリックするか、[Debug]>[Debug Project] または [Debug]>[Step Into] を選択して、デバッグ セッションを始めます。

[Debug Run] アイコン

デバッグ実行時の動作

make 処理がビルドを必要と判断した場合、「Debug Project」をクリックするとビルドが始まります。

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に自動的にイメージ (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。

MPLAB X IDE は、コード実行時またはデバッグ実行時にだけハードウェア ツールに接続します。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] 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」チェックボックスを有効にします。

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。

デバッグ実行の進捗状況も [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を一時停止する方法

- [Pause] アイコンをクリックするか、[Debug]>[Pause] を選択すると、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か、[Debug]>[Continue] を選択すると、一時停止していた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] アイコンをクリックするか、[Debug]>[Finish Debugger Session] を選択すると、プログラムの実行が終了します。

C コード プロジェクトのデバッグに関する詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Debugging C/C++/Fortran Applications with gdb]) を参照してください。

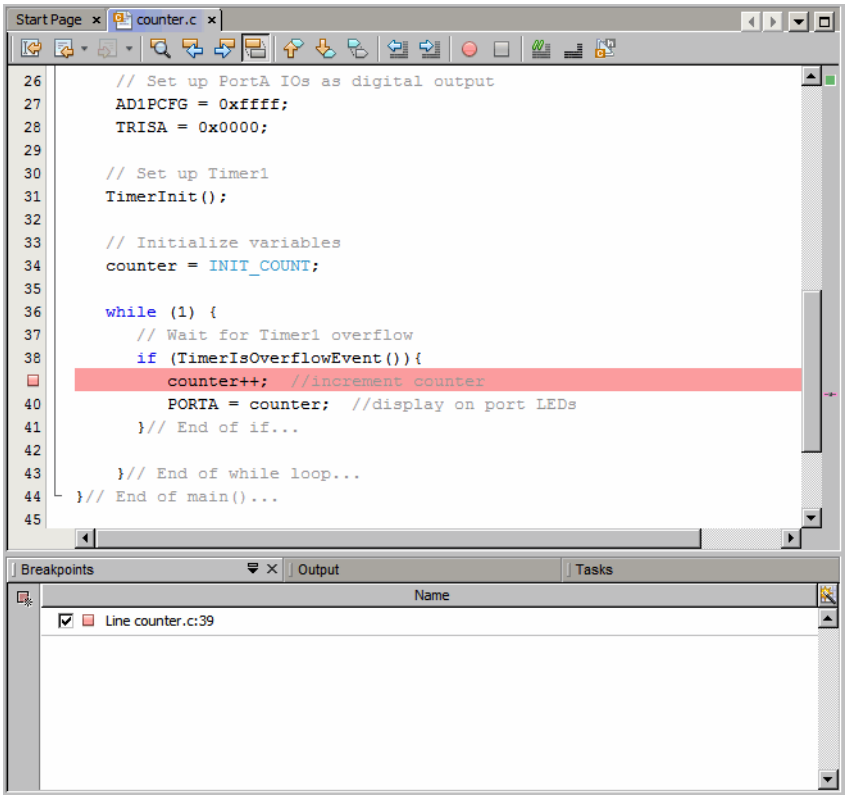
通常の実行とデバッグ実行の違いは、次の **4.17「ブレークポイントによるプログラム実行の制御」**から始まるデバッグ機能の説明を読むとよくわかります。

4.17　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 ([C/C++/Fortran Development]>[Debugging C/C++/Fortran Applications with gdb]>[Setting C/C++/Fortran Breakpoints]) を参照してください。

図 4-19:　[BREAKPOINTS] ウィンドウ内のブレークポイント



4.17.1　単純なブレークポイントの設定 / クリア

下記のいずれかの方法により、コードの 1 行にブレークポイントを設定できます。

- ソースエディタ内の該当する行の左側の余白をクリックする
- Ctrl-F8 を押す

下記のいずれかの方法により、設定したブレークポイントをクリアできます。

- ブレークポイントの設定と同じ操作を繰り返す
- [Debug]>[Toggle Breakpoint] を選択する

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

[New Breakpoint] ダイアログを使ってブレークポイントを設定する方法

1. [Debug]>[New Breakpoint] を選択します。
2. [New Breakpoint] ダイアログで、ブレークポイントのタイプ等のオプションを設定します。

4.17.3 [Breakpoints] ウィンドウによるブレークポイントの有効化 / 無効化

[Breakpoints] ウィンドウを使ってブレークポイントを表示し、有効 / 無効にする方法

1. [Window]>[Debugging]>[Breakpoints] を選択します。
2. チェックボックスを使ってブレークポイントの有効 / 無効を切り換えます。

[New Breakpoint] ダイアログを使ってブレークポイントを有効にする方法

1. ウィンドウの左上にあるアイコンをクリックします。

4.17.4 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は停止せず、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に、その中の 1 つのパスの動作だけを調べたい時に便利です。

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の上で右クリックします。
2. ポップアップ 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の名前を入力し、**[OK]** をクリックします。
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* はシーケンスの名前)。
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、ポップアップ 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. シーケンスの下に表示されるブレークポイントをドラッグして移動する事により、シーケンスの並び順を変更できます。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

1. アイテムを右クリックし、「Disable」を選択すると、そのアイテムは一時的に無効になります。
2. アイテムを右クリックし、「Delete」を選択すると、そのアイテムは完全に削除されます。

4.17.5 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 1 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 つのブレークポイント間だけで可能です。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。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アイコンをクリックすると、[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. 上記 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の名前を入力し、**[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* はタプルの名前)。

タプルまたはブレークポイントの無効化 / 削除方法

1. アイテムを右クリックし、「Disable」を選択すると、そのアイテムは一時的に無効になります。
2. アイテムを右クリックし、「Delete」を選択すると、そのアイテムは完全に削除されます。

4.17.6 ブレークポイント用のアプリケーション

ブレークポイント間のタイミングを調べる方法

- ストップウォッチを使います (**5.7「ストップウォッチの使用」**参照)。

ブレークポイントのリソースを調べる方法

- [Dashboard] ウィンドウ (**5.10「ダッシュボードの表示」**参照) を開いて、利用可能および使用中のブレークポイントの数と、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調べます。

4.17.7 ブレークポイントの使用に関する注意事項

スタータキット、インサーキット デバッガ (PICkit 2 および 3 を含む)、MPLAB REAL ICE インサーキット エミュレータでは、利用できるブレークポイントの数が制限されます。利用可能なブレークポイントの数は、選択した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 (**5.10「ダッシュボードの表示」**) で調べる事ができます。

下記の 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表示します。

4.18 コードのステップ実行

[Debug] メニューおよび [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の直後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、下記の方法で行えます。

- Step Over – プログラムの 1 ソース行を実行します。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 1 ソース行を実行します。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 1 ソース行を実行します。関数をステップ実行中である場合、関数の残りを全て実行した後に、呼び出し元のルーチンに戻って停止します。
- 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。
- Animate – 1 ステップずつ画面に表示しているレジスタの値を更新しながらプログラムを実行します。Animate は通常の実行よりも実行速度が遅くなりますが、[Special Function Register] ウィンドウまたは [Watch] ウィンドウでレジスタ値の変化を観察できます。

[Editor] ウィンドウ以外に、[Disassembly] ウィンドウ (**5.8「[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。

ステップ実行の詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Debugging C/C++/Fortran Applications with gdb]>[C/C++/Fortran Debugging Sessions]>[Stepping Through Your C/C++/Fortran Program]) を参照してください。

4.19 シンボルおよび変数値の変化の観察

選択したシンボルとローカル変数の値を、それぞれ[Watches]ウィンドウと[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期待通りに変化するかどうかを確認する事により、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、デバッグ実行を一時停止する必要があります。しかし、一部のツールでは、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、ツールの文書を参照してください。

- ウォッチ
- 変数

4.19.1 ウォッチ

[Watches] ウィンドウは、下記のいずれかの方法で表示できます。

- [Window]>[Debugging]>[Watches] を選択すると [Watches] ウィンドウが開きます。
- [Watches] 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。

ウォッチの新規作成方法

シンボルまたは SFR を [Watches] ウィンドウに追加するには、[New Watch] ダイアログを使うか、あるいは、アイテムを直接追加します。

- [New Watch] ダイアログは、下記のいずれかの方法で使えます。
  - [Watches] ウィンドウ内で右クリックし、「New Watch」を選択するか、あるいは、[Tools]>[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。最後に **[OK]** をクリックします。
- 下記のいずれかの方法により、シンボルまたは SFR を直接追加できます。
  - [Watches] ウィンドウ内で <enter new watch> をクリックし、シンボルまたは SFR の名前をタイプ入力します。
  - [Editor] ウィンドウ内でシンボルまたは SFR 名を選択し、これを「Watches」ウィンドウにドラッグ & ドロップします。

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、クロックを下記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. 実行時命令速度を設定します。

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、上記の「ウォッチの新規作成方法」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。

シンボル値の変化を表示する方法

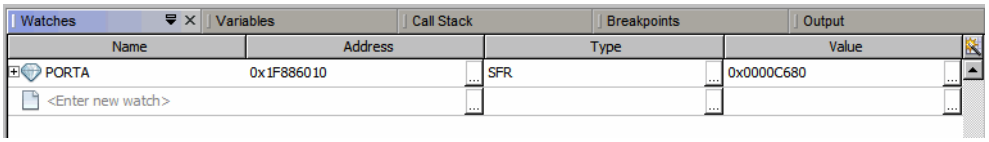
1. プログラムをデバッグ実行し、一時停止します。
2. [Watches] タブをクリックしてアクティブにします。
3. シンボルをウォッチする場合、コードをデバッグ実行し、一時停止して値の変化を観察します。シンボルをランタイム ウォッチする場合、コードをデバッグ実行すると、プログラムを実行しながら値の変化を観察できます。

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。

ウォッチの詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Debugging C/C++/Fortran Applications with gdb]>[Viewing C/C++/Fortran Program Information]>[Creating a C/C++/Fortran Watch]) を参照してください。

図 4-20: [WATCHES] ウィンドウ – プログラムの一時停止



4.19.2 変数

[Variables] ウィンドウは、下記のいずれかの方法で開けます。

- [Window]>[Debugging]>[Variables] を選択すると [Variables] ウィンドウが開きます。
- [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

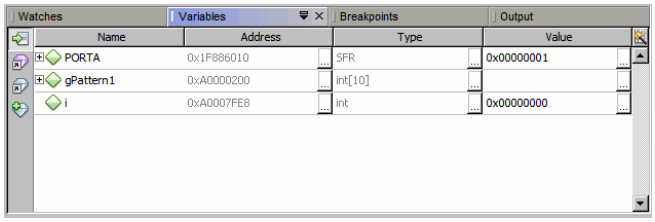
1. プログラムをデバッグ実行し、一時停止します。
2. [Variables] タブをクリックしてウィンドウを開くと、ローカル変数値が表示されます。

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。

変数の詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Debugging C/C++/Fortran Applications with gdb]>[Viewing C/C++/Fortran Program Information]>[C and C++ Variables and Expressions in the IDE]) を参照してください。

図 4-21: [VARIABLES] ウィンドウ – プログラムの一時停止



4.20 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 / 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。このウィンドウでは、デバッグ中に、デバイスメモリの各種タイプの画面をより柔軟にカスタマイズできます。このウィンドウ内の値を更新するには、デバッグ実行を一時停止する必要があります。

デバイスメモリを表示および変更する方法

1. [Window]>[PIC Memory Views] を選択し、下表のオプションを選択します。

表 4-2: メモリ画面

タイプ	内容
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)
SFRs	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
EE Data Memory	デバイス上の全ての EE データメモリ
Other Memory	その他のメモリ

表 4-3: メモリ画面 – PIC32 MCU

タイプ	内容
Flash 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
Other Memory	その他のメモリ

2. [Memory] ウィンドウは [Editor] 枠内に開きます。[Memory] ウィンドウでは、メモリのタイプとメモリのフォーマットをドロップダウン ボックスで選択する事により、表示内容を変更できます。

表 4-4: [MEMORY] ウィンドウのオプション

オプション	値	内容
Memory	File Registers	デバイス上の全てのファイルレジスタ メモリ
	Program	デバイス上の全てのプログラムメモリ
	SFR	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
	EEPROM	全ての EEPROM メモリ
	User ID	ユーザ ID メモリ
Format	Data	データメモリ (RAM)
	Code	プログラムメモリ (ROM)

表 4-5: [MEMORY] ウィンドウのオプション – PIC32 MCU

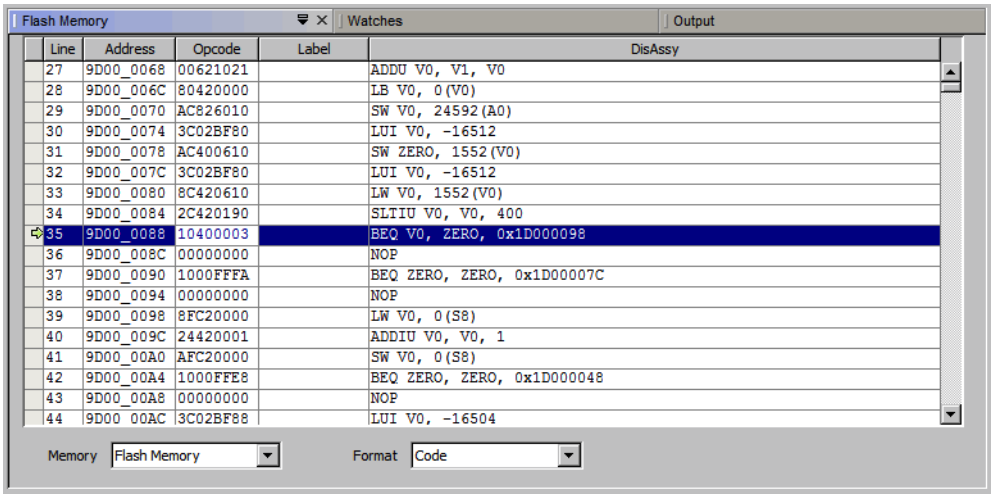
オプション	値	内容
Memory	RAM Memory	デバイス上の全ての RAM
	Flash Memory	デバイス上の全てのフラッシュメモリ
	Peripheral	全ての特殊機能レジスタ (SFR)
	Configuration Bits	全てのコンフィグレーション レジスタ
	CPU Memory	全ての CPU メモリ
	Memory	全てのメモリ
	User ID	ユーザ ID メモリ
Format	Data	データメモリ (RAM)
	Code	プログラムメモリ (ROM)

3. デバッグ実行を一時停止した後に、選択したメモリの内容が [Memory] ウィンドウに表示されます。
4. 適切な列をクリックし、新しいデータを選択または入力する事により、[Memory] ウィンドウ内の値を変更できます。一部のウィンドウでは、変更されたデータを赤いテキストで表示します。
5. 変更したデータをターゲットにプログラミングしてデバッガを起動するには、[Debug]>[Discrete Debugger Operation] を選択します。

> **Note:** 変更したデータは、デバッグ実行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。

6. ウィンドウを閉じるには、[Memory] タブの右上の「x」をクリックします。

図 4-22: [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方法

[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) を含むポップアップ メニューが開きます。このメニューの詳細は、**10.3.4.1「[Memory] ウィンドウのメニュー」**を参照してください。

コンフィグレーション ビットの設定方法

コンフィグレーション ビットはコードで設定する必要があります。しかしデバッグ中は、[Configuration Bits] ウィンドウを使ってコンフィグレーション ビットを一時的に変更できます。前述の「デバイスメモリを表示および変更する方法」を参照してください。

[Configuration bits] ウィンドウ内で右クリックし、「Generate Source Code to Output」を選択する事により、このウィンドウ内の設定を [Output] ウィンドウにエクスポートする事もできます。[Output] ウィンドウにエクスポートしたコードは、アプリケーション コードにコピーできます。

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概要は、本書の **Chapter 12.「コンフィグレーション設定の要約」**に記載しています。

4.21 コールスタックの表示

16 および 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 および 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、コードの最適化を OFF にする事を推奨します)。

[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。

コールスタックに関する詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Debugging C/C++/Fortran Applications with gdb]>[Using the C/C++/Fortran Call Stack]) を参照してください。

4.22 デバイスのプログラミング

コードをデバッグした後に、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。

- **Make and Program Device Project**: これをクリックすると、プロジェクトがビルドされ (必要な場合のみ)、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。

その他のプログラミング関連の機能 :

- **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。
- **Hold in Reset**: デバイスをリセットと実行の間で交互に切り換えます。
- **Run**: 「Make and Program Device Project」と同じです。既定値ではこのアイコンはツールバーに含まれていません。必要な場合は、[View]>[Toolbars]>[Customize] を選択して追加する必要があります。

図 4-23: プログラミング関連のアイコン


[Make and Program Device Project] アイコン


[Read Device Memory] アイコン


[Hold in Reset] アイコン


[Run] アイコン

NOTE:

Chapter 5. 追加の作業

本章では、下の「追加の作業の概要」に従って、基本作業に対する追加の作業について解説します。

5.1 追加の作業の概要

以下は、MPLAB X IDE の追加作業の要約です。

プロジェクトでの作業

1. ウィザードを使って MPLAB X IDE 内で MPLAB IDE v8 のプロジェクトを開く
2. ウィザードを使ってプリビルド イメージ (HEX、COF、ELF) を開く
3. 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
4. プロジェクトで使う既定値のファイル テンプレートを作成または変更する
5. 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

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
2. ウィンドウに逆アセンブラコードを表示する
3. コールグラフを使って関数コールを調べる
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

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良する *
2. ビルトイン ファイル履歴機能またはバージョン管理システムを使ってソースコードを管理する
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 *

機能拡張

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

* この機能については、**[Start Page]** の **[My MPLAB X IDE]** タブで、「Extend MPLAB」セクション内の「Selecting Simple or Full-Featured Menus」を参照してください。

5.2 旧バージョンの MPLAB プロジェクトのインポート

[Import Legacy 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。

- 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。
- インポートする MPLAB IDE v8 プロジェクトが、PIC24 MCU および dsPIC DSC 向け MPLAB C コンパイラと COFF デバッグファイル フォーマットを使っている場合、下記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 プロジェクトをインストールする際に、ファイルの拡張子が .H または .C (大文字) であれば、小文字 (.h、.c) に変更されます。業界標準では、C++ ファイルには大文字の拡張子 (.C) を使い、C ファイルには小文字の拡張子 (.c) を使います。現行のマイクロチップ社製コンパイラは、大文字と小文字を区別せずに、ソースコードの拡張子をそのまま使うため、ファイルをお客様のソース管理システムに登録する際に、問題が生じる可能性があります。

5.2.1 ウィザードを開く

このウィザードは、下記の 2 通りの方法で開く事ができます。

5.2.1.1 [START PAGE] から開く方法

- **[Start Page]** で、**[Learn & Discover]** タブを開き、「Dive In」セクションで「Import MPLAB Legacy Project」へのリンクを選択します。
- すると「Import Legacy Project」ウィザードが開きます。

5.2.1.2 [NEW PROJECT] ウィザードを使う方法

[New Project] ウィザードは、下記のいずれかの方法で開けます。

- **[Start Page]** で **[Learn & Discover]** タブを開き、「Dive In」セクションで「Create New Project」へのリンクを選択する
- [File]>[New Project] を選択する (または Ctrl+Shift+N)

[New Project] ウィザードが起動し、新規プロジェクトのセットアップ手順をガイドします。

- **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。
- すると「Import Legacy Project」ウィザードが開きます。

5.2.2 [Import Legacy Project] ウィザード

下記の手順により、MPLAB IDE v8 プロジェクトをインポートします。各ステップの手順を終えたら、**[Next>]** をクリックして次のステップへ進んでください。

- **Step 1 または 2. Import Legacy Project:** レガシー プロジェクトを入力するか選択します。
- **Step 2または3. 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 または 4. 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、使用するデバイスがデバッグ回路を内蔵しているかどうかは、ヘッダ仕様 (DS51292 またはオンラインヘルプ) で確認してください。上記を確認してからヘッダを使うかどうかを選択します。

- **Step 4 または 5. 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。

  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 または 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。

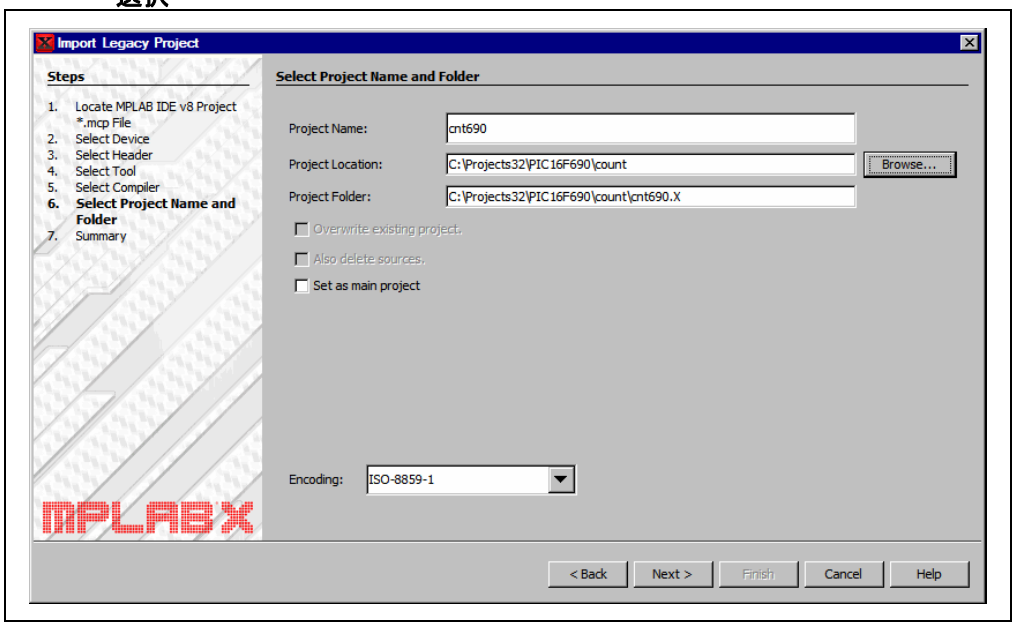
  選択したデバイスに対するツールのサポート レベルが、ツール名の横のマークで色分け表示されます。緑は完全サポート、黄はベータサポート、赤は非サポートを意味します。
- **Step 6 または 7. Select Project Name and Folder:** 双方のプロジェクトの保守性を保つために、既定値の名前とフォルダは変更しない事を推奨します。

  File Locations: ソースファイルは、この新規プロジェクトのプロジェクト フォルダにコピーされません。そのかわりに、v8 フォルダ内のソースファイルのパスを参照します。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。

  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、チェックボックスを ON にします。

  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。

図 5-1: [IMPORT LEGACY PROJECT] ウィザード – プロジェクト名とフォルダの選択



- **ステップ 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。

最後に **[Finish]** をクリックすると、作成したレガシー プロジェクトが [Project] ウィンドウ内に開きます。

5.3 プリビルド プロジェクト

[Import Image File]ウィザードを使って、プリビルドした読み込み可能イメージ(HEX、COF、ELF ファイル) からプロジェクトを作成します。このウィザードは、下記の 2 通りの方法で開く事ができます。

5.3.1 [Start Page] から開く方法

- **[Start Page]** で **[Learn & Discover]** タブを開き、「Dive In」セクションでリンク「Import Hex (Prebuilt) Project」を選択します。
- すると「Import Image File」ウィザードが開きます。

5.3.2 [New Project] ウィザードを使う方法

[New Project] ウィザードは、下記のいずれかの方法で開けます。

- **[Start Page]** で、**[Learn & Discover]** タブを開き、「Dive In」セクションで「Create New Project」へのリンクを選択する
- [File]>[New Project] を選択する (または Ctrl+Shift+N)

[New Project] ウィザードが起動し、新規プロジェクトのセットアップ手順をガイドします。

- **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。
- すると「Import Image File」ウィザードが開きます。

5.3.3 [Import Image File] ウィザード

下記の手順により、イメージファイルをインポートします。各ステップの手順を終えたら、**[Next>]** をクリックして次のステップへ進んでください。

- **Step 1 または 2. Import Image File:** イメージファイルの名前と格納場所を選択します。ダイアログを使って格納場所を選択できます。
- **Step 2または3. 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 または 4. 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、使用するデバイスがデバッグ回路を内蔵しているかどうかは、ヘッダ仕様 (DS51292 またはオンラインヘルプ) で確認してください。上記を確認してからヘッダを使うかどうかを選択します。
- **Step 4 または 5. 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。

  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 6 または 7. Select Project Name and Folder:** 新規プロジェクトの名前と格納場所を選択します。ブラウザ機能を使って格納場所を選択できます。
- **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。

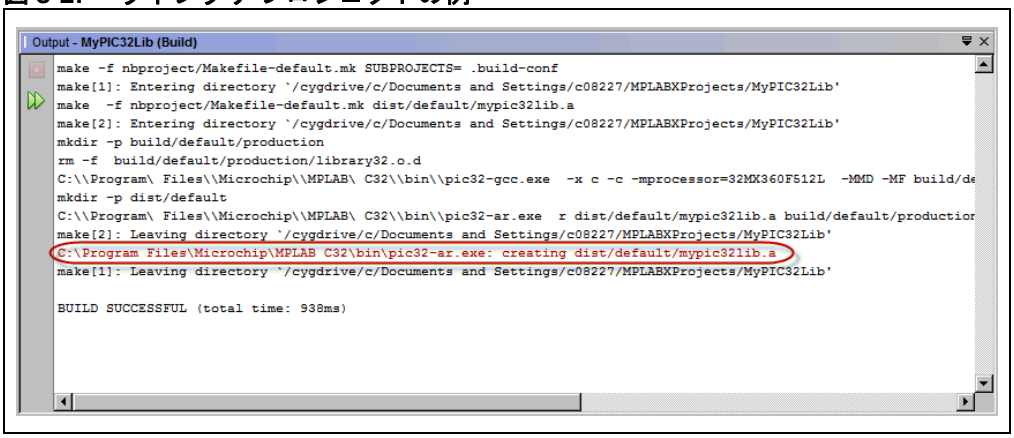
最後に [Finish] をクリックすると、作成したプロジェクトが [Project] ウィンドウ内に開きます。

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、**10.3.1.2「[Projects] ウィンドウ – [Project] メニュー」**を参照してください。

5.4 ライブラリ プロジェクト

IDE が生成した makefile を使うライブラリ プロジェクトを新規作成し、プロジェクトを実行ファイルではなくライブラリファイルとしてビルドします。

図 5-2: ライブラリ プロジェクトの例



まず、下記のいずれかの方法で [New Project] ウィザードを開きます。

- **[Start Page]** で **[Learn & Discover]** タブを開き、「Dive In」セクションでリンク「Create New Project」を選択する
- [File]>[New Project] を選択する (または Ctrl+Shift+N)

[New Project] ウィザードが起動し、新規プロジェクトのセットアップ手順をガイドします。各ステップの手順を終えたら、**[Next>]** をクリックして次のステップへ進んでください。

- **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、使用するデバイスがデバッグ回路を内蔵しているかどうかは、ヘッダ仕様 (DS51292 またはオンラインヘルプ) で確認してください。上記を確認してからヘッダを使うかどうかを選択します。
- **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。

  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. 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。

  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、赤は非サポートを意味します。
- **Step 6. Select Project Name and Folder:** 新規プロジェクトの名前と格納場所を選択します。ブラウザ機能を使って格納場所を選択できます。

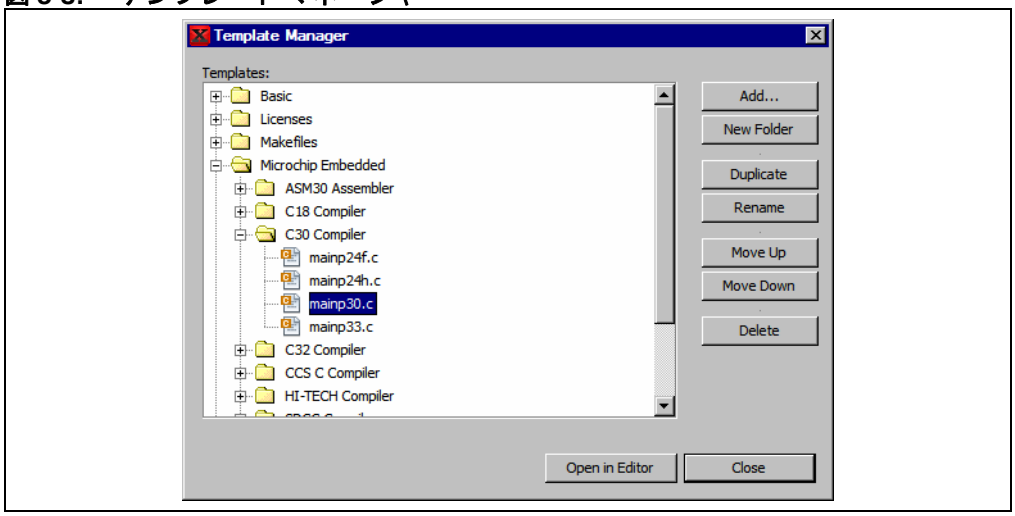
最後に [Finish] をクリックすると、作成したプロジェクトが [Project] ウィンドウ内に開きます。

5.5 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 (**4.8「新規プロジェクト ファイルの作成」**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 5-3: テンプレート マネージャ



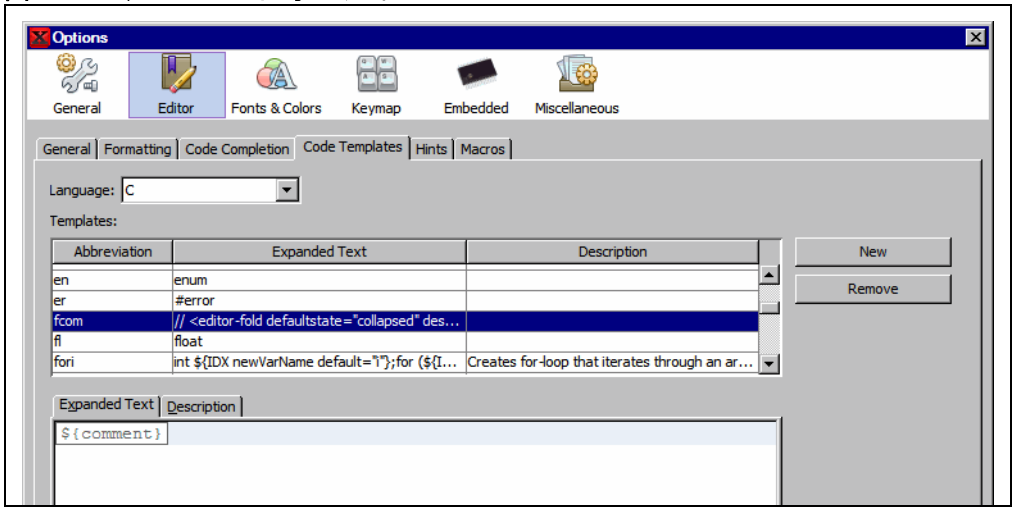
テンプレート オプションは、[Tools]>[Options] を選択し、**[Editor]** ボタンをクリックし、**[Code Templates]** タブで設定できます (図 5-4)。

「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] ウィンドウで、「fcom」をタイプ入力してから <Tab> キーを押すと、下記のテキストがソースコードに挿入されます。

```
// <editor-fold defaultstate="collapsed" desc="comment">
// </editor-fold>
```

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

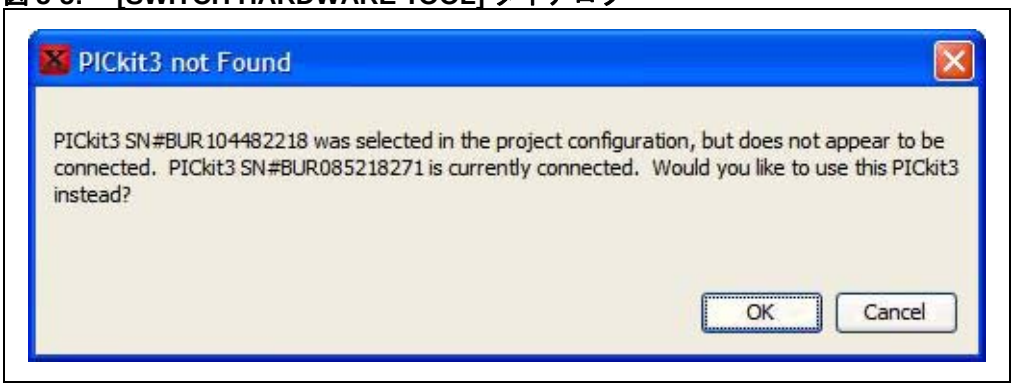
図 5-4: テンプレート オプション



5.6 ハードウェア ツールまたは言語ツールの切り換え

既存のプロジェクトを開いた時に、そのプロジェクト内で指定されているハードウェア ツールとは異なるツールを接続していると、現在接続しているハードウェア ツールをプロジェクト用ツールとして使うかどうかを確認するために、下記のダイアログが開きます。

図 5-5: [SWITCH HARDWARE TOOL] ダイアログ



複数のハードウェア ツールをプラグインし、[Project Properties] ダイアログ ([File]>[Project Properties]) を使って、それらのツールを切り換える事もできます。

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] ダイアログを使います。

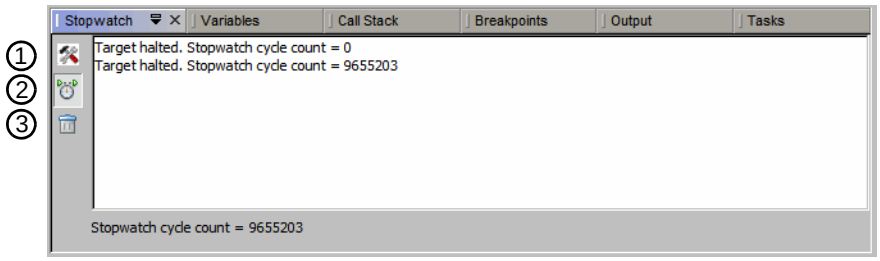
5.7 ストップウォッチの使用

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タイミング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]>[Breakpoints] 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. その後にプログラムをデバッグ実行すると、ストップウォッチのタイミング結果が表示されます。

図 5-6: [STOPWATCH] ウィンドウの内容



1. Properties
2. Reset Stopwatch on Run
3. Clear History

5.8 [DISASSEMBLY] ウィンドウの表示

このウィンドウは逆アセンブラコードを表示します。このウィンドウは、[Window]>[Output]>[Disassembly Listing File] を選択すると開きます。

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。このファイルを開くには、[File]>[Open File] を選択し、`ProjectName.lst` ファイルを選択します。

5.9 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

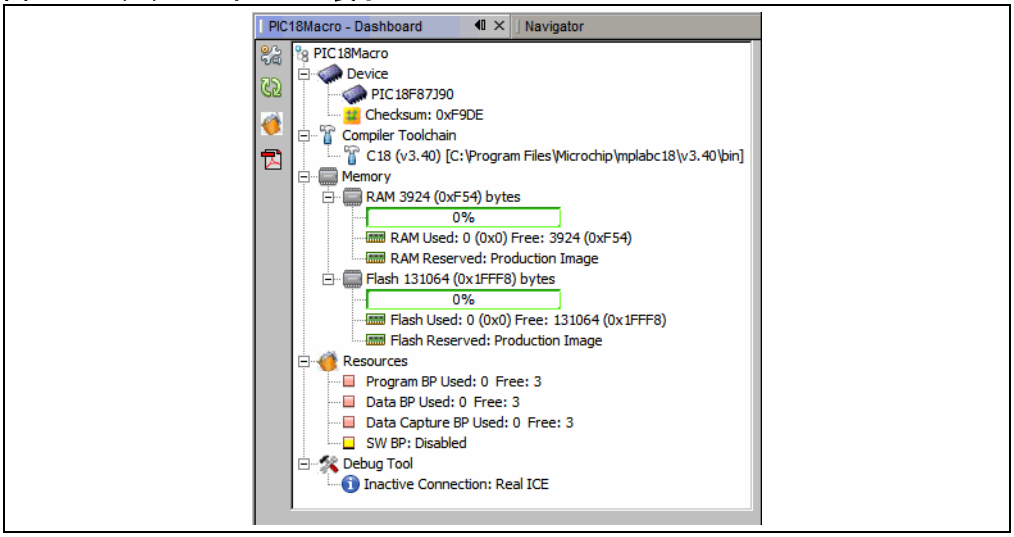
1. [Window]>[Output]>[Call Graph] を選択します。
2. [Code] ウィンドウで関数を選択します。
3. 選択 (強調表示) した関数を右クリックし、ドロップダウン メニューから「Show Call Graph」を選択します。

詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[C/C++/Fortran Project Basics]>[Navigating and Editing C/C++/Fortran Source Files]>[Using the Call Graph]) を参照してください。

5.10 ダッシュボードの表示

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

図 5-7: ダッシュボードの表示



ダッシュボードは、下記のいずれかのプロジェクトを表示します。

- メイン プロジェクトを選択していない ([Run]>[Set Main Project]>[None]) 場合、[Project] ウィンドウ内でアクティブなプロジェクト ([Project] 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 (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)

ダッシュボードの機能を下表に示します。

表 5-1: ダッシュボードのグループ

グループ	定義と内容
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 ビルド済みプログラムのチェックサム(これを表示するにはビルドする必要があります)
Compiler Toolchain	プロジェクトで使うコンパイラ ツールチェーン (コンパイラ、アセンブラ、リンカ等) 実行ファイルへのパスを含む
Memory	プロジェクトで使うメモリのタイプと容量、および、デバッグ用に予約されているメモリ
Resources (Breakpoints)	プロジェクト デバイスで利用できるハードウェア ブレークポイントの数 現在使用中のブレークポイントの数 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るかどうか
Debug Tool	デバッグツールの接続状態 ハードウェア デバッグツールの場合、デバッグ実行、実行、プログラミング中にのみ接続がアクティブになります。その他の状態での接続は非アクティブです。ツールの接続を常時アクティブにするには、[Tools]>[Options] 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」にチェックマークを付けます。 「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。

表 5-2: サイドバーのアイコン

アイコン	機能
	[Project Properties] ダイアログを表示します。
	デバッグツールのステータス表示を更新します (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します (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)。
	デバイスのデータシートを開きます (ローカルに保存したデータシートを開くか、ブラウザを開いてマイクロチップ社ウェブサイトでデータシートを検索できます)。

5.11 コードの改良

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良します。

> **Note:** * 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。

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 コードでは下記を行えます。

- 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を変更する

詳細は **6.4「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、プログラムの実行中に CPU 使用率、メモリ使用率、スレッド使用率を常時監視します。プロファイリング ツールは、C プロジェクトの実行時に、常時自動的に動作します。

プロファイラに関する詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Working with C/C++/Fortran Projects]>[Profiling C/C++/Fortran Projects]) を参照してください。

5.12 ソースコードの管理

MPLAB X IDE は、NetBeans プラットフォームの機能を補完するローカルなファイル履歴機能を内蔵しています。この機能は、ローカルなプロジェクトおよびファイル向けに、一般的なバージョン管理システムに類似したビルトインバージョン管理サポートを提供します。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。[Project] または [File] ウィンドウでファイルを右クリックすると、[Local History] オプションを選択できます。

ファイルのローカル履歴を表示する方法

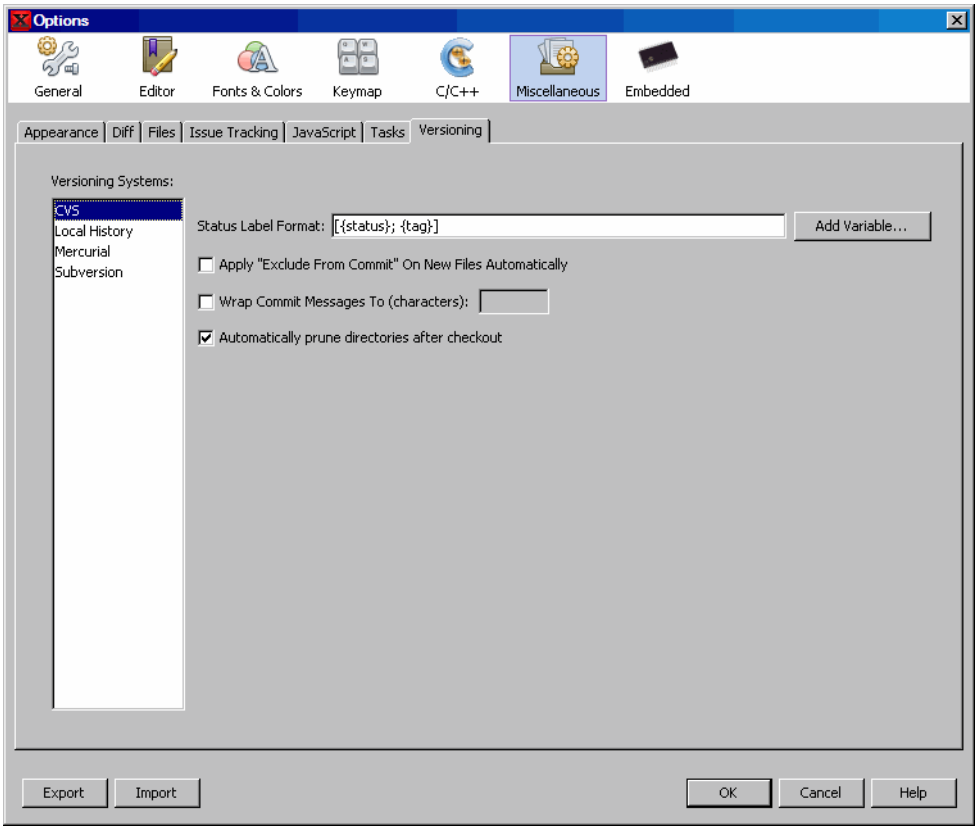
- [Project] または [File] ウィンドウでファイルを右クリックし、[Local History]>[Show Local History] を選択します。すると、過去にそのファイルに加えられた全ての変更の一覧が表示されます。
- [Project] または [File] ウィンドウでファイルを右クリックし、[Local History]>[Revert to] を選択します。すると、[Revert to] ダイアログが開き、そのファイルの過去のバージョンが全て表示されます。いずれかのバージョンを選択し、**[OK]** をクリックすると、そのバージョンが復元されます。

機能の完備したバージョン管理システムを使いたい場合、CVS、Subversion、Mercurial 向けのサポートを利用できます。

ソース管理機能は下記から利用できます。

- [Tools]>[Options] を選択して [Options] ウィンドウを開き、**[Miscellaneous]** アイコンをクリックし、**[Versioning]** タブを開いてバージョン管理プログラムをセットアップする (図 5-8 参照)。
- [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する
- [Window]>[Versioning] を選択して [Version Control] ウィンドウを開く

図 5-8:　バージョン管理のオプション



ローカルファイルの履歴および / またはソース管理機能の使用方法に関する詳細は、NetBeans ヘルプトピック ([IDE Basics]>[Version Control and File History]) を参照してください。

上記のソース管理プログラムの詳細は、下記を参照してください。

- CVS – http://www.nongnu.org/cvs/
- Subversion – http://subversion.tigris.org/
- Mercurial – http://mercurial.selenic.com/

5.13 コード開発とエラー追跡の連携

MPLAB X IDE は、チームサーバ (Kenai.com 等) 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。

> **Note:** * 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。

サポートするメニュー項目は下記の通りです。

- [Team]>[Team Server] – メインのチームサーバ メニューです (アカウントへのログイン、プロジェクトの作成または読み込み、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)。
- [File]>[Open Team Project] – 既存のチーム プロジェクトを開きます。
- [Windows]>[Team] – [Team] ウィンドウに自分のプロジェクトを表示します。
- [Windows]>[Chat] – チーム間の連携のために [Chat] ウィンドウを開きます。

問題追跡システム (Bugzilla™ と JIRA®、プラグインが必要) を使って、バグの追跡で連携します。これに関しては、下記のメニュー項目をサポートします。

- [Windows]>[Services] – 「Issue Tracker」を右クリックし、問題トラッカーを追加します。
- [Team]>[Find Issues] – [Issue Tracker] ウィンドウ内で、プロジェクトで使う問題トラッカーと評価基準を選択し、[Search] をクリックします。
- [Team]>[Report Issues] – [Issue Tracker] ウィンドウ内で、プロジェクトで使う問題トラッカーを選択し、問題の詳細を指定し、[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、NetBeans ヘルプトピック ([IDE Basics]>[Collaborative Development]) を参照してください。

これらのツールについて、さらに詳しく知りたい場合は、下記を参照してください。

- Kenai – http://kenai.com/
- Bugzilla – http://www.bugzilla.org/
- JIRA – http://www.atlassian.com/software/jira/

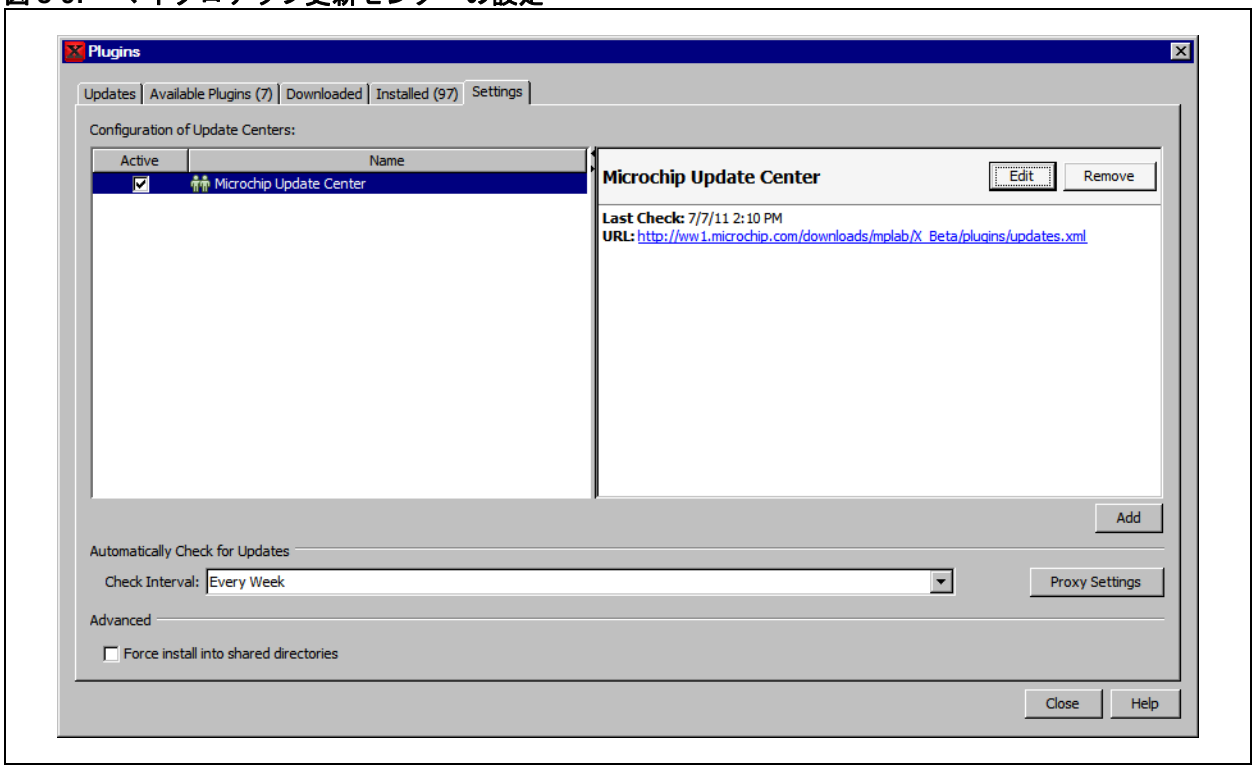
5.14 プラグインツールの追加

MPLAB IDE v8 のプラグインツール (DMCI 、MATLAB 等) は、プラグイン マネージャ ([Tools]>[Plugins]) を介して MPLAV X IDE でも利用できます。MPLAB IDE v8 では [Tools] メニューに含めていたキーボード マクロ機能は、実際にはエディタ機能であるため、MPLAB X IDE では [Edit] メニューに移動しています。

マイクロチップ社製プラグインを追加するには、プラグイン マネージャ内でマイクロチップ更新センターを事前に設定しておく必要があります。その手順は下記の通りです。

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. 「Name」に「Microchip Update Center」を入力します。
4. 下記の URL を入力します。
   http://ww1.microchip.com/downloads/mplab/X_Beta/plugins/updates.xml
5. **[OK]** をクリックします。

図 5-9: マイクロチップ更新センターの設定



MPLAB X IDE にプラグインを追加する方法

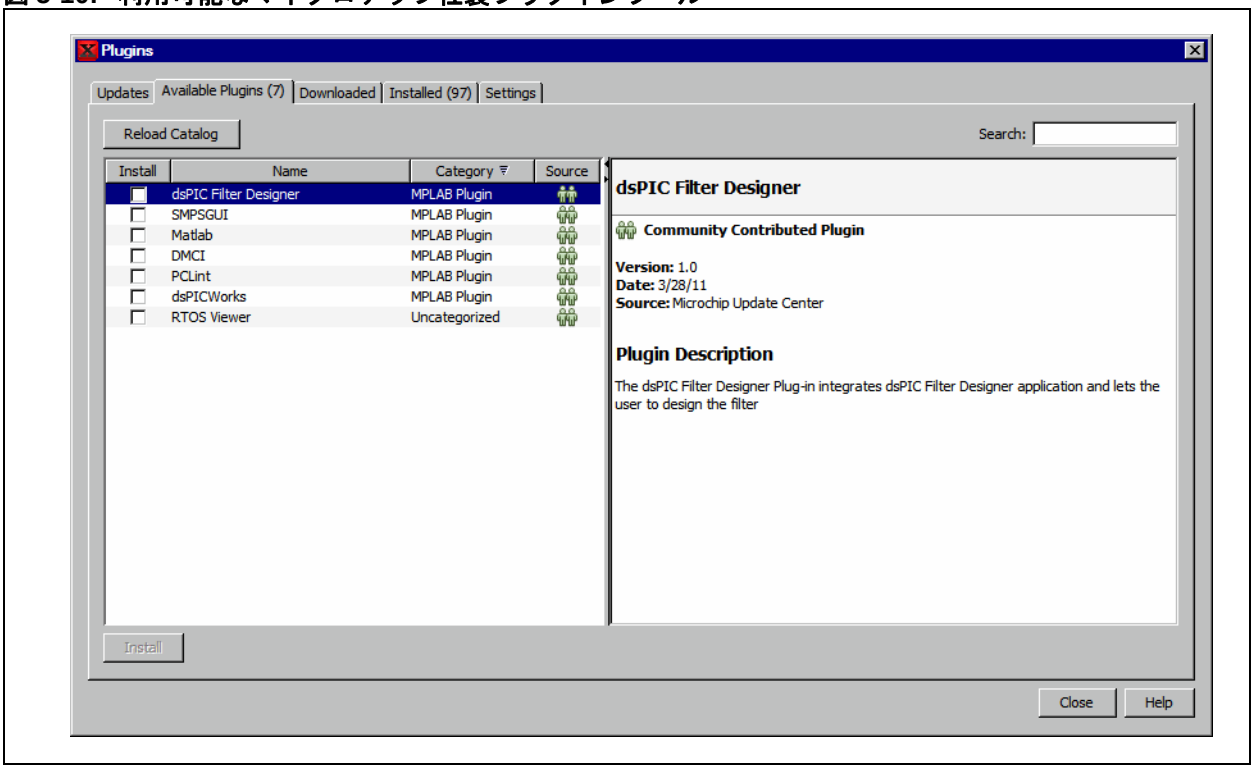
1. **[Available Plugins]** タブをクリックします。
2. チェックボックスでインストールするプラグインを選択します。
3. **[Install]** をクリックします。
4. 画面の指示に従って、プラグインをダウンロードおよびインストールします。

> **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。

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 5-10: 利用可能なマイクロチップ社製プラグインツール



NOTE:

Chapter 6. 高度な作業

本章では、MPLAB X IDE の高度な作業について解説します。これ以外の作業に関する説明は、**Chapter 4.「基本作業」**と **Chapter 5.「追加の作業」**に記載しています。

- 複数プロジェクトの使用
- 複数コンフィグレーションの使用
- NetBeans™ エディタ
- C コードのリファクタリング

6.1 複数プロジェクトの使用

MPLAB X IDE では、2 つ以上のプロジェクトを使って作業できます。

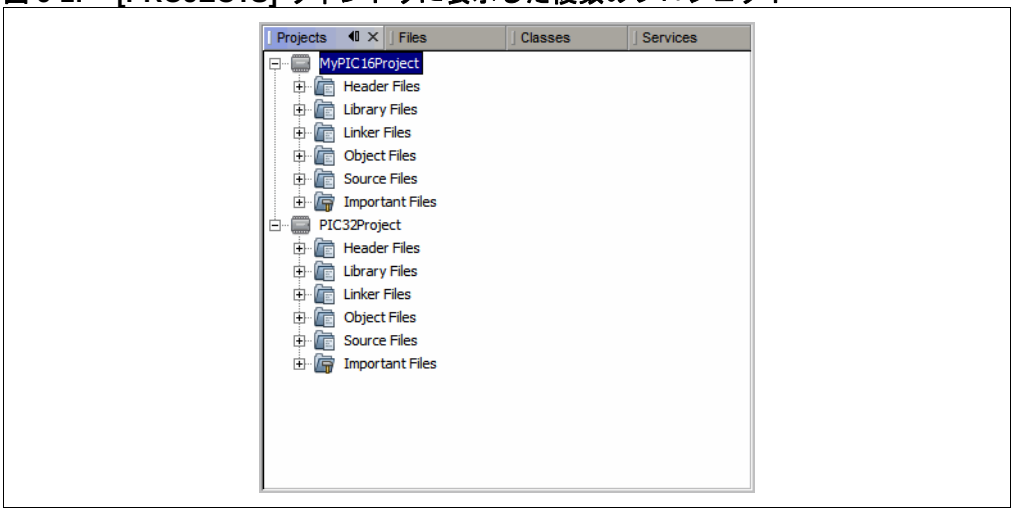
- 複数プロジェクトを使った作業 – プロパティの異なる複数のプロジェクトを使います。
- 複合プロジェクトによる作業 – プロパティが同一の複数のプロジェクトを使います。

6.1.1 複数プロジェクトを使った作業

2 つ以上のプロジェクトで同時に作業する必要がある場合、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、**10.3.1「[Projects] ウィンドウ」**を参照してください。

1 つの複雑なアプリケーションを複数のプロジェクトに分割した場合 (つまり全てのプロジェクトが同一アプリケーションの一部である場合) は、複合プロジェクト (**6.1.2「複合プロジェクトによる作業」**) の使用を検討してください。

図 6-1: [PROJECTS] ウィンドウに表示した複数のプロジェクト



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

下記のいずれかの方法により、1 つのプロジェクトをメインプロジェクトにできます。

- プロジェクト名を右クリックし、「Set as Main Project」を選択する
- [Run]>[Set Main Project] を選択する

メインプロジェクトの名前は**太字**で表示されます。

6.1.2 複合プロジェクトによる作業

複雑なコードを開発する場合、複数のプロジェクトを作成するのが最良の方法です。MPLAB IDE v8 では、複数のプロジェクトを開く事はできても、ビルド、デバッグ、実行を別々に行う必要がありました。MPLAB X IDE では、複数のプロジェクトを 1 つの複合プロジェクトとしてグループ化し、一括してビルド、デバッグ、実行できます。

複合プロジェクトは複数の異なるプロジェクト (「メンバー」と呼ぶ) で構成され、各プロジェクトは同一のデバイスおよびツール向けに設定されている必要があります。複合プロジェクトをビルドすると、その中のメンバーがビルドされ、IDE はこれらの中間生成結果を同一デバイスに書き込みます。

6.1.3 複合プロジェクトの作成方法

下記のいずれかの方法で [New Project] ウィザードを開きます。

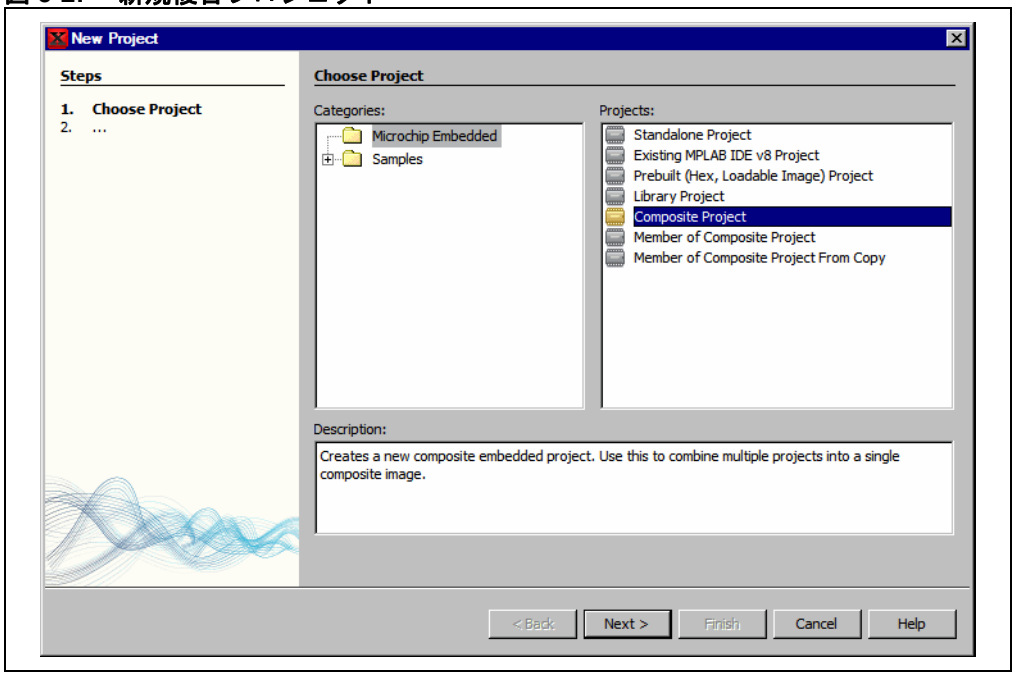
- **[Start Page]** の「**Learn & Discover**」タブを開き、「Dive In」セクションの「Create New Project」を選択する
- [File]>[New Project] を選択する (または Ctrl+Shift+N)

[New Project] ウィザードが起動し、新規プロジェクトのセットアップの手順をガイドします。

Step 1 では、プロジェクト カテゴリを選ぶよう指示されます。これは NetBeans のダイアログです。マイクロチップ社製品を使うには、「Microchip Embedded」を選択する必要があります。次にプロジェクトのタイプを選択します。複合プロジェクトの場合「Composite Project」を選択する必要があります。

[Next>] をクリックして次のダイアログに進みます。

図 6-2: 新規複合プロジェクト



Step 2 ～ Step 5 では、複合プロジェクトを構成する全てのメンバー プロジェクトに共通のデバイスヘッダ (使用可能な場合)、デバッグツール、言語ツールパッケージを選択します。これらの設定項目に関する詳細は、**4.2「新規プロジェクトの作成」**を参照してください。

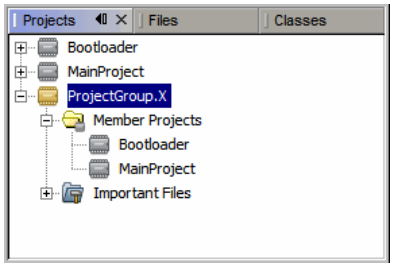
Step 6 では、複合プロジェクトの名前と格納場所を指定します。**[Finish]** をクリックして手順を終了します。

複合プロジェクトにメンバーを追加する方法

複合プロジェクトの下の「Member Projects」フォルダを右クリックし、下記のオプションを選択します。

- Add New – プロジェクトを新規作成し、複合プロジェクトに追加します。既定値では、作成したプロジェクトは複合プロジェクトのフォルダ内に保存されます。
- Add Existing – 既存のプロジェクトを複合プロジェクトに追加します。このプロジェクトは参照されるだけであり、保存場所は移動しません (コピーされません)。
- Add Copy of Exisiting – 既存プロジェクトのコピーを複合プロジェクトに追加します。既定値では、プロジェクトのコピーは複合プロジェクトのフォルダ内に保存されます。

図 6-3: 複合プロジェクトの例



複合プロジェクトのビルド方法

1. まだ開いていない場合は複合プロジェクトを開き、これをメイン プロジェクトとして設定します (複合プロジェクトの名前を右クリックし、「Set as Main Project」を選択する)。
2. 全てのメンバー プロジェクトを開きます (「Member Projects」フォルダ内の各メンバーをダブルクリックする)。複合プロジェクトを開いただけでは、メンバー プロジェクトは開きません。
3. 複合プロジェクトをビルドします (複合プロジェクト名を右クリックし、「Build」を選択する)。全てのメンバー プロジェクトがビルドされ、1 つの実行ファイルに結合されます。

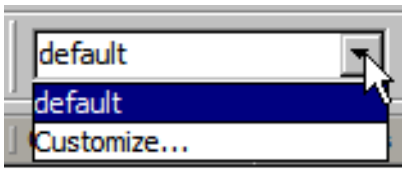
6.2 複数コンフィグレーションの使用

MPLAB X IDE では、同一プロジェクトで複数の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 (マイクロチップ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、既定値のコンフィグレーションが作成されます。ユーザ独自のコンフィグレーションの作成は、下記のいずれ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 (図 6-4) から「Customize」を選択して、[Project Properties] ダイアログを開く
- プロジェクト名を右クリックし、「Properties」を選択して、[Project Properties] ダイアログを開く

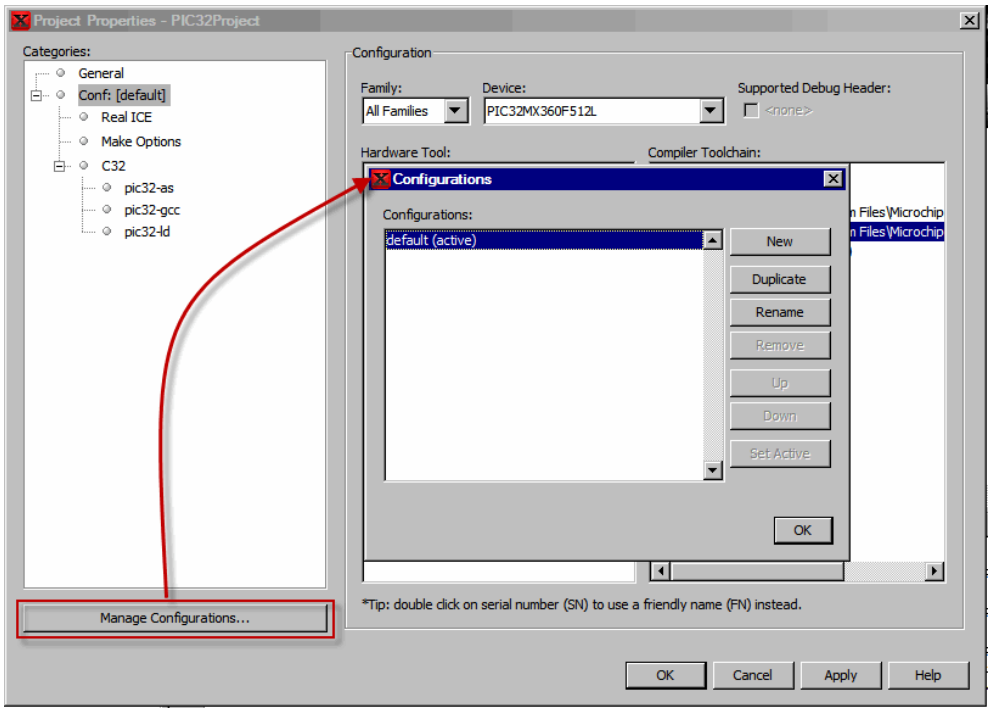
図 6-4:　[PROJECT CONFIGURATION] ドロップダウン ボックス



[Project Properties] ダイアログで **[Manage Configurations]** をクリックして、[Configurations] ダイアログ (図 6-5) を開きます。このダイアログでは、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。

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できます。

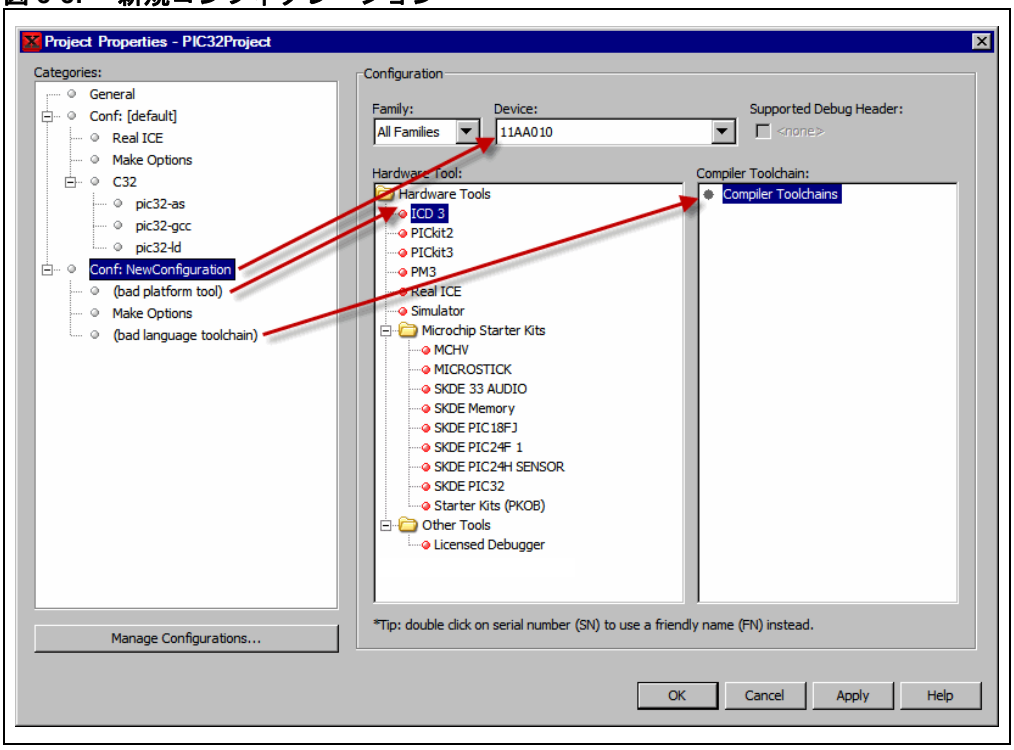
図 6-5:　[PROJECT PROPERTIES] – [CONFIGURATIONS] ダイアログ



6.2.1　新規コンフィグレーションの追加

新規コンフィグレーションを追加した場合、デバイスと言語ツールを [Project Properties] ダイアログで割り当てる必要があります。

図 6-6:　新規コンフィグレーション



6.2.2　　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。コンフィグレーションの複製は、デバッグ用に便利です。

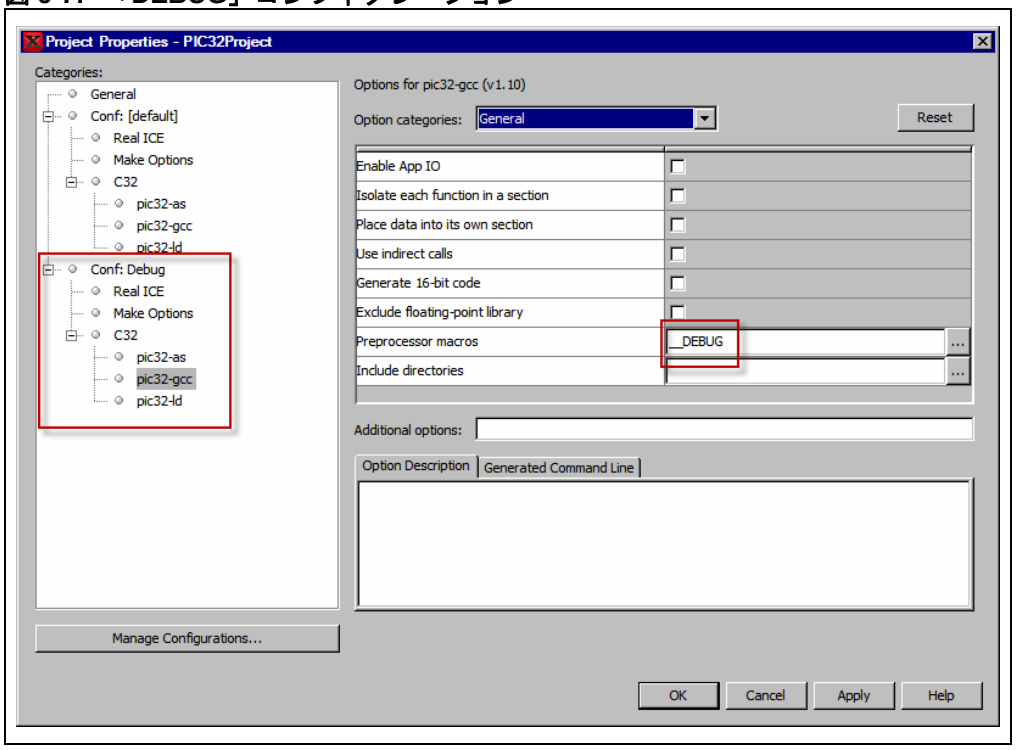
デバッグ コンフィグレーションの設定方法

1. [Configuration] ダイアログで、いずれかのプロジェクト コンフィグレーションを選択してから **[Duplicate]** をクリックします。
2. **[Rename]** をクリックし、[New Configuration Name] ダイアログで「Debug」と入力します。
3. **[OK]** を 2 回クリックして [Project Properties] ダイアログに戻ります。以上により、「Debug」コンフィグレーション (Conf:Debug) が表示されます。
4. ツールチェーン内のコンパイラまたはアセンブラをクリックします。「Option categories」で「General」オプションを選択し、その下の「Preprocessor macro」テキストボックスをクリックします。
5. 「reprocessor Macros」テキストボックスに「__DEBUG」と入力し、**[OK]** をクリックします。

以上により、デバッグを行う際に「Debug」コンフィグレーションに切り換える事ができるようになります。プロセッサマクロは、下記のコンディショナル テキスト内で使えます。

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

図 6-7: 「DEBUG」コンフィグレーション



6.3 NETBEANS™ エディタ

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。新しいコードの作成や既存コードの編集には、NetBeans プラットフォームが提供するビルトイン エディタを使います。

このエディタに関する一般情報は、NetBeans ヘルプトピック ([IDE Basics]>[Basic File Features]) を参照してください。エディタに関連する C コンパイラ情報は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Working with C/C++/Fortran Projects]>[Editing Source Files in C/C++/Fortran Projects]>[About Editing C and C++ Files]) を参照してください。

エディタ機能は下記から利用できます。

- [File] メニュー (**9.2.1「[File] メニュー」**参照) を使ってファイルを [Editor] ウィンドウに開く
- [Edit] メニュー (**9.2.2「[Edit] メニュー」**参照) を使って、編集コマンドを実行する
- 各ファイルのエディタ ウィンドウの上段にあるツールバー (**9.3.9「[Editor] ツールバー」**参照) から一部の編集コマンドを選択する
- ウィンドウを右クリックしてコンテクスト メニューを開き、その他のコマンドを選択する

エディタ プロパティの設定方法

1. [Tools]>[Options] 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セットアップします。
3. **[Fonts and Colors]** ボタンをクリックします。タブをクリックして色オプションを設定します。

エディタ機能に関する参照先

エディタ機能	参照先
ユニコードのサポート	[IDE Basics]>[Configurating the IDE]>[About Project Encodings]
構文に基づくコードの色分け	[Tools]>[Options]、**[Fonts and Colors]** ボタン、**[Syntax]** タブ
コードをタイプ入力した時のエラーフラグ	[C/C++/Fortran Development]>[Working with C/C++/Fortran Projects]>[Editing Source Files in C/C++/Fortran Projects]>[Error Highlighting]
シンボル、エラー等の色付きマークによる識別	[Tools]>[Options]、**[Fonts and Colors]** ボタン、**[Annotations]** タブ
スマートなコード補完機能によるアドバイスやヒント	[Tools]>[Options]、**[Editor]** ボタン、**[Code Completion]** タブ
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。これは検索を関数内だけに制限する事もできます (例 : ローカル変数 `i` の検索)	[Find Usages] ダイアログ
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	**5.9「コールグラフの表示」**
ソースコード内でタスクキーワード (例 : `//TODO`) を使ってコメントを作成し、それらのタスクをスキャンして [Task] ウィンドウに自動的に追加する	[IDE Basics]>[Basic File Features]>[Working with Task Lists]
バージョン管理システムを使わずに、ファイル履歴機能を使って最近の変更を表示し、古いバージョンを復元する	**5.12「ソースコードの管理」** – ローカル履歴
「Go to File」、「Go to Type」、「Go to Symbol」、「Go to Header」、「Go to Declaration」等を使って、ファイルやファイル内のアイテムに素早く移動する	**9.2.4「[Navigate] メニュー」**
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	**6.4「C コードのリファクタリング」**

6.4 C コードのリファクタリング

> **Note:** * この機能については、**[Start Page]** の **[My MPLAB 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。

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。リファクタリング後のプログラムは、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、下記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
- リファクタリングした変更の取り消し
- 関数の使用箇所の検索
- 関数またはパラメータの名前変更
- C コードの移動、コピー、安全な削除

6.4.1 [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 (**9.2.6「[Refactor] メニュー」**参照) から利用できます。

6.4.2 リファクタリングした変更の取り消し

[Refactor] メニュー内のコマンドを使って適用した変更は、下記の手順で取り消す事ができます。リファクタリングを取り消すと、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箇所を元に戻します。

リファクタリングを取り消すには下記を行います。

1. メインメニュー バー内の [Refactor] メニューを開く
2. [Undo] を選択する
   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。

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] は利用できなくなります。

6.4.3 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、いずれかの関数名を右クリックし、[Find Usages] を選択します (または <Alt-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、ファイルのその位置へ移動する方法

- [Usages] ウィンドウの左ウィンドウ枠内で、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

下記の 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、検索するテキスト (文字列) をタイプ入力します。

[Navigator] ウィンドウ内で、いずれかの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、コンテクストメニューから [Navigate]>[Go To Declaration] を選択します。

6.4.4 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、下記を行います。

1. [Source Editor] 内で、名前を変更する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、新しい名前をタイプ入力します。
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、変更される予定のコード行が表示されます。各コード行の横のチェックマークをクリアすると、その行は変更されません。
4. **[Do Refactoring]** をクリックして、選択した変更を適用します。

名前を変更するアイテムにカーソルを置いてから<Ctrl-R>を押して新しい名前をタイプ入力すると、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]** をクリックします。

6.4.5 C コードの移動、コピー、安全な削除

これらの機能は C++ コード専用であり、現在サポートしていません。

NOTE:

Chapter 7. 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は対処できなかった場合はマイクロチップ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。

- USB ドライバのインストールに関する問題
- 異なるプラットフォームで使用する場合の問題
- MPLAB X IDE の問題
- NetBeans プラットフォームの問題
- エラー
- フォーラム

7.1 USB ドライバのインストールに関する問題

USB ドライバの適切なインストール方法については、 **2.2「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については、**2.2.2.4「ツールの通信の問題」**を参照してください。

7.2 異なるプラットフォームで使用する場合の問題

MPLAB X IDE を複数の異なるプラットフォーム (Windows、Mac、Linux OS) で使う場合、下記の点に注意してください。

- 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 OS プラットフォーム以外では使えません。
  例 : `#include headers/myheader.h`.
- Linux OS 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。

7.3 MPLAB X IDE の問題

MPLAB IDE v8 プロジェクトのインポート

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。

7.4 NETBEANS プラットフォームの問題

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 NetBeans ウェブサイト (www.netbeans.org) をご覧ください。

下記も参照してください。

http://netbeans.org/community/releases/69/relnotes.html

7.5 エラー

IDE は各種の形態でエラーを表示しますが、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。エラーを示すアイコン上にカーソルを置くと、問題の内容を示すポップアップ テキストが開きます。テキスト メッセージについては、オンラインヘルプを参照して該当するエラーを検索してください。

一部のエラー メッセージを以下に挙げます。

Couldn’t reserve space for cygwin’s heap (Windows 7 OS)

MPLAB X IDE は、make 処理に Cygwin MinGW を使います。Windows 7 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。

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、[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」に下記の値を入力します。

- 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。

Could not access the URL through the external browser.Check the browser configuration

MPLAB X IDE 内で、[Tools]>[Options] を選択し、**[General]** タブを開きます。[Web Browser] ドロップダウン リストで、使用するブラウザを選択して、**[OK]** をクリックします。

詳細は下記を参照してください。

http://netbeans.org/bugzilla/show_bug.cgi?id=21236

http://netbeans.org/bugzilla/show_bug.cgi?id=38211

http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4646857

7.6 フォーラム

上記では問題の解決策が見つからない場合、下記のマイクロチップ社フォーラムもご覧ください。

http://www.microchip.com/forums/f238.aspx

このフォーラムには、問題に関するディスカッションや最近投稿された解決策が掲載されています。

Chapter 8. MPLAB X IDE と MPLAB IDE v8 の相違点

MPLAB X IDE は MPLAB IDE v8 以前のバージョンから大きく変更されています。本章には、MPLAB X IDE に移行する際に役立つ情報を記載しています。

- 主な相違点
- 機能上の相違点
- メニューの相違点
- ツールのサポートに関する相違点

8.1 主な相違点

1. **MPLAB X IDE はオープンソースのマルチ プラットフォームに基づきます。**

   MPLAB X IDE はオープンソースのクロスプラットフォーム NetBeans IDE に基づきます。サードパーティーは、プラグインとして機能を容易に追加できます。

   MPLAB IDE v8 は独自のプラットフォームと Windows に基づきます。サードパーティーは、MPLAB 開発グループからの設計情報に基づいて機能を v8 に追加できます。

2. **MPLAB X IDE はプロジェクトに基づきます (ワークスペースを使いません)。**

   MPLAB X IDE では、プロジェクトを作成する必要があります。プロジェクトを作成するには、デバイスを選択し、言語ツール、デバッグツール、プログラミング ツールとその他のプロジェクト仕様を選択および設定する必要があります。これにより、アプリケーションの開発に必要な全てのアイテムを確保します。プロジェクトをグループ化する機能を備えるため、複数のプロジェクトを処理できます。

   MPLAB IDE v8 はデバイスに基づきます。v8 でも、プロジェクトを使ってアプリケーションを作成する事を強く推奨しますが、プロジェクトの使用は必須ではありません。ワークスペースは、一部の設定情報 (複数プロジェクトのグループ化等) を格納するために使われます。

3. **MPLAB X IDE では、複数のツールを選択できます。**

   例 1: ターゲットボードに接続した複数台の MPLAB ICD 3 デバッガを、PC の複数の USB ポートに接続し、[Project Properties] ダイアログでこれらのデバッガを容易に切り換える事ができます (デバッガはシリアル番号 (SN) で識別)。

   例 2: ターゲットボードに接続した 1 台の MPLAB ICD 3 デバッガと 1 台の MPLAB PM3 プログラマを、それぞれ PC の USB ポートに接続し、[Project Properties] ダイアログでこれらのツールを容易に切り換える事ができます。

   MPLAB IDE v8 では、複数のツールを選択する事はできません。

4. **MPLAB X IDE では、言語ツールの複数のバージョンを選択できます。**

   MPLAB X IDE では、言語ツールの複数の異なるバージョンを選択できます。

   例 : PIC18 MCU 向け MPLAB C コンパイラの 2 つのバージョンをインストールし、

   [Project Properties] ダイアログでコンパイラ ツールチェーンのバージョンを容易に切り換える事ができます。

   MPLAB IDE v8 では、複数の言語ツールバージョンを選択する事はできません。

8.2 機能上の相違点

MPLAB X IDE は NetBeans に基づくため、独自アプリケーションであった MPLAB IDE v8 以前のバージョンとは多くの点で異なります。これらの相違点を完全に理解するには、本書の下記の章を参照してください。

- **Chapter 2.「使用前の準備」**
- **Chapter 3.「チュートリアル」**
- **Chapter 5.「追加の作業」**

機能的相違点を要約して以下に記載します。

- v8 では、コードを実行またはデバッグ実行する前に、[Build Configuration] ドロップダウン ボックスを使うか、あるいはビルド、プログラミング、実行を別々に行う必要がありましたが、MPLAB X IDE ではその必要はありません。コード実行の場合、[Run]>[Run Project] を選択するだけでビルド、ターゲットデバイスのプログラミング (ハードウェア ツールの場合)、コードの実行を行えます。デバッグ実行の場合、[Debug]>[Debug Project] を選択するだけでビルド、ターゲットデバイスのプログラミング (ハードウェア ツールの場合)、デバッグモードでのコード実行を行えます。**3.12「コードの実行」**と **3.13「コードのデバッグ実行」**を参照してください。
- デバイスをプログラミングする際に、上記の [Run Project] を使うとビルドとプログラミング後即座にコードを実行できますが、[Hold in Reset] を使ってビルドとプログラミング後にデバイスをリセット状態にホールドする事もできます。**3.18「デバイスのプログラミング」**を参照してください。
- MPLAB X IDE では、複数のビルド コンフィグレーションを使えます。従来のバージョンでは、[Build Configuration] ドロップダウン ボックスから「Release」または「Debug」のいずれかを選択し、コード内で __DEBUG を使いました。MPLAB X IDE ではデバッグ コンフィグレーションを自由に設定する事も、従来バージョンと同じ手順に設定する事もできます。**6.2.2「複製したコンフィグレーションの追加」**の説明に従ってユーザ独自のデバッグ コンフィグレーションと __DEBUG マクロを作成する事により、MPLAB IDE v8 の機能を再現できます。
- コンフィグレーション ビットは、コード内で設定する必要がありますが、デバッグ中に一時的に変更する事もできます (**4.20「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 / 変更」**参照)。
- [Memory] ウィンドウは 全て [Window]>[PIC Memory Views] から開くようになりました。各ウィンドウは、全てのマイクロチップ社製デバイスのメモリタイプとメモリ フォーマット向けにカスタマイズできます。**3.17「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」**を参照してください。
- ブレークポイント リソース ツールバー、チェックサム ツールバー、メモリゲージは全て 1 箇所 ([Window]>[Dashboard]) に集約しました。**5.10「ダッシュボードの表示」**を参照してください。
- その他、NetBeans の豊富な編集およびデバッグ機能も利用できます。詳細は、NetBeans のヘルプを参照してください。

8.3 メニューの相違点

下記のテーブルに、MPLAB IDE v8 から MPLAB X IDE へのメニューの変更点を示します。機能的な変更により、MPLAB X IDE と MPLAB IDE v8 のメニュー項目は一対一に対応しません。主な相違点を表の「コメント」列に記載しています。

MPLAB X IDE で追加されたメニュー項目もあります。詳細は、NetBeans ヘルプトピック「IDE Basics」を参照してください。

表 8-1: [FILE] メニューの相違点

MPLAB® IDE v8	MPLAB® X IDE	コメント
New	File>New File	ファイル ウィザードで関連するプロジェクトを選択
Add New File to Project		
Open	File>Open File	
Close	File>Close	
Save	File>Save	
Save As	File>Save As	
Save All	File>Save All	
Open Workspace	N/A	全てのデータをプロジェクトに保存
Save Workspace	N/A	
Save Workspace As	N/A	
Close Workspace	N/A	
Import	File>Import	
Export	タブ形式の [Project] ウィンドウ内でファイルを右クリックし「Export Hex」を選択	
Print	File>Print File>Print to HTML File>Page Setup	
Recent Files	File>Open Recent File	
Recent Workspaces	N/A	全てのデータをプロジェクトに保存
Exit	File>Exit	

表 8-2: [EDIT] メニューの相違点

MPLAB® IDE v8	MPLAB® X IDE	コメント
Undo	Edit>Undo	
Redo	Edit>Redo	
Cut	Edit>Cut	
Copy	Edit>Copy	
Paste	Edit>Paste Edit>Paste Formatted	
Delete	Edit>Delete	
Select All	Edit>Select All Edit>Select Identifier	
Find	Edit>Find Edit>Find Selection	
Find Next	Edit>Find Next Edit>Find Previous	

表 8-2: [EDIT] メニューの相違点 (続き)

MPLAB® IDE v8	MPLAB® X IDE	コメント
Find In Files	Edit>Find in Projects Edit>Replace in Projects	
Replace	Edit>Replace	
Go To	Navigate>Go to Line Navigate>Go to Declaration	
Go To Locator	Navigate>Go to Symbol	
Go Backward	Navigate>Back	
Go Forward	Navigate>Forward	
External DIFF	Tools>Diff	下記も参照 : Tools>Options、 Miscellaneous、[Diff] タブ
Advanced	Source>Format Source>Shift Left/Right Source>Move Up/Down Source>Toggle Comment	
Bookmarks	Navigate>Toggle Bookmark Navigate>Next Bookmark Navigate>Previous Bookmark	
Properties	Tools>Options>Editor tab Tools>Options>Fonts & Colors	

表 8-3: [VIEW] メニューの相違点

MPLAB® IDE v8	MPLAB® X IDE	コメント
Project	Window>Projects	
Output	Window>Output>Output	
Toolbars	View>Toolbars	
CPU Registers*	Window>PIC Memory Views>CPU	
Call Stack	Window>Debugging>Call Stack	
Disassembly Listing	Window>Output>Disassembly Listing File	
EEPROM	Window>PIC Memory Views>EE Data Memory	ウィンドウ内でカスタマイズ
File Registers	Window>PIC Memory Views>Data Memory	
Flash Data	Window>PIC Memory Views>Data Memory	
Hardware Stack	Debug>Stack	
LCD Pixel	未実装	
Locals	Window>Debugging>Variables	
Memory*	Window>PIC Memory Views>Other Memory	ウィンドウ内でカスタマイズ
Program Memory	Window>PIC Memory Views>Flash Memory	
SFR/Peripherals*	Window>PIC Memory Views>Peripherals	

表 8-3: [VIEW] メニューの相違点 (続き)

MPLAB® IDE v8	MPLAB® X IDE	コメント
Special Function Registers	Window>PIC Memory Views>SFRs	
Watch	Window>Debugging>Watches	下記も参照 : Debug>New Watch
Memory Usage Gauge	Window>Dashboard	メモリ使用率は [PE] ウィンドウに表示
Trace	Window>Debugging>Trace	下記も参照 : File>Project Properties
* PIC32 MCU のみ		

表 8-4: [PROJECT] メニューの相違点

MPLAB® IDE v8	MPLAB® X IDE	コメント
Project Wizard	File>New Project	常に新規プロジェクト用に使用
New	File>New Project	
Open	File>Open Project File>Open Recent Project	
Close	File>Close Project	
Set Active Project	Run>Set Main Project	
Quickbuild	N/A	全ての開発にはプロジェクトが必要
Package in .zip	タブ形式の [Project] ウィンドウ内でプロジェクトを右クリックし、「Package」を選択	
Clean	タブ形式の [Project] ウィンドウ内でプロジェクトを右クリックし、「Clean and Build」を選択	
Locate Headers	N/A	
Export Makefile	未実装	
Build All	Run>Run Project Debug>Debug Project	プログラマによるビルド / 実行 デバッガによるビルド / 実行
Make		
Build Configuration	File>Project Group Run>Set Project Configuration	
Build Options	File>Project Properties	言語ツールのセットアップ
Save Project	File>Save File>Save All	
Save Project As	File>Save As	
Add Files to Project	タブ形式の [Project] ウィンドウ内でフォルダを右クリックし、「Add Existing Item」を選択	
Add New File to Project	File>New File	
Remove File from Project	タブ形式の [Project] ウィンドウ内でファイルを右クリックし、「Remove From Project」を選択	
Select Language Toolsuite	File>Project Properties	
Set Language Tool Locations	Tools>Options, Embedded	

表 8-4: [PROJECT] メニューの相違点 (続き)

MPLAB® IDE v8	MPLAB® X IDE	コメント
Version Control	Tools>Options、Miscellaneous、[Versioning] タブ	下記も参照 : [Team] メニュー、Window>Versioning

表 8-5: [DEBUGGER] メニューの相違点

MPLAB® IDE v8	MPLAB® X IDE	コメント
Select Tool	File>New Project File>Project Properties	新規に選択 既存の選択を変更
Clear Memory	TBD	プログラマが使用
Run	Run>Run Project Debug>Debug Project Debug>Continue	プログラマによる実行 デバッガによる実行 一時停止後の実行再開
Animate	N/A	
Halt	Debug>Pause	
Step Into	Debug>Step Into	
Step Over	Debug>Step Over	
Step Out	Debug>Step Out	
Reset	Debug>Reset	
Breakpoints	Debug>New Breakpoint Debug>Toggle Breakpoint	
Settings	File>Project Properties	
Stopwatch	Window>Debugging>Stopwatch	

表 8-6: [PROGRAMMER] メニューの相違点

MPLAB® IDE v8	MPLAB® X IDE	コメント
Select Programmer	File>New Project File>Project Properties	新規に選択 既存の選択を変更
Enable Programmer	N/A	選択するとプログラマは有効になる
Disable Programmer		
Program	Program Target Project	[Run] ツールバーから選択
Verify	未実装	
Read	Upload Target Project	[Run] ツールバーから選択
Blank Check All	未実装	
Blank Check OTP	未実装	
Erase Flash Device	未実装	
Reset Program Statistics	未実装	
Download OS	未実装	
Settings	File>Project Properties	

表 8-7: [TOOLS] メニューの相違点

MPLAB® IDE v8	MPLAB® X IDE	コメント
Data Monitor and Control Interface (DMCI) and other plugins	Tools>Plugins Tools>Embedded	ツールをインストール ツールを使用
MPLAB® Macros	Edit>Start Macro Recording Edit>Stop Macro Recording	
RTOS Viewer	Tools>Plugins Tools>Embedded	ツールをインストール ツールを使用

表 8-8: [CONFIGURE] メニューの相違点

MPLAB® IDE v8	MPLAB® X IDE	コメント
Select Device	File>New Project File>Project Properties	新規に選択 既存の選択を変更
Configuration Bits	Window>PIC Memory Views>Configuration Bits	
External Memory	Window>PIC Memory Views>Other Memory	ウィンドウ内でカスタマイズ
ID Memory	Window>PIC Memory Views>Other Memory	ウィンドウ内でカスタマイズ
Settings	Tools>Options	

表 8-9: [WINDOW] メニューの相違点

MPLAB® IDE v8	MPLAB® X IDE	コメント
Close All	Window>Close All Documents	
Cascade	サポートせず	文書管理については [Window]>[Documents] を参照
Tile Horizontally		
Tile Vertically		
Arrange Icons		
Window Sets		
Create Window Set		
Destroy Window Set		
Recent Windows		

表 8-10: [HELP] メニューの相違点

MPLAB® IDE v8	MPLAB® X IDE	コメント
Topics	Help>Contents	PDF の表示については **11.1「[Projects] ウィンドウの表示」**参照
Release Notes	Help>Online Docs and Support	
Driver Installation	Help>Online Docs and Support	
Check for Updates	Help>Check for Updates	
Web Links	Help>Online Docs and Support	
About MPLAB® IDE	Help>About	

8.4 ツールのサポートに関する相違点

下のマイクロチップ社製開発ツールの一覧表に、MPLAB X IDE におけるサポートの有無を示します。

表 8-11: MPLAB X IDE におけるツールのサポート

開発ツール	MPLAB® X IDE でのサポート	
	Yes	No
MPLAB® ICD 2		X
MPLAB® ICD 3	X	
MPLAB® ICE 2000		X
MPLAB® ICE 4000		X
MPLAB® REAL ICE™	X	
PICkit™ 1		X
PICkit™ 2	X	
PICkit™ 3	X	
MPLAB® PM3	X	
PRO MATE II		X
PICSTART® Plus		X
MPLAB® VDI		X
プラグイン		
AN851 ブートローダ		X
AN901/908		X
DMCI	X	
dsPIC® フィルタデザイン	X	
MATLAB®	X	
PC-Lint	X	
dsPIC30F SMPS 降圧型コンバータ	X	
dsPIC33F SMPS 昇降圧型コンバータ	X	
セグメント ディスプレイ デザイナ *	X	
グラフィカル ディスプレイ デザイナ *	X	
メモリ スタータキット	X	
KEELOQ® プラグイン		X
* 開発中のツール		

NOTE:

Chapter 9. デスクトップの詳細

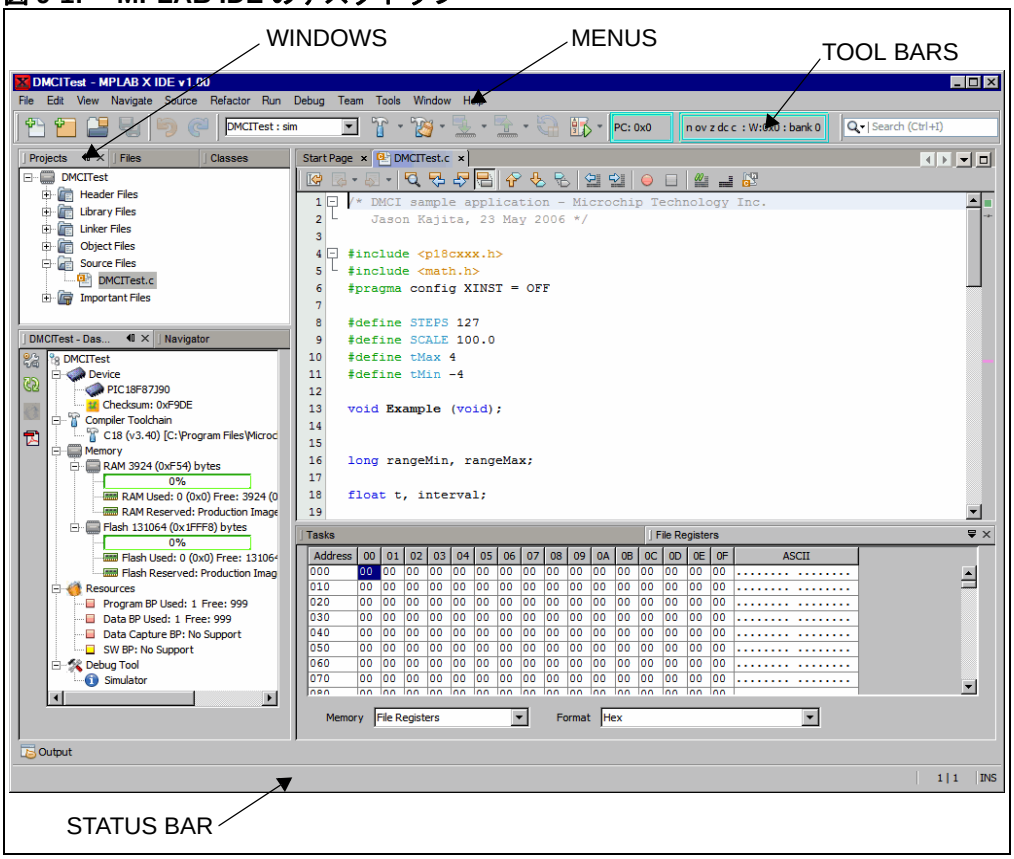
9.1 はじめに

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ます（ 図 9-1)。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**。

- メニュー
- ツールバー
- ステータスバー
- メニュー項目とボタンの灰色表示または非表示

図 9-1: MPLAB IDE のデスクトップ



9.2 メニュー

MPLAB 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 **Chapter 10.「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 キーも表示されます (例 :「New File」のショートカットキーは Control-N (CTRL+N))。ショートカットキーの詳細は、NetBeans ヘルプトピック ([IDE Basics]>[Keyboard Shortcuts]) 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) される場合があります。**13.4「メニュー項目および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、追加のコンテクスト メニューを利用できます。これらのメニューの詳細は、**10.3.1「[Projects] ウィンドウ」**を参照してください。

メニューの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、組み込みシステムの開発では使わない項目も含まれています。

- [File] メニュー
- [Edit] メニュー
- [View] メニュー
- [Navigate] メニュー
- [Source] メニュー
- [Refactor] メニュー
- [Run] メニュー
- [Debug] メニュー
- [Team] メニュー
- [Tools] メニュー
- [Window] メニュー
- [Help] メニュー

9.2.1 [File] メニュー

[File] メニュー内のメニュー項目を下表に示します。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、NetBeans ヘルプトピック ([IDE Basics]>[Keyboard Shortcuts]>[Menu Shortcuts]) を参照してください。

表 9-1: [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新しいプロジェクトを作成します。
New File	[New File] ウィザードを使って新しいファイルを作成します。
Open Project	既存のプロジェクトを開きます。
Open Recent Project	プロジェクトを選択するために、最近開いた全てのプロジェクトの一覧を表示します。
Import	下記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
Open Team Project	チーム プロジェクトを開きます (チームサーバ プロジェクトの詳細は、NetBeans ヘルプトピック ([IDE Basics]>[Collaborative Development]) を参照)。
Close Project (Name)	現在の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。

表 9-1: [FILE] メニューの項目 (続き)

コマンド	動作
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] ダイアログを開きます。
Save	現在のファイルを保存します。
Save As	現在のファイルを、パスおよび / または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] オプションを選択した場合、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 (印刷を実行する前にプレビューを表示します)。
Print to HTML	現在のファイルを HTML フォーマットの新しいファイルに印刷します。
Exit	MPLAB® IDE を終了します。

9.2.2 [Edit] メニュー

[Edit] メニュー内のメニュー項目を下表に示します。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、NetBeans ヘルプトピック ([IDE Basics]>[Keyboard Shortcuts]>[Menu Shortcuts]) を参照してください。

表 9-2: [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	書式設定したクリップボードの内容を、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	一致したテキストから順方向にインスタンスを検索します。
Find Previous	一致したテキストから逆方向にインスタンスを検索します。
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。マクロを使うには、[Tools]>[Options] を選択し、**[Editor]** ボタンをクリックし、**[Macros]** タブを開きます。

9.2.3 [View] メニュー

[View] メニュー内のメニュー項目を下表に示します。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、NetBeans ヘルプトピック ([IDE Basics]>[Keyboard Shortcuts]>[Menu Shortcuts]) を参照してください。

表 9-3: [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。
Code Folds>Collapse Fold	選択位置が折り畳み可能なテキスト セクション内である場合、そのセクション内の全ての行を 1 行に畳み込みます。
Code Folds>Expand Fold	ソースエディタ内で選択している行が折り畳まれた行である場合、そこに折り畳まれている全ての行を展開して表示します。
Code Folds>Collapse All	ソースエディタ内の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	ソースエディタ内の折り畳み可能なテキスト セクションを全て展開します。
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	[Task] ウィンドウのタブに、MPLAB® IDE のログファイルを開きます。
Toolbars>File, etc.	チェックマークを付けると、対応するツールバーを表示します。
Toolbars>Small Toolbar Icons	チェックマークを付けると、ツールバー上のアイコンが小さくなります。
Toolbars>Reset Toolbars	ツールバーを既定値の状態に戻します。
Toolbars>Customize	既存ツールバーの項目をカスタマイズして新しいツールバーを作成できます。
Show Editor Toolbar	チェックマークを付けると、[File] タブに [Editor] ツールバーを表示します。
Show Line Numbers	チェックマークを付けると、行番号を表示します。
Show Diff Sidebar	チェックマークを付けると、DIFF サイドバーを表示します。
Show Versioning Labels	チェックマークを付けると、バージョン管理ラベルを表示します。
Synchronize Editor with Views	チェックマークを付けると、開いている画面にエディタを同期させます。
Show Profiler Metrics	チェックマークを付けると、プロファイラ メトリクスを表示します。
Full Screen	ウィンドウを全画面表示にします。

9.2.4 [Navigate] メニュー

[Navigate] メニュー内のメニュー項目を下表に示します。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、NetBeans ヘルプトピック ([IDE Basics]>[Keyboard Shortcuts]>[Menu Shortcuts]) を参照してください。

表 9-4: [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 Document	現在開いているドキュメントの直前に開いていたドキュメントを開きます。
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 インプリメンテーション位置へ移動します。
Last Edit Location	直前に編集した位置までエディタをスクロールします。
Back	逆方向に移動します。

表 9-4: [NAVIGATE] メニューの項目 (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	コードの 1 行にブックマークを設定します。
Next Bookmark	次のブックマークに移動します (順方向に循環)。
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 Error	ソースエディタを次のビルドエラーを含む行までスクロールします。
Previous Error	ソースエディタを 1 つ前のビルドエラーを含む行までスクロールします。
Select in Projects	[Projects] ウィンドウを開いて、その中で現在のドキュメントを選択します。
Select in Files	[Opens Files] ウィンドウを開いて、その中で現在のドキュメントを選択します。
Select in Classes	[Opens Classes] ウィンドウを開いて、その中で現在のドキュメントを選択します。
Select in Favorites	[Opens Favorites] ウィンドウを開いて、その中で現在のドキュメントを選択します。

9.2.5 [Source] メニュー

[Source] メニュー内のメニュー項目を下表に示します。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、NetBeans ヘルプトピック ([IDE Basics]>[Keyboard Shortcuts]>[Menu Shortcuts]) を参照してください。

表 9-5: [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	行の末尾のスペースを削除します。
Shift Left	選択した 1 行または複数行をタブ 1 つ左へシフトします。
Shift Right	選択した 1 行または複数行をタブ 1 つ右へシフトします。
Move Up	選択した 1 行または複数行を 1 行上へ移動します。
Move Down	選択した 1 行または複数行を 1 行下へ移動します。
Duplicate Up	選択した 1 行または複数行を 1 行上に複製します。
Duplicate Down	選択した 1 行または複数行を 1 行下に複製します。
Toggle Comment	選択した 1 行または複数行のコメント化を ON/OFF 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。
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。
Scan for external changes	ファイルをスキャンして、MPLAB® IDE の外部で加えられた変更を見つけます。

9.2.6 [Refactor] メニュー

[Refactor] メニュー内のメニュー項目を下表に示します。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 **6.4「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、NetBeans ヘルプトピック ([IDE Basics]>[Keyboard Shortcuts]>[Menu Shortcuts]) を参照してください。

表 9-6: [REFACTOR] メニューの項目

コマンド	動作
Rename	変数または関数の名前を任意の新しい名前に変更し、そのエレメントを新しい名前で参照するために、プロジェクト内の全てのソースコードを更新します。
Move(1)	1 つのクラスを別のパッケージまたは別のクラスに移動し、移動したクラスを参照するために、プロジェクト内の全てのコードを更新します。
Copy(1)	1 つの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(1)	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、自動的にそのエレメントを削除します。
Undo	リファクタリングを取り消します。
Redo	リファクタリングをやり直します。
(1) Java および C++ 動作 (現在未サポート)	

9.2.7 [Run] メニュー

[Run] メニュー内のメニュー項目を下表に示します。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、NetBeans ヘルプトピック ([IDE Basics]>[Keyboard Shortcuts]>[Menu Shortcuts]) を参照してください。

表 9-7: [RUN] メニューの項目

コマンド	動作
Run Project	メインプロジェクトまたは選択したプロジェクトを実行します。
Test Project	プロジェクトの JUnit テストを開始します (Java 関連)。
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 (Only embedded available)。
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」を選択する必要があります。
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選択します。
Run File	現在選択中のファイルを実行します。
Test File	現在のファイルの JUnit テストを開始します (Java 関連)。
Check File	ファイルが規格に従っているかどうかチェックします (XML 関連)。
Validate File	ファイルが規格に従っているかどうか検証します (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。

9.2.8 [Debug] メニュー

[Debug] メニュー内のメニュー項目を下表に示します。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、NetBeans ヘルプトピック ([IDE Basics]>[Keyboard Shortcuts]>[Menu Shortcuts]) を参照してください。

表 9-8: [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。
Debug File	現在選択しているファイルでデバッグ セッションを始めます。
Debug Test File	ファイルの JUnit によるデバッグテストを始めます (Java 関連)。
Discrete Debugger Operation	ビルド、ターゲットのプログラミング、デバッガの起動を別々に実行します。 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	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。
Animate	プログラムを自動的に 1 ステップずつ実行します。
Step Over	プログラムの 1 ソース行を実行します。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 1 ソース行を実行します。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 Instruction	プログラムの 1 ソース行を実行します。呼び出した関数内でステップ実行中である場合、関数の残りを全て実行した後に、呼び出し元のルーチンに戻って停止します。
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。
Reset	プロセッサをリセットします。
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。
Focus Cursor at PC	カーソルを現在の PC アドレス位置へ移動します。
Stack>Make Callee Current	現在の呼び出しで呼び出される側のメソッドを make します。この機能は、[Call Stack] ウィンドウで 1 つの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	現在の呼び出しで呼び出す側のメソッドを make します。この機能は、[Call Stack] ウィンドウで 1 つの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	呼び出しを top of the 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	指定した変数をウォッチに追加します。
New Run Time Watch	指定した変数を、プログラム実行中に値の変化を観察可能なウォッチ (ランタイム 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通信を切断します。[Run] または [Debug Run] を選択すると再接続します。
Run Debugger/Programmer Self Test	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。

9.2.9 [Team] メニュー

[Team] メニュー内のメニュー項目を下表に示します。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、NetBeans ヘルプトピック ([IDE Basics]>[Keyboard Shortcuts]>[Menu Shortcuts]) を参照してください。

表 9-9: [TEAM] メニューの項目

コマンド	動作
Team Server	チーム プロジェクトを作成します。チームプロジェクトの詳細は、NetBeans ヘルプ ([IDE Basics]>[Collaborative Development]) を参照してください。
CVS, Mercurial, Subversion	バージョン管理システムごとに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については、各製品の文書を参照してください。
Local History	ファイルのローカル履歴を表示します。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 Issues	バージョン管理システム内の問題を見つけます。
Report an Issue	バージョン管理システム内の問題を報告します。
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。

9.2.10 [Tools] メニュー

[Tools] メニュー内のメニュー項目を下表に示します。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、NetBeans ヘルプトピック ([IDE Basics]>[Keyboard Shortcuts]>[Menu Shortcuts]) を参照してください。

表 9-10: [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。
Apply Diff Patch	差分パッチファイルを選択し、コードに適用します。
Diff	IDE 内で選択した 2 つのファイルを比較します。
Add to Favorites	選択したファイルを [Favorites] ウィンドウに追加します。
Templates	[Templates 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、NetBeans ヘルプトピック ([IDE Basics]>[Plugins]>[About Managing Plugins]) を参照してください。
Options	[Options] ダイアログを開きます。

9.2.11 [Window] メニュー

[Window] メニュー内のメニュー項目を下表に示します。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、NetBeans ヘルプトピック ([IDE Basics]>[Keyboard Shortcuts]>[Menu Shortcuts]) を参照してください。

表 9-11: [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。
Files	[Files] ウィンドウを開きます。
Classes	[Classes] ウィンドウを開きます。
Favorites	[Favorites] ウィンドウを開きます。
Services	[Services] ウィンドウを開きます。
Team	[Team] ウィンドウを開きます。**5.13「コード開発とエラー追跡の連携」**を参照してください。
Tasks	[Task List] ウィンドウを開きます。

表 9-11: [WINDOW] メニューの項目 (続き)

コマンド	動作
Dashboard	[Dashboard] ウィンドウを開きます。**5.10「ダッシュボードの表示」**を参照してください。
Palette	[Palette] ウィンドウを開きます (Java 関連)。
Properties	[Properties] ウィンドウを開いて、現在選択しているファイルのプロパティ情報を表示します。
Chat	[Team Chat] ウィンドウを開きます。
Output>Output	[Output] ウィンドウを開くか、最前面に移動します。
Output>Search Results	[Search] ウィンドウを開くか、最前面に移動します。
Output>Find Usages Results	ウィンドウに [Find Usages] の結果を表示します。
Output>Test Results	C プロジェクトの [Test Results] ウィンドウを開きます。
Output>Versioning Output	ウィンドウにバージョン管理動作の結果を表示します。
Output>Refactoring Preview	リファクタリング結果の [Preview] ウィンドウを開きます。
Output>Disassembly Listing File (Project *Project*)	*Project* の逆アセンブリ リストファイルを開きます。
Output>Call Graph	[Call Graph] ウィンドウを開きます。
Navigating>Navigator	[Navigator] ウィンドウを開きます。
Navigating>Hierarchy	[Hierarchy] ウィンドウを開きます。
Debugging>Variables	[Local Variables] デバッガ ウィンドウを開きます。
Debugging>Watches	[Watches] デバッガ ウィンドウを開きます。
Debugging>Call Stack	[Call Stack] デバッガ ウィンドウを開きます。
Debugging>Breakpoints	[Breakpoints] ウィンドウを開きます。
Debugging>Sessions	[Sessions] ウィンドウを開きます。
Debugging>Threads	[Threads] 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。
Debugging>Trace	[Trace] ウィンドウを開きます。この機能を使うには、ツールとデバイスがトレース機能をサポートしている必要があります。
Debugging>PIC App IO	[Application In/Out] ウィンドウを開きます。この機能を使うには、ツールとデバイスがApp IOをサポートしている必要があります。
Debugging>Stop Watch	[Stop Watch] ウィンドウを開きます。
Versioning>CVS	CVS バージョン管理アイテムを選択します。
Versioning>Subversion	Subversion バージョン管理アイテムを選択します。
Versioning>Mercurial	Mercurial バージョン管理アイテムを選択します。
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。
Simulator>Stimulus	シミュレータ用の [Stimulus] ウィンドウを開きます。
Other>Exception Reporter	例外ブレークポイント用の [Exception Reporter] ウィンドウを開きます。
Other>CSS Preview	CSS (カスケーディング スタイルシート) 用の [Preview] ウィンドウを開きます。
Other>CSS Style Builder	CSS ルール用の [Style Builder] ウィンドウを開きます。
Other>Macro Expansion	[Macro Expansion] ウィンドウを開いてマクロ構造を表示します。
Editor	空白の [Editor] ウィンドウを開きます。
Processes	C プロセス実行用の [Processes] ウィンドウを開きます。
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。
Maximize Window	[Source Editor] または現在のウィンドウを最大化します。

表 9-11: [WINDOW] メニューの項目 (続き)

コマンド	動作
Undock Window	IDE からウィンドウを分離します。
Clone Document	アクティブなドキュメントを複製します。
Close All Documents	ソースエディタで開いている全てのドキュメントを閉じます。
Close Other Documents	現在アクティブなドキュメント以外の全てのドキュメントを閉じます。
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、現在開いているドキュメントのグループを保存して閉じる事ができます。
Reset Windows	ウィンドウを既定値状態にリセットします。

9.2.12 [Help] メニュー

[Help] メニュー内のメニュー項目を下表に示します。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、NetBeans ヘルプトピック ([IDE Basics]>[Keyboard Shortcuts]>[Menu Shortcuts]) を参照してください。

表 9-12: [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セットを表示します。
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 ショートカットの説明を表示します。
Check for Updates	MPLAB® X IDE プログラムの更新を検索します。
Start Page	[Start Page] タブを開きます。既に開いている場合は、このタブを他のタブの最前面に移動します。
About	[about MPLAB IDE] ウィンドウを開きます。

9.3 ツールバー

MPLAB IDE は、現在使用中の機能またはツールに応じて、異なるツールバーを表示します。これらのツールバー内のアイコンは、よく使われる操作へのショートカットを提供します。

ツールバーのボタンは状況に応じて無効化 (灰色表示) される場合があります。**13.4「メニュー項目およびボタンの灰色表示または非表示」**を参照してください。

利用できるツールバー

以下の基本的なツールバーを利用できます。

- [File] ツールバー
- [Clipboard] ツールバー
- [Status Flags] ツールバー
- [Undo/Redo] ツールバー
- [Run] ツールバー
- [Debug] ツールバー
- [Memory] ツールバー
- [Quick Search] ツールバー
- [Editor] ツールバー

ツールバーの特長

ツールバーは以下の特長を備えます。

- マウスポインタをアイコンの上に移動すると、そのアイコンの機能を表示します。
- クリック & ドラッグ操作により、ツールバーをツールバー領域内の別の場所に移動できます。
- 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行えます。

9.3.1 [File] ツールバー

[File] ツールバーは下記のアイコンを含みます。これらの機能は [File] メニューからも利用できます。

- 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。
- 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。

9.3.2 [Clipboard] ツールバー

[Clipboard] ツールバーは下記のアイコンを含みます。これらの機能は [Edit] メニューからも利用できます。

- 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。

9.3.3 [Status Flags] ツールバー

[Status Flags] ツールバーは下記のアイコンを含みます。

- PC – プログラム カウンタ (PC) の現在値

9.3.4 [Undo/Redo] ツールバー

[Undo/Redo] ツールバーは下記のアイコンを含みます。これらの機能は [Edit] メニューからも利用できます。

- 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。

9.3.5 [Run] ツールバー

[Run] ツールバーは下記のアイコンを含みます。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。

- Set Project Configuration – プロジェクトのコンフィグレーションを選択します。「default」を選択する必要があります。
- Build Project – 全てのプロジェクト ファイルをビルドします。
- Clean and Build Project – 以前のビルドで作成されたファイルを削除した後に、全てのプロジェクト ファイルをビルドします。
- Make and Program Device Project – 選択したプロジェクトをビルドし、ターゲットにプログラミングした後に、プログラムを実行します。
- Hold in Reset – 選択したプロジェクトをビルドし、ターゲットにプログラミングした後に、リセット状態に保持します。
- Read Device Memory – ターゲットデバイスのメモリの内容を MPLAB X IDE に読み込みます。
- 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した後に、デバッグ実行します。

9.3.6 [Debug] ツールバー

[Debug] ツールバーは下記のアイコンを含みます。これらの機能は [Debug] メニューからも利用できます。

- 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。
- Reset – 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。
- 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。
- Animate – プログラムを自動的に 1 ステップずつ実行します。
- Step Over – プログラムの 1 ソース行を実行します。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 Over Expression – 式をステップオーバーし、その後にデバッグを停止します。
- Step Into – プログラムの 1 ソース行を実行します。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 1 ソース行を実行します。呼び出した関数内でステップ実行中である場合、関数の残りを全て実行した後に、呼び出し元のルーチンに戻って停止します。
- 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。
- Apply Code Changes – コード内の全ての変更を実行中のプログラムに適用します。

9.3.7 [Memory] ツールバー

[Memory] ツールバーは、MPLAB IDE による PC メモリの現在の使用率を表示します。表示をクリックすると、ガベージコレクションを実行できます。

9.3.8 [Quick Search] ツールバー

[Quick Search] ツールバーは、検索用テキストボックスを表示します。虫眼鏡アイコンの横の下向き矢印をクリックすると、検索のタイプを選択できます。

9.3.9 [Editor] ツールバー

[Editor]ツールバーは下記のアイコンを含みます。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最上段に表示されます。

- Last Edited – 直前に編集した行へ移動します。
- Back – 逆方向に移動します。
- Forward – 順方向に移動します。
- Find Selection – 選択したテキストに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 – 検索用に選択したテキストを ON/OFF します。
- Previous Bookmark – ブックマークを逆方向に循環します。
- Next Bookmark – ブックマークを正方向に循環します。
- Toggle Bookmark – コードの 1 行にブックマークを設定します。
- Shift Left – 選択した 1 行または複数行をタブ 1 つ左へシフトします。
- Shift Right – 選択した 1 行または複数行をタブ 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。
- 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。

9.4 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。現在は、エディタ情報だけを表示します。

9.5 メニュー項目とボタンの灰色表示または非表示

下記のような条件では、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)

NOTE:

Chapter 10. ウィンドウとダイアログ

10.1 はじめに

MPLAB X IDE のデスクトップは、タブ形式の複数のウィンドウ枠に分割されます。これらの一部は、特定の機能を選択するまで表示されません。

例えば、MPLAB X IDE の初回起動時には、**[Start Page]** 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 (左上の枠内に [Project]、[Files]、[Services] ウィンドウ、左下の枠内に [Navigation] ウィンドウ、右下の枠内に [Output] ウィンドウ) が開きます。**[Start Page]** ウィンドウは、右上の枠内に移動します。

MPLAB IDE では、NetBeans の基本的ウィンドウと MPLAB IDE 専用のウィンドウを組み合わせて使います。

- NetBeans ウィンドウとウィンドウ メニュー
- MPLAB X IDE 専用ウィンドウとウィンドウ メニュー

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 の基本的ダイアログと MPLAB IDE 専用ダイアログを組み合わせて使います。

- NetBeans ダイアログ
- MPLAB X IDE 専用ダイアログ

10.2 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 ヘルプに記載されています。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。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。あるいは、NetBeans ヘルプトピック「Managing IDE Windows」を参照してください。

大部分のウィンドウは、**9.2.11「[Window] 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 (**[Window]** 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 [Fill]、[Goto]、[Edit Cells] 等の項目を選択できます。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。これらのメニュー項目の大部分は、デスクトップ メニューバー内のメニューからも利用できます (**9.2「メニュー」** 参照)。

ウィンドウのプロパティを設定するには、[Tools]>[Options] を選択し、**[Miscellaneous]** ボタンをクリックし、**[Appearance]** タブを開きます。

10.3 MPLAB X IDE 専用ウィンドウとウィンドウ メニュー

MPLAB X IDE は各種の NetBeans ウィンドウ (**10.2「NetBeans ウィンドウとウィンドウ メニュー」**参照) に加えて、組み込み開発専用にカスタマイズまたは作成したウィンドウとメニューも使います。

- [Projects] ウィンドウ
- [Project Properties] ウィンドウ
- [Dashboard] ウィンドウ
- [Memory] ウィンドウ
- [Options] ウィンドウ
- [Output] ウィンドウ

10.3.1 [Projects] ウィンドウ

[Projects] ウィンドウは NetBeans ウィンドウです。しかし、MPLAB X IDE プロジェクトに関連する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

10.3.1.1 [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。従って下記の全てのフォルダが表示されます。

- **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。[Project] ウィンドウ内でいずれかのファイルをダブルクリックすると、そのファイルがエディタ内に開きます。
- **Library Files** – 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 ファイルを参照し、それらを最終的なリンクステップでインクルードします。
- **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は、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します。独自のリンカスクリプトを作成する場合にのみ、このフォルダ内にファイルが 1 つだけ必要です。複数のリンカスクリプトが存在しても、最初の 1 つだけが効果を持ちます。これは、ツールがリンクステップで使うリンカスクリプトです。
- **Object Files** – このカテゴリは、プリコンパイル済みのオブジェクト ファイル用です (プロジェクトのソースファイルをアセンブルまたはコンパイルして作成したオブジェクト ファイル用ではありません)。このカテゴリ内のファイルは、ビルドの最終段階でリンクされるという点において、ライブラリ ファイルと機能的に明確な違いはありません。
- **Source Files** – このカテゴリは、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ファイルだけを含みます。
- **Important Files** – 他のどのカテゴリにも属さないファイルは、全てこのカテゴリに含めます。例えばシミュレータ ファイル (SBS、STC、SCL) をこのカテゴリに追加しておけば、それらをダブルクリックして SCL ジェネレータ、スティミュラス コントローラ、エディタを開く事ができます。あるいは、プロジェクトに関連するデータシート (PDF) を、[Project] 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 (これにはPDF リーダーをインストールしておく必要があります)。

10.3.1.2 [PROJECTS] ウィンドウ – [PROJECT] メニュー

[Projects] ウィンドウ内で、いずれかのプロジェクト名を右クリックすると、[Project] メニュー (図 10-1) が開きます。MPLAB X IDE 専用のメニュー項目を表 10-1 に示します。

図 10-1: [PROJECT] コンテクスト メニュー

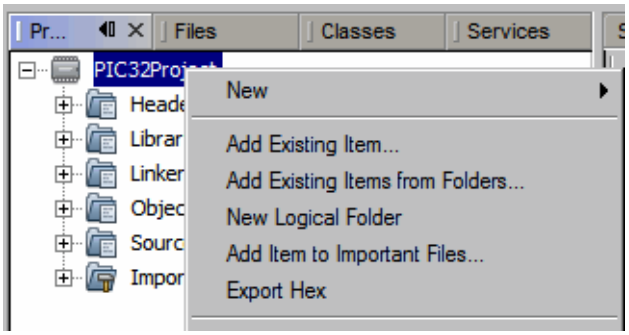


表 10-1: [PROJECT] コンテクスト メニューの項目

メニュー項目	内容
New	新しいアイテムを追加します。MPLAB® X IDE では、「embedded」ファイルタイプを選択できます。
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をエクスポートするのに対し、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。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。
Package	現在のプロジェクトを 1 つの ZIP ファイルにパッケージします。MPLAB X IDE はファイルを ZIP ファイルに圧縮できますが、ZIP ファイルを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 (絶対パスではない) 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」ディレクトリが含まれます。
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] ダイアログを開きます。
Hold in Reset	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。
Upload Target Memory	ターゲットデバイスからメモリの内容を読み出します。
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。**10.3.2「[Project Properties] ウィンドウ」**を参照してください。

10.3.1.3 [PROJECTS] ウィンドウ – [FILE] メニュー

[Projects] ウィンドウ内で、いずれかのファイル名を右クリックすると、[File] メニューが開きます。表 10-2 に、MPLAB X IDE 専用のメニュー項目を示します。

表 10-2: [FILE] コンテクスト メニューの項目

メニュー項目	内容
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」チェックボックスを有効にすると、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」チェックボックスを有効にすると、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。

10.3.2 [Project Properties] ウィンドウ

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。詳細は下記を参照してください。

- **3.5「プロジェクト プロパティの表示と変更」**
- **3.6「デバッガ、プログラマ、言語ツールのオプション設定」**
- **4.13「ビルド プロパティの設定」**

10.3.3 [Dashboard] ウィンドウ

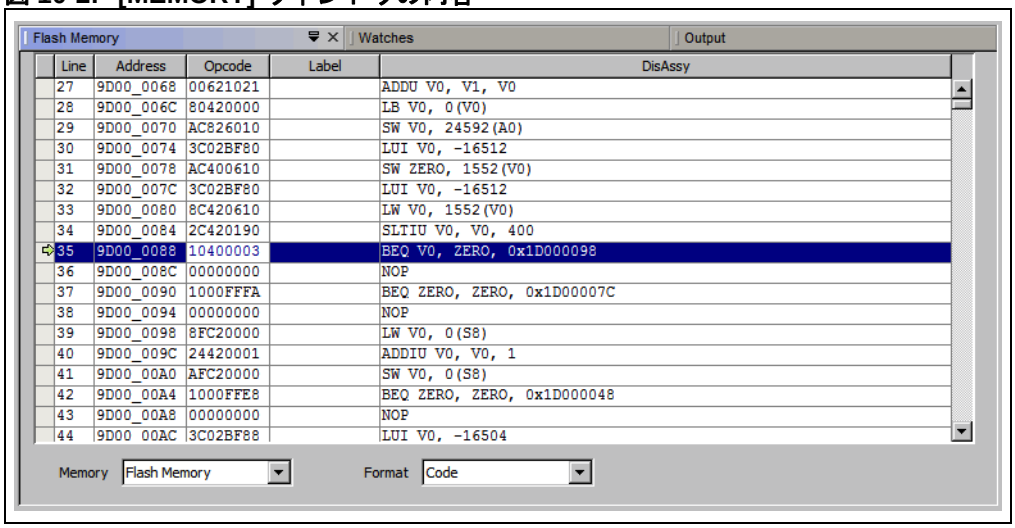
[Dashboard] ウィンドウはチェックサム、メモリ使用率、ブレークポイント リソース等の一般的なプロジェクト情報を表示します。詳細は **5.10「ダッシュボードの表示」**を参照してください。

10.3.4 [Memory] ウィンドウ

[Memory] ウィンドウは「PIC Memory Views」とも呼ばれ、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 **4.20「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 / 変更」**を参照してください。

図 10-2: [MEMORY] ウィンドウの内容



10.3.4.1　[MEMORY] ウィンドウのメニュー

[Memory] ウィンドウ内で右クリックすると、下表のオプションが表示されます。一部のデバイスでは使えないオプションもあります。

表 10-3:　[MEMORY] ウィンドウのメニュー項目

項目	内容
Virtual Address Physical Address	[Address] 列の下に表示するアドレスのタイプをチェックマークで選択します。
Hex Display Width	16 進数でディスプレイの幅を設定します。選択肢はデバイスによって異なります。 32 ビットデバイスの例 : One byte (例 : 00 01 02 ...0E 0F) Two bytes (例 : 00 02 04 ...0C 0E) Four bytes (例 : 00 04 08 0C)
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。
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。
Center Debug Location	現在の PC 位置をウィンドウの中央に表示します。
Symbolic Mode	逆アセンブラ HEX コードをシンボルで表示します。
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。
Find	[Find] ダイアログで指定したテキストを検索します。
Enable Multiline Rows	[Configuration Bits] ウィンドウ内の行の間隔を変更します。
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込みます。[Output to File Range] ダイアログで、出力するデータの範囲を指定します。
Import Table	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。[Import Table Range] ダイアログで、インポートするデータの範囲を指定します。
Export Table	[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。[Export Table Range] ダイアログで、エクスポートするデータの範囲を指定します。1 列にエクスポートするかどうかも指定します。
Print	このウィンドウの内容を印刷します。
Adjust Table Columns	表示する列を選択します。
Refresh	このウィンドウ内のデータを更新します。

[Window] タブを右クリックすると、その他のオプション (Close、Maximize/Minimize window、Dock/Undock window 等) を選択できます。

10.3.5 [Options] ウィンドウ

[Options] ウィンドウ ([Tools]>[Options]) は NetBeans ウィンドウです。ただし、**[Embedded]** ボタンを追加する事により、MPLAB X IDE プロジェクト用にカスタマイズしています。このボタンをクリックすると、下記のタブとオプションを利用できます。

- [Build Tools] タブ
- [Project Options] タブ
- [Generic Settings] タブ
- [Other] タブ

10.3.5.1 [BUILD TOOLS] タブ

Mac PCの場合、このタブ上の情報にアクセスする方法が異なります(メイン メニュー バーから [mplab_ide]>[/preferences] を選択してビルドツールにアクセス)。**3.7「言語ツールのパスの指定」**を参照してください。

インストールしていない言語ツールは「Tool Collection」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]** をクリックしてください。それでも表示されない場合は、**[Add]** をクリックして、そのツールをリストに追加します。

下記の言語ツールは MPLAB X IDE に付属しています。

- MPASM toolsuite – MPASM アセンブラ、MPLINK リンカ、各種ユーティリティ
- ASM30 toolsuite – 16 ビット アセンブラ、リンカ、各種ユーティリティ
- C32 toolsuite (free version) – 32 ビット コンパイラ、アセンブラ、リンカ、各種ユーティリティ

その他のツールは、マイクロチップ社ウェブサイト (www.microchip.com) またはサードパーティーから入手できます。

表 10-4: [BUILD TOOLS] タブの項目

項目	内容
Tool Collection	PC に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 compilers]** を使う方法もあります。
Duplicate	既存の言語ツールのリストアイテムを複製して別の名前を付け、必要に応じて設定を変更します。
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	いずれかのツールをクリックしてから **[Default]** を選択すると、そのツールは、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 compilers	PC に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。

10.3.5.2 [PROJECT OPTIONS] タブ

プロジェクトに関連するオプションを設定します。

表 10-5: [PROJECT OPTIONS] タブの項目

項目	内容
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 Options	プロジェクト内のファイルの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 (完全なパス) 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。このプロパティは選択したままにしておく事を推奨します。これを選択しない場合、変更したファイルを先にディスクに保存しておかない限り、それらの変更は make に反映されません。
Reuse Output Tabs from Finished Processes	これを選択した場合、make 出力は、前回の処理の出力を表示している [Output] ウィンドウのタブを上書きします (前回の結果は削除されます)。これを選択しない場合、メイク処理を実行するたびに、新しいタブを追加します。
Enable dependency checking in generated makefiles (requires support from toolchain)	make state命令文をmakefileに追加します。依存性チェックのサポートについては、使用する言語ツールの文書を参照してください。
Show binary files in Project view	これを選択した場合、[Projects] ウィンドウのディレクトリ ツリーにバイナリ オブジェクトを含む全てのファイルが表示されます。このオプションは、既存のソースを使って作成したプロジェクトで重要となります (そのようなプロジェクトでは、ソースとバイナリが同じ場所に保存される可能性があるため)。このオプションの選択を解除すると、C ソースファイルとヘッダファイルだけを見やすく表示できます。
Show profiler indicators during run (new projects only)	これを選択した場合、新規作成したプロジェクトの実行時に、プロファイラ ツール (CPU 使用率、メモリ使用率等) を実行します。表示されるツールは、[Tools]>[Profiler Tools] で選択した「Profile Configuration」によって決まります。 プロファイラに関する詳細は、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Working with C/C++/Fortran Projects]>[Profiling C/C++/Fortran Projects]) を参照してください。

10.3.5.3 [GENERIC SETTINGS] タブ

ログファイル等のプロジェクト機能を設定します。

表 10-6: [GENERIC SETTINGS] タブの項目

項目	内容
Logging Level	メッセージログのレベルを設定します。 **OFF:** ログしない **SEVERE:** 重大なメッセージのみログする **WARNING:** 警告メッセージのみログする **INFO:** 情報メッセージのみログする **CONFIG:** コンフィグレーション情報のみログする **FINE:** モジュール間の一部の通信をログする **FINER:** モジュール間の通信をより詳細にログする **FINEST:** モジュール間の全ての通信をログする
Log File	ログファイルのパス
Projects Folder	MPLAB® X IDE プロジェクトを保存するフォルダのパス
Maintain active connection to hardware tool	これを選択した場合、実行時以外もハードウェア ツールを常時接続したままにします (MPLAB® IDE v8 と同じ挙動)。
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。
Reset @	リセット時の動作を選択します。 **Main:** リセット時に main で停止する **Reset Vector:** リセット時にリセットベクタ位置で停止する
Debug start-up	デバッグ開始時の動作を選択します。 **Run**: 即座に実行を開始する **Main:** main で停止する **Reset Vector:** リセットベクタ位置で停止する

10.3.5.4 [OTHER] タブ

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も設定します。

> **Note:** 現在、MPLAB X IDE は C++ または Fortran によるプログラミングをサポートしていません。

10.3.6 [Output] ウィンドウ

[Task] ウィンドウ枠は、各種の NetBeans ウィンドウと MPLAB X IDE 専用ウィンドウを表示します。[Output] ウィンドウ内の各種タブには、MPLAB X IDE 出力情報が表示されます。

表 10-7: [OUTPUT] ウィンドウのタブ

項目	内容
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	ビルドとプログラミング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。

10.4 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。ダイアログ上の [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> キーを押してください。これでは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]** タブをクリックして、そのダイアログに関する情報を検索してください。

大部分のダイアログは、**9.2「メニュー」**に記載したメニューから開く事ができます。

10.5 MPLAB X IDE 専用ダイアログ

MPLAB X IDE は各種の NetBeans ダイアログに加えて、組み込み開発用に一部のダイアログをカスタマイズするか、専用に作成しています。

- [New Project] ウィザード – **3.3「新規プロジェクトの作成」**参照
- プロジェクト プロパティ – **3.5「プロジェクト プロパティの表示と変更」**参照
- ファイル プロパティ – **4.12「ファイル プロパティの設定」**参照
- ハードウェア ツールのセットアップ – **3.6「デバッガ、プログラマ、言語ツールのオプション設定」**参照
- 言語ツールのセットアップ – **3.6「デバッガ、プログラマ、言語ツールのオプション設定」**および **3.7「言語ツールのパスの指定」**参照
- MPLAB IDE v8 プロジェクトから MPLAB X IDE プロジェクトへの変換 – **5.2「旧バージョンの MPLAB プロジェクトのインポート」**参照

NOTE:

Chapter 11. プロジェクト ファイルおよびフォルダ

本章では、MPLAB X IDE 内のプロジェクト ファイルとプロジェクト フォルダを表示する下記のウィンドウについて説明します。

- [Projects] ウィンドウの表示
- [Files] ウィンドウの表示

その他に、MPLAB X IDE のファイルおよびフォルダに関する下記の内容も記載しています。

- 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
- MPLAB X IDE の外部でのプロジェクトのビルド

11.1 [PROJECTS] ウィンドウの表示

[Projects] ウィンドウが表示するプロジェクト フォルダは論理 (仮想) フォルダです。このウィンドウに関する詳細は、NetBeans ヘルプトピック「Projects Window」を参照してください。

ユーザ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 11-1: [PROJECTS] ウィンドウ – 単純な C コード プロジェクト

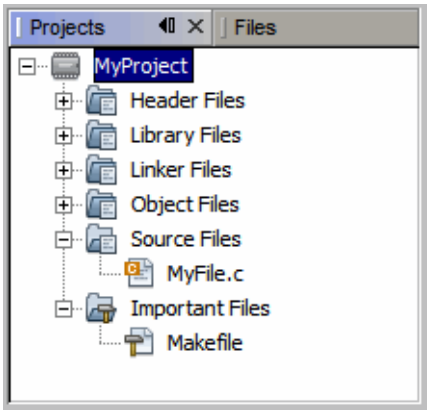


表 11-1: [PROJECTS] ウィンドウのフォルダの定義

仮想フォルダ	格納するファイル
Header Files	ヘッダファイル (.h、.inc)
Library Files	ライブラリ ファイル (.a、.lib)
Linker Files	リンカファイル (.ld、.gld、.lkr)
Object Files	プリコンパイル済みオブジェクト ファイル (.o)
Source Files	ソースファイル (.c、.asm)
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。

11.2 [FILES] ウィンドウの表示

[Files] フォルダに表示されるプロジェクト フォルダは、実際に PC 上に存在するフォルダです。C コード プロジェクトでこのウィンドウを使う場合の詳細は、NetBeans ヘルプトピック「Files Window」を参照してください。

図 11-2: [FILES] ウィンドウ – プロジェクト フォルダ

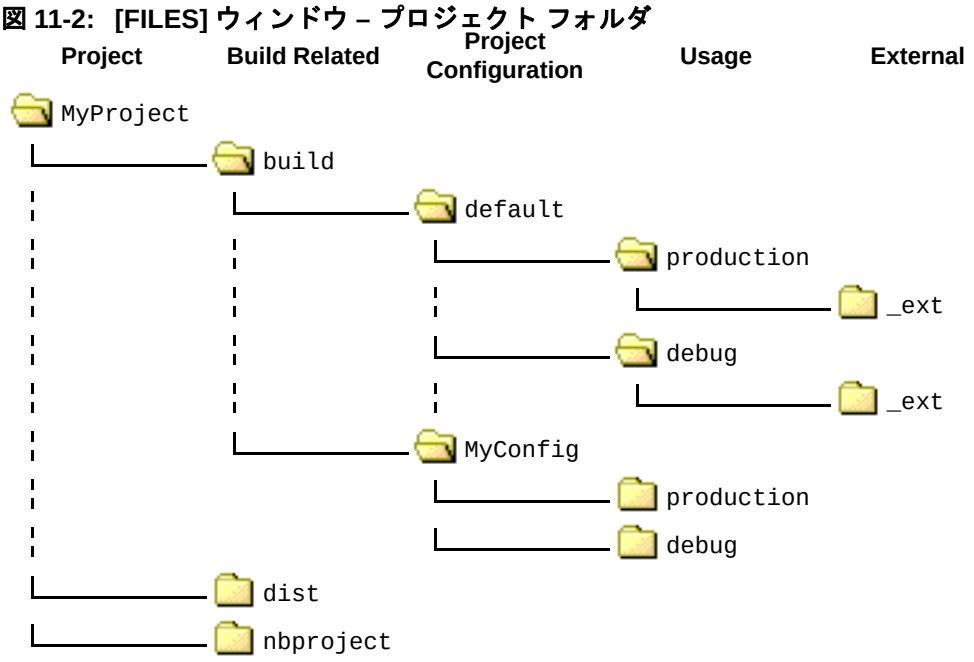


表 11-2: [FILES] ウィンドウのフォルダの定義

フォルダ	内容
MyProject	下記のファイルを格納するプロジェクト フォルダです。 • Makefile – プロジェクトのマスタ makefile • C コードまたはアセンブリ アプリケーション ファイル
build*	中間生成ファイルを格納するフォルダです。プロジェクト コンフィグレーション、使用方法、格納場所に応じたサブフォルダにファイルを格納します。このフォルダは下記のファイルを格納します。 • 実行ファイル (.o) • デバッグ実行ファイル (.o.d) • HI-TECH® の中間生成ファイル (.p1)
dist*	出力ファイルを格納するフォルダです。プロジェクト コンフィグレーション、使用方法、格納場所に応じたサブフォルダにファイルを格納します。このフォルダは下記のファイルを格納します。 • 実行ファイル (.hex) • ELF または COFF 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 ファイルを格納するフォルダです。このフォルダは下記のファイルを格納します。 • 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	プロジェクト コンフィグレーションを格納するフォルダです。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」フォルダに格納されます。
production, debug	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	外部プロジェクトのファイルを格納するフォルダです。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。
* これらのフォルダをソース管理に含めないでください。これらのフォルダはビルドによって作成されます。	

11.3 MPLAB IDE V8 プロジェクトのインポート – 相対パス

MPLAB IDE v8 プロジェクトは下記のパスに格納されています。

```
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、下記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。

この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。そのかわりに、v8 フォルダ内のファイル格納場所を参照します。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 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**5.2「旧バージョンのMPLABプロジェクトのインポート」**を参照してください。

11.4 プロジェクトの移動

プロジェクトを移動する方法については、NetBeans ヘルプトピック ([C/C++/Fortran Development]>[Working with C/C++/Fortran Projects]>[Moving, Copying, or Renaming a C/C++/Fortran Project]) を参照してください。

外部の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、ツールを MPLAB X IDE に限定しません。

11.5 MPLAB X IDE の外部でのプロジェクトのビルド

MPLAB X IDE は、プロジェクトをビルドするために makefile を使います。ターミナル (コマンドプロンプト) を開き、プロジェクト フォルダを指定して、make ユーティリティを実行する事により、IDE の外部でプロジェクトをビルドできます。これを行う前に、オペレーティング システムに make ユーティリティへのパスを指定しておく必要があります。この方法は、PC のオペレーティング システムによって下記のように異なります。

11.5.1 Windows の場合

MPLAB X IDE をインストールする時に、下記のディレクトリが作成されます。

```
C:\Program Files\Microchip\MPLABX\gnuBins\GnuWin32\bin
```

ここには、プロジェクトのビルドに必要な make ユーティリティ等が格納されます。

コマンドプロンプトから下記を入力して、上記のパスを PC の %PATH% 環境変数に事前に設定しておく事が必要です。

```
c:\>set PATH=C:\Program Files\Microchip\MPLABX\gnuBins\GnuWin32\bin;
%PATH%
```

64 ビットシステムの場合は下記を入力します。

```
c:\>set PATH=C:\Program Files (x86)\Microchip\MPLABX\gnuBins\
GnuWin32\bin;%PATH%
```

上記のパスは、本書のページ幅に収まり切らないために改行していますが、実際に入力する際は、改行せずに 1 行で入力してください。

11.5.2 MacOSX

Mac out-of-box は、GNU make 実行ファイルを除く全ての必要なツールを格納します。MPLAB X IDE をインストールする時に、下記のディレクトリが作成されます。

```
/Applications/microchip/mplabx/mplab_ide.app/Contents/Resources/
mplab_ide/bin
```

このディレクトリは make 実行ファイルを格納します。このディレクトリを下記のようにユーザパスに追加します。

```
$export PATH=/Applications/microchip/mplabx/mplab_ide.app/Contents/
Resources/mplab_ide/bin:$PATH
```

上記のディレクトリとパスは、本書のページ幅に収まり切らないために改行していますが、実際に入力する際は、改行せずに 1 行で入力してください。

11.5.3 Linux

標準の Linux ディストリビューションは、MPLAB X IDE が必要とするインストール済みのツール (GNU make を除く) を備えています。GNU make は、Linux ディストリビューション用のパッケージ マネージャを使ってインストールします。

下記のように、この make をテストします。

```
$ which make
/usr/bin/make
```

パスを設定した後に、MPLAB IDE X プロジェクト フォルダへのディレクトリを変更する必要があります。次に、下記を実行できます。

```
$ make -f nbproject/Makefile-$CONFIGURATION_NAME.mk SUBPROJECTS=
.build-conf
```

$CONFIGURATION_NAME は、ビルドするプロジェクト内のコンフィグレーションの名前です。MPLAB X IDE は常に既定値コンフィグレーションを作成します。他のコンフィグレーションを追加する事もできます。既定値コンフィグレーションの場合、コマンドは下記となります。

```
$ make -f nbproject/Makefile-default.mk SUBPROJECTS= .build-conf
```

コンフィグレーション (既定値) が 1 つしかない場合、下記をタイプ入力するだけで済みます。

```
$ make
```

プロジェクトをクリーンにする (ビルド内の中間生成ファイルと最終ファイルを削除する) には、下記をタイプ入力します。

```
$ make -f nbproject/$CONFIGURATION_NAME.mk SUBPROJECTS= .clean-conf
```

この場合の $CONFIGURATION_NAME は、クリーンにするプロジェクト内のコンフィグレーションの名前です。

```
$ make -f nbproject/Makefile-default.mk SUBPROJECTS= .clean-conf
```

上記により、既定値コンフィグレーションがクリーンになります。コンフィグレーションが 1 つしかない場合、下記をタイプ入力するだけで済みます。

```
$ make clean
```

Chapter 12. コンフィグレーション設定の要約

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について、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 ビットの設定に関するその他の情報については、使用する言語ツールの文書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、使用する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。

- MPASMWIN ツールチェーン
- HI-TECH® PICC™ ツールチェーン
- HI-TECH® PICC-18™ ツールチェーン
- C18 ツールチェーン
- ASM30 ツールチェーン
- C30 ツールチェーン
- C32 ツールチェーン

12.1 MPASMWIN 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。

12.1.1 __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」の使用を推奨します。以下に構文を示します。

__config *expr* ;コンフィグレーション ワードが 1 つだけの場合

または

__config *addr*, *expr* ;コンフィグレーション ワードが複数ある場合

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 OS の場合、このファイルは下記の既定値ディレクトリに格納されます。

```
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。

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
```

12.1.2 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: 1 つまたは複数のコンフィグレーション ビットを表現するマクロ

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 OS の場合、このファイルは下記の既定値ディレクトリに格納されます。

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。

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
```

12.2 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 OS の場合、このファイルは下記の既定値ディレクトリに格納されます。

C:\Program Files\HI-TECH Software\PICC*version*\include

version 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、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)。

PICC の例

```
#include <htc.h>

__CONFIG(WDTDIS & XT & UNPROTECT); // Program config. word 1
__CONFIG(FCMEN);                   // Program config. word 2
```

12.3 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

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 OS の場合、このファイルは下記の既定値ディレクトリに格納されます。

```
C:\Program Files\HI-TECH Software\PICC\version\include
```

version 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)。

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);
```

12.4 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: 1 つまたはコンマで区切った複数の「*setting-name* = *value-name*」マクロペア

マクロはデバイス ヘッダファイル (*.h) で定義されます。Windows OS の場合、このファイルは下記の既定値ディレクトリに格納されます。

```
C:\program files\microchip\mplabc18\version\.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。

右記も参照 : #pragma config

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
```

12.5 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

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 OS の場合、このファイルは下記の既定値ディレクトリに格納されます。

```
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 (PIC24F、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。

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
```

12.6 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。

12.6.1 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

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 OS の場合、このファイルは下記の既定値ディレクトリに格納されます。

```
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 – PIC24 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 )
```

12.6.2 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

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 OS の場合、このファイルは下記の既定値ディレクトリに格納されます。

```
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。

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);
```

12.7 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 1 つまたは複数の「*setting-name* = *value-name*」マクロペア

マクロはデバイス ヘッダファイル (*.h) で定義されます。Windows OS の場合、このファイルは下記の既定値ディレクトリに格納されます。

```
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。

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
```

NOTE:

サポート

はじめに

本章では、サポートについて説明します。

- 保証登録
- myMicrochip 変更通知サービス
- マイクロチップ社のウェブサイト
- マイクロチップ フォーラム
- カスタマ サポート
- マイクロチップ社の技術について

保証登録

ご購入になった開発ツール パッケージに保証登録カードが同梱されている場合、必要な事項を記入したカードを今すぐご送付ください。折り返し、製品のアップデート情報をお届けします。ソフトウェアのマイナー リリースは弊社ウェブサイトで提供しております。

myMICROCHIP 変更通知サービス

マイクロチップ社の変更通知サービスは、お客様にマイクロチップ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、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 からサービスに登録し、変更通知の配信をご希望になる製品カテゴリをお選びください。FAQ と登録方法の詳細も、上記のリンク先ページからご覧になれます。

変更通知の配信をご希望される製品カテゴリをお選びになる際に、「Development Systems」を選択すると、開発ツールのリストをご覧になれます。ツールの主なカテゴリは下記の通りです。

- **コンパイラ** – マイクロチップ社の C コンパイラ、アセンブラ、リンカ、その他の言語ツールの最新情報です。これには MPLAB C コンパイラ全製品、MPLAB アセンブラ全製品 (MPASM™ アセンブラを含む)、MPLAB リンカ全製品 (MPLINK™ オブジェクト リンカを含む)、MPLAB ライブラリアン全製品 (MPLIB™ オブジェクト ライブラリアンを含む) が含まれます。
- **エミュレータ** – マイクロチップ社製インサーキット エミュレータの最新情報です。これには MPLAB REAL ICE™ インサーキット エミュレータが含まれます。
- **インサーキット デバッガ** – マイクロチップ社のインサーキット デバッガの最新情報です。これには PICkit™ 2、PICkit 3、MPLAB ICD 3 インサーキット デバッガが含まれます。
- **MPLAB® IDE** – マイクロチップ社の MPLAB IDE(開発システムツール向け Windows® 統合開発環境) の最新情報です。これには MPLAB IDE、MPLAB IDE プロジェクト マネージャ、MPLAB エディタ、MPLAB SIM シミュレータ、一般的な編集およびデバッグ機能が含まれます。

- **プログラマ** – マイクロチップ社のプログラマの最新情報です。これには、デバイス (量産品) プログラマの MPLAB REAL ICE インサーキット エミュレータ、MPLAB ICD 3 インサーキット デバッガ、MPLAB PM3、および開発用プログラマの PICkit 2 および 3 が含まれます。
- **スタータ / デモボード** – これには MPLAB スタータキット ボード、PICDEM デモボード、その他の評価用ボードが含まれます。

マイクロチップ社のウェブサイト

マイクロチップ社は、自社が運営するウェブサイト (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。お好みのインターネット ブラウザを使って下記の内容をご覧になれます。

- **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、サンプルコード、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、最新ソフトウェア リリース、ソフトウェア アーカイブ
- **一般的技術サポート** – 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、マイクロチップ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 / 工場) の一覧

マイクロチップ フォーラム

マイクロチップ社のウエブフォーラム (http://www.microchip.com/forums) からもオンライン サポートをご利用いただけます。現在、下記のフォーラムを開設しています。

- 開発ツール
- 8 ビット PIC MCU
- 16 ビット PIC MCU
- 32 ビット PIC MCU

カスタマ サポート

マイクロチップ社製品をお使いのお客様は、下記のチャンネルからサポートをご利用いただけます。

- 代理店または販売担当者
- 各地の営業所
- フィールド アプリケーション エンジニア (FAE)
- 技術サポート

サポートについては代理店、販売担当者、フィールド アプリケーション エンジニア (FAE) までお問い合わせください。各地の営業所もご利用いただけます。本書の末尾には各国の営業所の一覧を記載しています。最新の営業所一覧は、弊社ウェブサイトでご確認ください。

技術サポートはウェブサイト(http://support.microchip.com)からご利用いただけます。

本書の内容に関して、誤りやご意見がございましたら、docerrors@microchip.com までメールでお寄せください。

マイクロチップ社の技術について

マイクロチップ社は、マイクロコントローラとアナログ半導体のトッププロバイダとして、世界各地のお客様の様々なアプリケーションに低リスクの製品開発、総システムコストの削減、開発期間の短縮を可能にする製品を提供しています。マイクロチップ社は本社をアリゾナ州チャンドラーに構え、優れた技術サポートを確かな納期と品質でお届けします。

電話 : (480) 792-7200

Fax: (480) 792-7277

myMicrochip: http://www.microchip.com/pcn

ウェブサイト : http://www.microchip.com

フォーラム : http://www.microchip.com/forums

サポート : http://support.microchip.com

NOTE:

用語集

A

絶対セクション (Absolute Section)
リンカによって変更されない固定 (絶対) アドレスを持つセクション。

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) の設定にかか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も提供する。

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 進数の数字の総称 (0,1, …, 9)。

AND 条件ブレークポイント (ANDed Breakpoints)
プログラム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® マイクロコントローラ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 / アーカイバ (Archive/Archiver)
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、制御文字等が含まれる。

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。

属性 (Attribute)
C 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用する。

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。

B

2 進数 (Binary)
0 と 1 の数字を使用する、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。

ブレークポイント (Breakpoint)
ハードウェア ブレークポイント : 実行するとファームウェアの実行が停止するイベント。
ソフトウェア ブレークポイント : ファームウェアの実行が停止するアドレス。通常、特別な Break 命令によって実行が停止される。

ビルド (Build)
全てのソースファイルをコンパイルおよびリンクしてアプリケーションを作成する事。

C

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 マイクロコントローラの内蔵 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 ファイルには、マシンコードの他、デバッグ等に関する情報が含まれる。

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 Bits)

PIC マイクロコントローラ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 Directives)

アセンブリ言語コード内で使用するディレクティブで、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」を参照。

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。残りのファイルにはシンボルへの参照が含まれる。

D

データ ディレクティブ (Data Directives)

アセンブラによって行われる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) を使って参照する手段としても使われる。

データメモリ (Data Memory)

マイクロチップ社の 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)

MPLAB 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 を参照。

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が出力される。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 Features)

後方互換性のためにサポートされているだけで現在は使用されておらず、いずれ廃止され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にプログラミングを行うためのツール。

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搭載したマイクロコントローラ。マイクロチップ社の dsPIC DSC デバイス等。

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 (Directives)

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ステートメント。

ダウンロード (Download)

ホストから別のデバイス (エミュレータ、プログラマ、ターゲットボード等) 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。

E

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はバイト単位で行われる。EEPROM の内容は電源を OFF にしても保持される。

ELF

Executable and Linking Format の略。この形式のオブジェクト ファイルにはマシンコードが含まれる。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 を参照。

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。エピローグは関数のユーザコードの後、関数リターンの直前に実行される。

EPROM

Erasable Programmable Read Only Memory の略。再書き込みが行えるタイプの ROM で、消去は紫外線照射によって行うものが主流。

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格納する。

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用する。

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 のデータを標準フォーマットで外部に出力する事。

式 (Expressions)

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)

関数や変数が、それ自身が定義され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)
オフチップの読み書き可能なメモリ。

F

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。メッセージも出力されない。

ファイル レジスタ (File Registers)
汎用レジスタ (GPR) と特殊機能レジスタ (SFR) で構成され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がプリフェッチされる。しかし、現在の命令によってプログラム 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。

G

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。

H

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われるメモリ領域。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われる。

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用する、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。

I

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をデバッグおよびプログラミングする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ボードにマウントされ、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われる。

ICSP

In-Circuit Serial Programming (インサーキット シリアル プログラミング) の略。マイクロチップ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 の IDE と同じ意味。

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 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される。

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 (Instructions)

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)
関数や変数が、それ自身が定義されたモジュールの外部からアクセスできない場合、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はじめ、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 として知られる。

割り込み (Interrupt)
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が再開される。

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 (Interrupt Service Request (IRQ))
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)
言語ツールの場合、割り込みを処理する関数。

MPLAB 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が実行される。

割り込みベクタ (Interrupt Vector)
割り込みサービスルーチン (ISR) または割り込みハンドラのアドレス。

L

左辺値 (L-value)
検査または変更が可能なオブジェクトを指し示す式。左辺値は代入演算子の左側で使用する。

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 / アーカイバ」を参照。

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 Files)
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 Directives)
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 ファイル (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やラベルには、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)

マイクロチップ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伝送 (Gbps) 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よって影響を受ける (信号線が長いと抵抗が増大し、電圧が低下する)。これに対し LVDS では、データストレージは実際の電圧レベルでなく正と負の電圧値によってのみ区別する。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.

M

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) で構成される。ある特定のプロセッサで使用でき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 Directives)

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。

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 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用する。

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。PIC18 C 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、多くの機能が統合されたチップ。

マイクロコントローラ モード (Microcontroller 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。マイクロコントローラ モードでは、内部実行のみが許可される。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利用できない。

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用されない。プログラムメモリ全体が外部にマッピングされる。

ニーモニック (Mnemonics)

マシンコードと 1 対 1 で対応したテキスト命令。オペコードとも呼ばれる。

MPASM™ アセンブラ

PIC マイクロコントローラ、KEELOQ® デバイス、マイクロチップ社のメモリデバイスに対応したマイクロチップ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マイクロチップ社の 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用する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は「MPLAB C Compiler for PIC18 MCU」を使用する。

MPLAB ICD

MPLAB IDE と組み合わせて使用するマイクロチップ社のインサーキット デバッガ。ICE/ICD を参照。

MPLAB IDE

マイクロチップ社の統合開発環境。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

マイクロチップ社のデバイス 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 コントローラの書き込みに対応。MPLAB IDE との併用も、単体での使用も可能。PROMATE II の後継製品。

MPLAB REAL ICE™ インサーキット エミュレータ

MPLAB IDE と組み合わせて使用するマイクロチップ社の次世代インサーキット エミュレータ。ICE/ICD を参照。

MPLAB SIM

MPLAB IDE と組み合わせて使用するマイクロチップ社のシミュレータで、PIC MCU と dsPIC DSC に対応する。

MPLIB™ オブジェクト ライブラリアン

MPLAB IDE と組み合わせて使用するマイクロチップ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 モジュールに使用するオブジェクト ライブラリアン。

MPLINK™ オブジェクト リンカ

マイクロチップ社の 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。マイクロチップ社の MPLIB ライブラリアンとの併用も可能。MPLAB IDE に統合して使用できるように設計されているが、MPLAB IDE 以外の環境でも使用できる。

MRU

Most Recently Used の略。最近使用したファイルおよびウィンドウの事。MPLAB IDE のメインメニューで選択できる。

N

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、[Watch] ウィンドウで使用する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含める事のできる階層の数。

ノード (Node)

MPLAB 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。拡張命令またはリテラル オフセットによるインデックス アドレス指定を利用しない。

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 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。

O

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によって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 (ライブラリ等) 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 Directives)

オブジェクト ファイル作成時にのみ使用するディレクティブ。

8 進数 (Octal)

0 ～ 7 の数字のみ使用する、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。ターゲットボード上のメモリを使用するか、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の選択を行う。

オペコード (Opcodes)

Operational Code の略。「ニーモニック」を参照。

演算子 (Operators)

定義可能な式を構成する際に使用される「+」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が評価される。

OTP

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。

P

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、イベントの条件が満たされる。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)

サポートされた Windows オペレーティング システムが動作する PC。

永続データ (Persistent Data)

クリアも初期化も行えないデータ。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用する。

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用する未実装バイト。dsPIC の hex ファイルに見られる。

PIC MCU (PIC MCUs)

マイクロチップ社の全てのマイクロコントローラ ファミリの総称。

PICkit 2/3

マイクロチップ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 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によって、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 ランダム化処理。

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われる。MPLAB C30 は属性を利用してこの情報を伝達する。

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)
デバイスのプログラムを高速に行え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も完全に準拠している。量産環境ではアプリケーション回路が組立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が特に重視される。

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 – デバイス内で命令が保存されるメモリ領域。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領域もプログラムメモリと呼ばれる。
16 ビット アセンブラ / コンパイラ – デバイス内で命令が保存されるメモリ領域。

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 (ソースコードやリンカ スクリプト ファイル等) 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される。

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。

PWM 信号 (PWM Signals)
パルス幅変調信号。一部の PIC MCU には周辺モジュールとして PWM が内蔵されている。

Q

修飾子 (Qualifier)
パスカウンタで使用する、または複合トリガにおける次の動作前のイベントとして使用するアドレスまたはアドレス範囲。

R

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。

ロー (raw) 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、通常のチップと全く同じ挙動をする。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、またはユーザによって実行が停止され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)
定義した関数や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 (Reentrant)
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 ASM30 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)
アドレスが固定されたメモリ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によって、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シンボルは全て新しいアドレスに更新される。

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 (Runtime Watch)

アプリケーション実行中に [Watch] ウィンドウで変数の値がリアルタイムに変化する事。ランタイム ウォッチの設定方法は、各ツールのマニュアルを参照。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。

S

シナリオ (Scenario)

MPLAB SIM シミュレータにおいて、スティミュラス制御を具体的に設定したもの。

セクション (Section)

特定のメモリ番地にあるアプリケーションの一部。

セクション属性 (Section Attribute)

セクションの特徴を表す情報 (access セクション等)。

シーケンス ブレークポイント (Sequenced Breakpoints)

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はボトムアップ方式で行われる。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する。

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ムする際に、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。エントリコード、パスワード、ID 番号等を書き込む目的で使用する。

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リングするソフトウェア プログラム。

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 のレジスタ ウィンドウ、ウォッチ変数、ステータス ディスプレイの表示が更新されるため、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、その場合は 1 命令ずつ実行されるのではなく、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がシングルステップで実行される。

スキュー (Skew)

命令実行に関する情報は、異なる複数のタイミングでプロセッサバスに現れる。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現れる。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現れる。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現れる。トレースバッファには、1 インスタンスでバス上に存在する情報がキャプチャされる。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、ブレークポイントからさらに 1 つ以上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される命令の数をスキッドと呼ぶ。

ソースコード (Source Code)
プログラマ (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によって実行される。

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 (Special Function Registers: 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用するデータメモリ (RAM) 領域。

SQTP
「Serialized Quick Turn Programming」を参照。

ハードウェア スタック (Stack, Hardware)
PICマイクロコントローラで関数呼び出しを行う時にリターン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リターンアドレス、関数パラメータ、ローカル変数を保存するのに使用するメモリ。高級言語でコードを開発する場合、このメモリは通常コンパイラによって管理される。

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マイクロチップ社のスタータキット。既製のアプリケーションの動作を確認した後で、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ムを行う。

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。頻繁に書き換える必要のないプログラムを書き込む。

ステータスバー (Status Bar)
MPLAB 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。

Step Into
Single Step と同じコマンド。Step Over とは異なり、Step Into では CALL 命令によって呼び出されるサブルーチンもステップ実行される。

Step Over
Step Over を実行すると、サブルーチン内はステップ実行せずにデバッグできる。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。

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が全て実行され、サブルーチンのリターン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されるデータ。通常、このデータはテキストファイルにアクションのリストとして記述される。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)
実行サイクルを測定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を格納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 (const 等)。

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 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。

絶対シンボル (Symbol, Absolute)
アセンブリの .equ ディレクティブによる定義等、即値を表す。

システム ウィンドウ コントロール (System Window Control)
ウィンドウや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。

T

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プログラマブルなデバイス。

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 ボードで使用されているマイクロコントローラ デバイス。

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 ファイルに保存される。

ツールバー (Tool Bar)
MPLAB 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)
プログラム実行のログ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 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)
エミュレータに内蔵された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ムする必要がある。

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、トレースおよびブレークポイント設定から独立したエミュレータ出力信号の事。トリガ出力ポイントはいくつでも設定できる。

トライグラフ (Trigraphs)
「??」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。

U

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;
```

として定義した変数は非初期化データ セクションに格納される。

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 PC 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 でサポートされるデータ転送レートは 12 Mbps。USB 2.0 (Hi-Speed USB) は最大 480 Mbps のデータ レートをサポートしている。

V

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。

W

警告 (Warning)
MPLAB IDE – 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 - 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。MPLAB C30 の警告メッセージではソースファイル名と行番号が報告されるが、エラーメッセージと区別するために「warning:」の文字列も付加される。

ウォッチ変数 (Watch Variable)
デバッグ セッション中に [Watch] ウィンドウで観察できる変数。

[Watch] ウィンドウ (Watch Window)
ウォッチ変数の一覧が表示され、ブレークポイントで毎回表示が更新されるウィンドウ。

ウォッチドッグ タイマ (WDT)

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。

NOTE:

索引

NOTE:

各国の営業所とサービス

北米

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:
www.microchip.com

アトランタ
Duluth, GA
Tel:678-957-9614
Fax:678-957-1455

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

クリーブランド
Independence, OH
Tel:216-447-0464
Fax:216-447-0643

ダラス
Addison, TX
Tel:972-818-7423
Fax:972-818-2924

デトロイト
Farmington Hills, MI
Tel:248-538-2250
Fax:248-538-2260

インディアナポリス
Noblesville, IN
Tel:317-773-8323
Fax:317-773-5453

ロサンゼルス
Mission Viejo, CA
Tel:949-462-9523
Fax:949-462-9608

サンタクララ
Santa Clara, CA
Tel:408-961-6444
Fax:408-961-6445

トロント
Mississauga, Ontario, Canada
Tel:905-673-0699
Fax:905-673-6509

アジア / 太平洋

アジア太平洋支社
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel:852-2401-1200
Fax:852-2401-3431

オーストラリア - シドニー
Tel:61-2-9868-6733
Fax:61-2-9868-6755

中国 - 北京
Tel:86-10-8569-7000
Fax:86-10-8528-2104

中国 - 成都
Tel:86-28-8665-5511
Fax:86-28-8665-7889

中国 - 重慶
Tel:86-23-8980-9588
Fax:86-23-8980-9500

中国 - 杭州
Tel:86-571-2819-3187
Fax:86-571-2819-3189

中国 - 香港 SAR
Tel:852-2401-1200
Fax:852-2401-3431

中国 - 南京
Tel:86-25-8473-2460
Fax:86-25-8473-2470

中国 - 青島
Tel:86-532-8502-7355
Fax:86-532-8502-7205

中国 - 上海
Tel:86-21-5407-5533
Fax:86-21-5407-5066

中国 - 瀋陽
Tel:86-24-2334-2829
Fax:86-24-2334-2393

中国 - 深圳
Tel:86-755-8203-2660
Fax:86-755-8203-1760

中国 - 武漢
Tel:86-27-5980-5300
Fax:86-27-5980-5118

中国 - 西安
Tel:86-29-8833-7252
Fax:86-29-8833-7256

中国 - 厦門
Tel:86-592-2388138
Fax:86-592-2388130

中国 - 珠海
Tel:86-756-3210040
Fax:86-756-3210049

アジア / 太平洋

インド - バンガロール
Tel:91-80-3090-4444
Fax:91-80-3090-4123

インド - ニューデリー
Tel:91-11-4160-8631
Fax:91-11-4160-8632

インド - プネ
Tel:91-20-2566-1512
Fax:91-20-2566-1513

日本 - 大阪
Tel:81-66-152-7160
Fax:81-66-152-9310

日本 - 横浜
Tel:81-45-471- 6166
Fax:81-45-471-6122

韓国 - 大邱
Tel:82-53-744-4301
Fax:82-53-744-4302

韓国 - ソウル
Tel:82-2-554-7200
Fax:82-2-558-5932 または
82-2-558-5934

マレーシア - クアラルンプール
Tel:60-3-6201-9857
Fax:60-3-6201-9859

マレーシア - ペナン
Tel:60-4-227-8870
Fax:60-4-227-4068

フィリピン - マニラ
Tel:63-2-634-9065
Fax:63-2-634-9069

シンガポール
Tel:65-6334-8870
Fax:65-6334-8850

台湾 - 新竹
Tel:886-3-5778-366
Fax:886-3-5770-955

台湾 - 高雄
Tel:886-7-536-4818
Fax:886-7-330-9305

台湾 - 台北
Tel:886-2-2500-6610
Fax:886-2-2508-0102

タイ - バンコク
Tel:66-2-694-1351
Fax:66-2-694-1350

ヨーロッパ

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

フランス - パリ
Tel:33-1-69-53-63-20
Fax:33-1-69-30-90-79

ドイツ - ミュンヘン
Tel:49-89-627-144-0
Fax:49-89-627-144-44

イタリア - ミラノ
Tel:39-0331-742611
Fax:39-0331-466781

オランダ - ドリューネン
Tel:31-416-690399
Fax:31-416-690340

スペイン - マドリッド
Tel:34-91-708-08-90
Fax:34-91-708-08-91

イギリス - ウォーキンガム
Tel:44-118-921-5869
Fax:44-118-921-5820

11/29/11