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

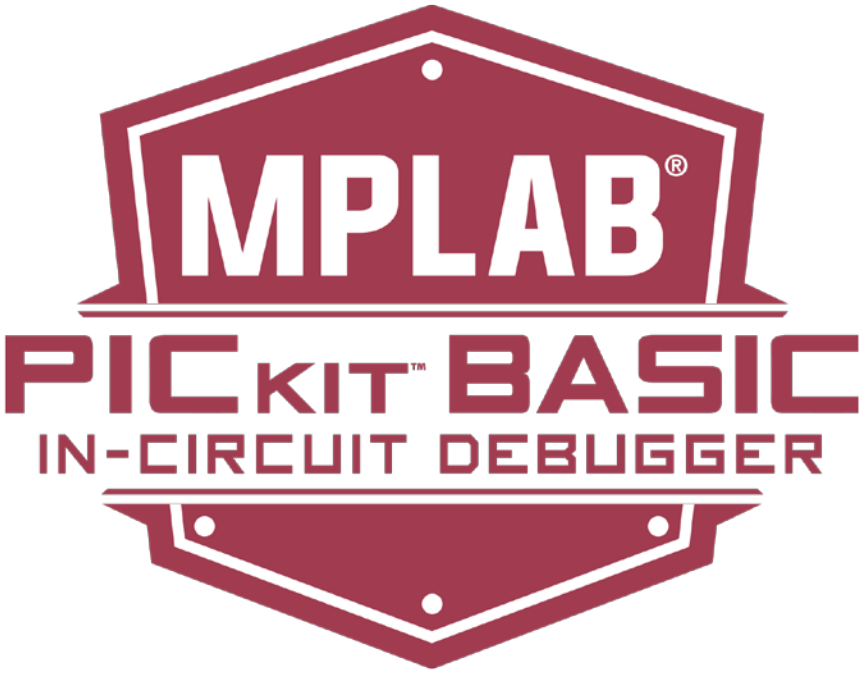
MPLAB® PICkit™ Basic インサーキット デバッガ ユーザガイド

開発ツールをご使用のお客様へ

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

文書は各ページのフッタに表記している「DS」番号で識別します。DS 番号のフォーマットは DS<文書番号><リビジョン>です。<文書番号>は 8 桁の番号、<リビジョン>はアルファベットの大文字です。

最新情報はツールのヘルプ(onlinedocs.microchip.com/)をご覧ください。



目次

1. はじめに

MPLAB® PICkit™ Basic インサーキット デバッガ(PG164110)は、高電圧プログラミングや先進デバッグ機能を必要としないプロジェクト向けの超低価格デバッグ ソリューションです。本デバッガは Microchip 社の多くの最新 MCU 製品をサポートしますが、一部のレガシー製品をサポートしません。基本的な機能セットを備えた本デバッガは、先進機能を必要としない開発を対象としています。
本デバッガは量産プログラミング向けではありません。

重要: UPDI を備えた少ピン数 AVR® デバイスにおいて、RSTPINCFG コンフィグレーション ビットによって UPDI ピンが GPIO または RESET として設定されている場合、MPLAB PICkit Basic は UPDI インターフェイスを再度有効にするための高電圧パルスを生成できません。これを行うには、MPLAB PICkit 5 等のツールが必要です。

1.1 本書の表記規則

本書には以下の表記規則を適用しています。

表 1-1. 本書の表記規則

内容	表記方法	例
参考資料	二重かぎカッコ: 『 』	『MPLAB® PICkit™ Basic インサーキット デバッガ ユーザガイド』
テキストの強調	**太字**	...は**唯一の**コンパイラです...
ウィンドウ名、ダイアログ名	角カッコ([ ])で囲んだ太字	**[Output]**ウィンドウ **[New Watch]**ダイアログ
ウィンドウまたはダイアログ内のフィールド名	かぎカッコ(「 」)で囲んだ太字	**「Optimizations」**オプション カテゴリを選択する
メニュー名または項目	かぎカッコ(「 」)で囲んだ太字	**「File」**メニューから**「Save」**を選択する
メニュー操作	右山カッコ(>)で区切った太字	**File** > **Save**
タブ	角カッコ([ ])で囲んだ太字	**[Power]**タブをクリックする
ソフトウェア ボタン	角カッコ([ ])で囲んだ太字	**[OK]**ボタンをクリックする
キーボードのキー	山カッコ(< >)で囲んだテキスト	<F1>キーを押す
ファイル名、ファイルパス	標準書体の `Courier New`	`C:/Users/User1/Projects`
本文中のソースコード	標準書体の `Courier New`	コードの先頭に`#define START` を忘れないでください

ソースコード	標準書体の `Courier New`	例:
		`#include <xc.h>`
		`main(void) {`
		`  while(1);`
		`}`

ユーザ入力データ	標準書体の `Courier New`	例: デバイス名の入力 `PIC18F47Q10`
キーワード	標準書体の `Courier New`	`static`、`auto`、`extern`
コマンドライン オプション	標準書体の `Courier New`	`-Opa+`、`-Opa-`
ビット値	標準書体の `Courier New`	`0`, `1`
定数	標準書体の `Courier New`	`0xFF`、`'A'`
変数の引数	斜体の *`Courier New`*	*`file`*`.o` (*`file`* は有効な任意のファイル名)
オプションの引数	角カッコ([ ])	`xc8 [`*`options`*`] `*`files`*

表 1-1. 本書の表記規則(続き)

内容	表記方法	例
引数のどれかを選択する場合(OR 選択)	中カッコとパイプ文字:{|}	`errorlevel {0|1}`
繰り返されるテキスト	省略記号: ...	*`var_name`* `[, ` *`var_name`*`...]`

ユーザが定義するコード	省略記号: ...	`void main (void)`
		`{ ...`
		`}`

Verilog 形式の数値 (N は総桁数、R は基数、n は各桁の値)	N'Rnnnn	4'b0010, 2'hF1
デバイス依存性(機能が一部のデバイスでサポートされない事)を示す、 詳細はデバイス データシートを参照	[DD]	アセンブラの特殊演算子等

1.2 推奨参考資料

本書では MPLAB PICkit Basic インサーキット デバッガの使い方を説明しています。以下の文書にも役に立つ情報が記載されています。参考資料として、Microchip 社が提供する以下の文書を推奨します。

Development Tools Design Advisory (DS-50001764)

最初にこの文書を読んでください。この文書には、MPLAB PICkit Basic インサーキット デバッガをお客様のターゲット回路で使用する際に考慮する必要がある動作上の問題に関する重要情報が記載されています。

MPLAB X IDE オンラインヘルプ

このオンラインヘルプは全ての Microchip 社ハードウェア ツールで使われます。これは MPLAB X IDE の広範なヘルプファイルです。この文書には組み込みシステムの概要、インストール要件、チュートリアル、プロジェクトの新規作成、ビルドプロパティの設定、コードのデバッグ、コンフィグレーションビットの設定、ブレークポイントの設定、デバイスへの書き込みに関する説明が記載されています。通常このヘルプファイルは MPLAB X IDE ウェブページからダウンロードできる PDF 版ユーザガイド(DS-50002027)よりも新しい情報を収めています。

MPLAB PICkit Basic のリリースノート

MPLAB PICkit Basic インサーキット デバッガに関する最新情報は、MPLAB X IDE 内のリリースノート(**Help** > **Release Notes**)に記載されています。リリースノートには、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。

MPLAB® PICkit™ Basic In-Circuit Debugger Quick Start Guide (DS-5000xxxx)

この文書には、MPLAB PICkit Basic インサーキット エミュレータ向けのハードウェアの接続方法とソフトウェアのインストール方法を記載しています。

2. 本デバッガについて

MPLAB® PICkit™ Basic インサーキット デバッガ(PG164110) を使うと、MPLAB® X IDE(統合開発環境)または MPLAB IPE(統合プログラミング環境) のパワフルなグラフィカル ユーザ インターフェイスを使って、低コストで迅速かつ容易にデバッグ/プログラミングが行えます。本デバッガがサポートするデバイスには以下が含まれます。

- PIC®および AVR®マイクロコントローラ(MCU)
- dsPIC®デジタル シグナル コントローラ(DCS)
- SAM (Arm® Cortex®ベース) マイクロコントローラ(MCU)およびマイクロプロセッサ(MPU)
- CEC (Arm Cortex ベース) MCU

重要: MPLAB PICkit Basic は高電圧プログラミングをサポートしないため、プログラミング用に高電圧を要求するデバイスをサポートしません。MPLAB PICkit Basic がサポートするデバイスについては、デバイスサポート リスト内の MPLAB PICkit Basic プログラマ(PKBasicP)列とデバッガ(PKBasicD)列を参照してください。

MPLAB PICkit Basic は、USB 2.0 ハイスピード インターフェイスを介してコンピュータに接続し、Microchip 社デバッグ 8 ピンシングル インライン(SIL)コネクタを介してターゲットに接続します。MPLAB PICkit Basic は、プロトタイプを迅速にデバッグするために必要な基礎レベルの機能を備えています。

本デバッガは強力な 32 ビット 300 MHz SAM E70 Arm Cortex-M7 ベースの MCU を備え、デバッグを迅速に反復できます。また、幅広いターゲット電圧レンジをサポートし、4 線式 JTAG およびシリアル ワイヤ デバッグ等をサポートします。2 線式 JTAG と In-Circuit Serial Programming™を使うデモボードおよびターゲット システムとの後方互換性も有します。

本デバッガシステムは特殊なデバッガチップではなくターゲット デバイスの内蔵エミュレーション回路を使うため、実際のデバイスと同じようにコードを実行します。ターゲット デバイスが実装している全機能にインタラクティブにアクセスでき、MPLAB X IDE インターフェイスによる設定と変更が可能です。

MPLAB PICkit Basic は以下のプラットフォームと互換です。

- Microsoft Windows® OS
- Linux® OS
- macOS®

サポートするバージョンについては、リリースノートを参照してください。

2.1 MPLAB PICkit Basic インサーキット デバッガの特長

MPLAB® PICkit™ Basic インサーキット デバッガは以下の特長を備えています。

機能:

- ハイスピード USB 2.0 (480 Mbits/s)ケーブルでコンピュータと接続
- USB ケーブルによる給電
  - 本デバッガからターゲットに給電する事はできません。ターゲットには別の電源が必要です。
- 8 ピン SIL プログラミング コネクタを備え、各種インターフェイスに対応
- MPLAB X IDE または MPLAB IPE を使ってデバイスをプログラミング

- SAM、CEC、PIC32 等の 32 ビット マイクロコントローラを含む多数の Microchip PIC、dsPIC、AVR、DSC デバイスに対応(詳細は MPLAB X IDE のデバイスサポート リスト参照)
- 4 線式 JTAG、シリアルワイヤ デバッグ、UPDI、PDI、SPI プログラミング、debugWIRE、TPI プログラミングをサポート
- 2 線式 JTAG と ICSP(インサーキット シリアル プログラミング)を使うデモボードおよびターゲットシステムとの後方互換性
- 複数のハードウェアおよびソフトウェア ブレークポイント、ストップウォッチ機能、ソースコードファイルのデバッグをサポート
- 実際のハードウェア上でのリアルタイム デバッグ
- 内部イベントに基づくブレークポイントの設定
- ターゲット MCU のフルスピードでのデバッグ
- ピンドライバの設定
- MPLAB X IDE の最新バージョン(http://www.microchip.com/mplabx/で無償提供)をインストールする事により最新デバイスのサポートと新機能を追加
- アクティブ/ステータス LED でデバッガのステータスを表示

性能/速度:

- ターゲット デバイス切り換え時にファームウェア ダウンロード遅延が生じない
- 300 MHz で動作する Arm® Cortex®-M7 コアを備えた 32 ビット MCU

安全性:

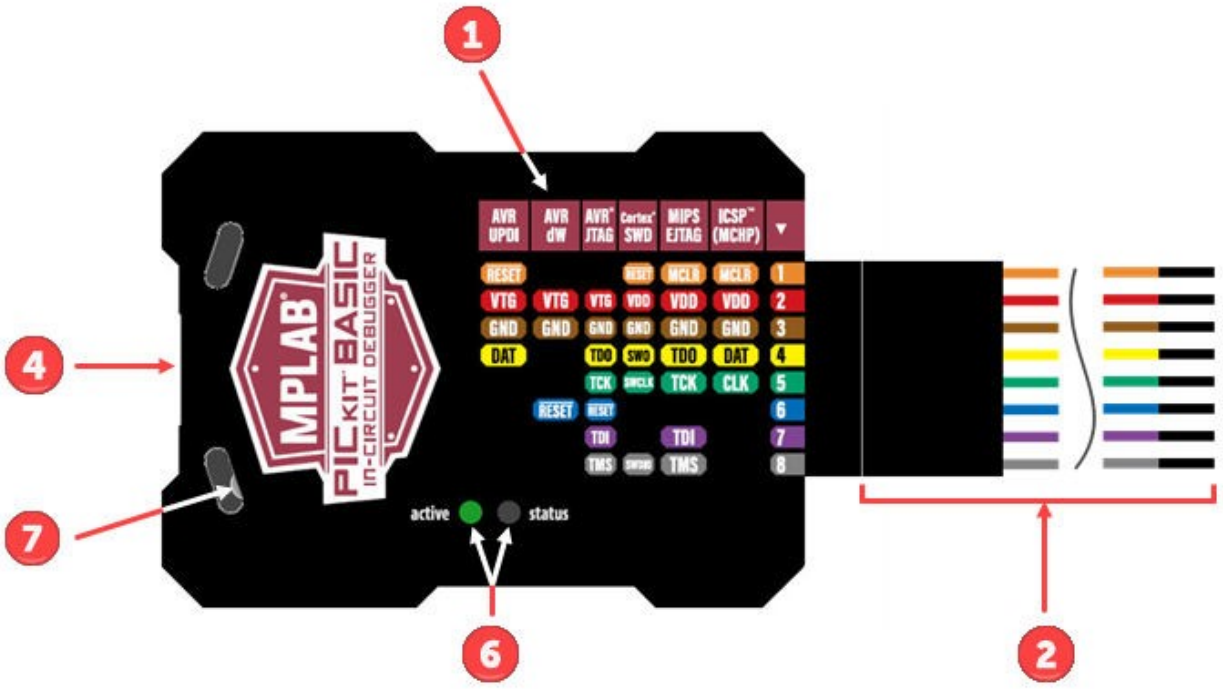
- RoHS、CE、China E に準拠
- 1.2～5.0V +/-10%のターゲット電源電圧をサポート

Note: PICkit Basic からターゲットに給電する事はできません。

2.2 MPLAB PICkit Basic インサーキット デバッガの構成要素

MPLAB PICkit Basic インサーキット デバッガシステムの構成要素を図 2-1 に示します。

図 2-1. MPLAB PICkit Basic インサーキット デバッガ



①: ケース前面の色分け信号ラベル( 列にピン番号を表示)

②: ケース底面の 8 ピン SIL コネクタと色分けワイヤ(色は上記信号ラベルに対応)

- 代わりに 8 ピン-10 ピン ARM SWD アダプタボードを接続する事も可能

④: ケース上面の USB Type-C®コネクタ

- コンピュータとのデータ/電源接続用に同梱のフル機能 USB Type-C®ケーブルを接続

⑥: ケース前面の 2 個のステータス LED

⑦: 緊急リカバリボタン(ケース前面の左側の穴から操作可能)

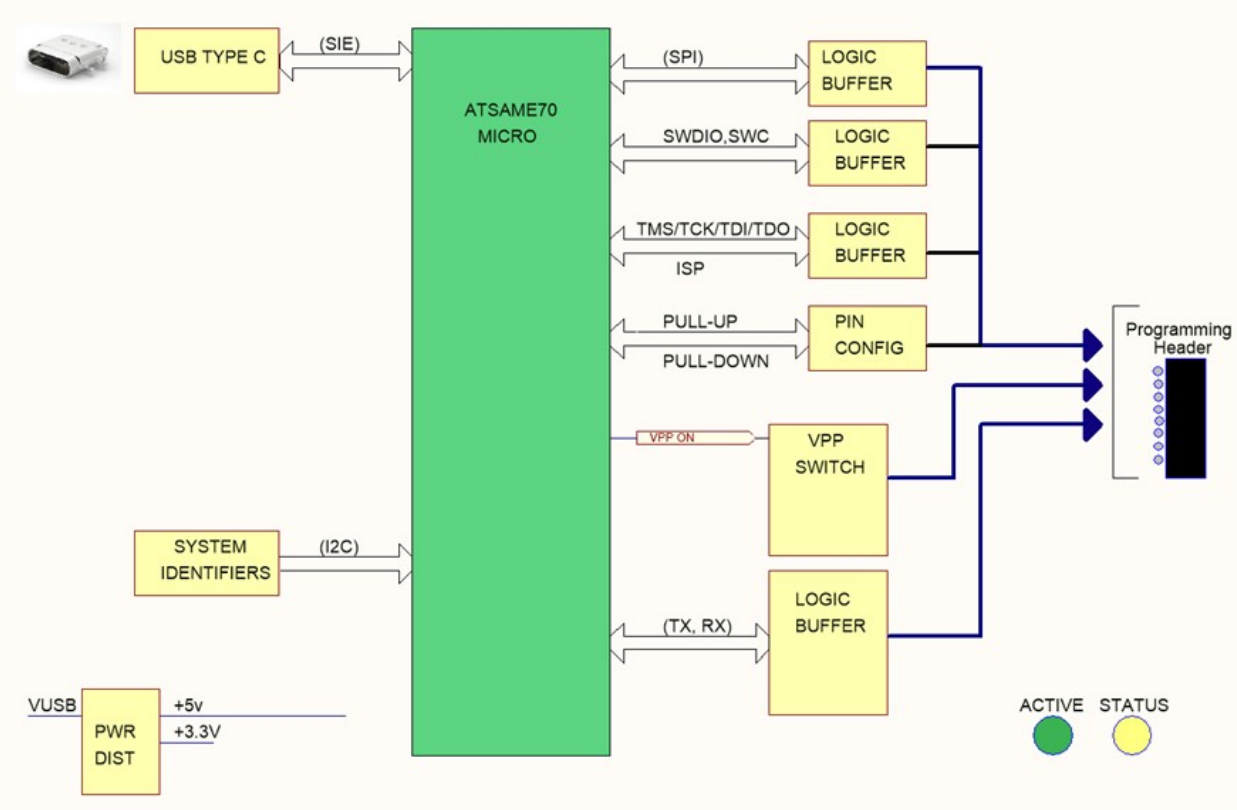
MPLAB PICkit Basic を使うには、以下を用意する必要があります。

- ターゲットボード
- ターゲットボードの電源
- お客様のアプリケーションに必要な全ての接続用インターフェイスまたはケーブル
  - これには AC164110 RJ-11 to ICSP アダプタ等が含まれます。

その他のハードウェアとアクセサリは Microchip Direct から購入できる場合があります。

- デバッガ アダプタボード(製品番号: AC002015)
  - JTAG、SWD、ICSP プロトコルをサポートする接続ボードです。このボードは MPLAB PICkit Basic を使って AVR®をデバッグする場合に使えます。

2.3 MPLAB PICkit Basic インサーキット デバッガのブロック図



2.4 MPLAB PICkit Basic インサーキット デバッガを MPLAB X IDE および MPLAB IPE で使う方法

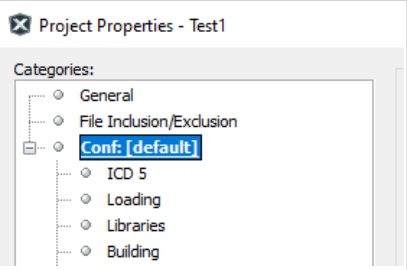
MPLAB X IDE ウェブページから MPLAB® X IDE の最新バージョンをダウンロードしてインストールします。MPLAB X IDE インストーラは MPLAB X IDE または MPLAB IPE (もしくはその両方)をインストールします。

MPLAB® PICkit™ Basic を MPLAB X IDE で使う場合

MPLAB PICkit Basic インサーキット デバッガは、ターゲット アプリケーションを開発するために MPLAB X IDE と連携します。『MPLAB® X IDE ユーザガイド』(DS-50002027)とその他の文書は MPLAB X IDE ウェブページで見つかります。あるいは、MPLAB X IDE ウェブヘルプをご覧ください。

表 2-1. MPLAB X IDE の概要

	デスクトップ アイコンを使って IDE を起動します。
	新規プロジェクトを作成するか、既存プロジェクトを開きます。ハードウェア ツールとして MPLAB PICkit Basic を選択します。

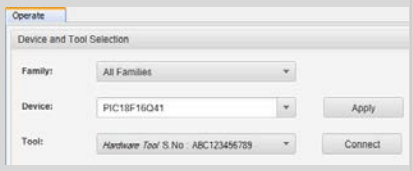
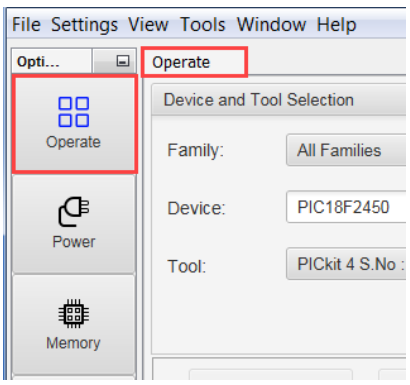
	プロジェクト名を右クリックして「**Properties**」を選択する事により**[Project Properties]**ウィンドウを開きます。このウィンドウを使ってデバッグ、プログラミング、その他の機能のオプションを設定します。MPLAB PICkit Basic オプションの説明を参照してください。

MPLAB PICkit Basic を MPLAB IPE で使う場合

MPLAB PICkit Basic インサーキット デバッガは、プログラマとして MPLAB X IPE と連携します。『MPLAB IPE ユーザガイド』(DS-50002227)とその他の文書は MPLAB X IDE ウェブページで見つかります。あるいは、MPLAB IPE ウェブヘルプをご覧ください。

オプションとして、コマンドライン IPE ツールも使えます。MPLAB X IDE インストール フォルダ内の docs サブフォルダを参照してください。

表 2-2. MPLAB IPE の概要

	デスクトップ アイコンを使って IPE を起動します。
	プログラミングするデバイスを選択し、ツールとして MPLAB PICkit Basic を選択します。
	**[Program]**、**[Erase]**、**[Read]**、**[Verify]**、**[Blank Check]** ボタンのいずれかを選択します。アドバンスト モードを含む MPLAB IPE の詳細は、『MPLAB IPE ユーザガイド』を参照してください。

3. 接続

MPLAB® PICkit™ Basic インサーキット デバッガ ハードウェアのセットアップは電源、通信ライン、ターゲットをデバッガに接続する事により始めます。以下にその詳細を記載します。

3.1 電源接続

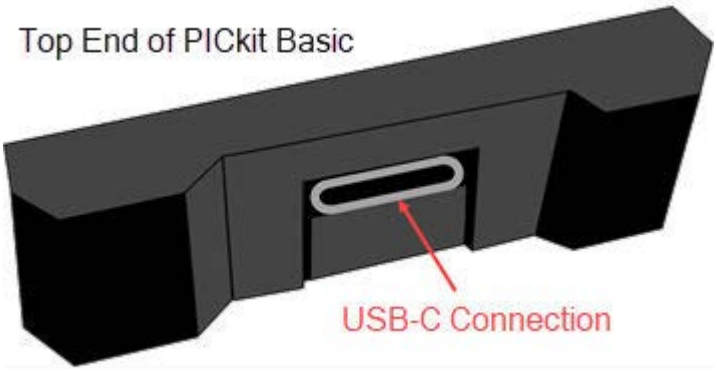
MPLAB PICkit Basic インサーキット デバッガには USB Type-C®コネクタを介して給電します。「9.1. USB コネクタ仕様」も参照してください。

MPLAB PICkit Basic からターゲットに給電する事はできません。ターゲットボードには別の電源から給電する必要があります。

3.2 PC との接続

MPLAB® PICkit™ Basic インサーキット デバッガは、USB を使って PC (および MPLAB X IDE)に接続できます(図 3-1 参照)。通信障害を防ぐため、キットに同梱のケーブルを使う事を推奨します。

図 3-1. USB による電源と通信の接続



接続タイプ	接続仕様	プログラミング [1]	デバッグ [1]	MPLAB Data Visualizer のサポート
USB Type-C®	HS USB 2.0	Yes	Yes	Yes

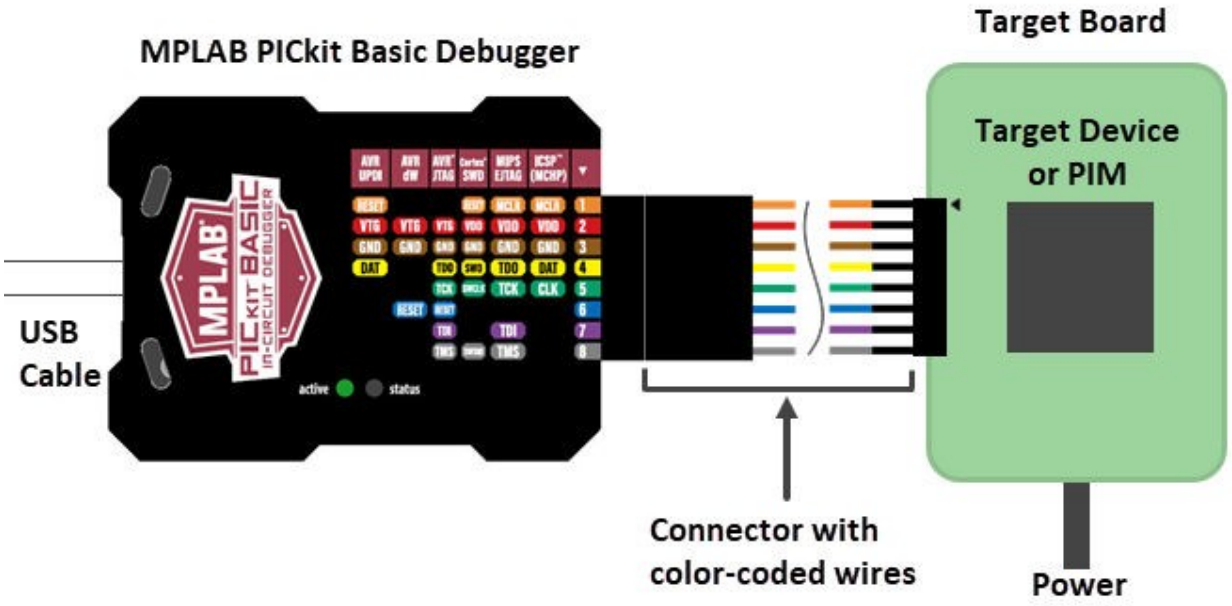
1. 通信速度については「9.2.1. 回路基板の仕様」を参照してください。

3.3 ターゲットとの接続

MPLAB® PICkit™ Basic インサーキット デバッガは、8 ピン シングル インライン(SIL)コネクタから色分けしワイヤを介してターゲットに接続します。デバッガ側のピン 1 をターゲット側のピン 1 (▼)に接続する必要があります(図 3-2 参照)。ターゲットによっては、同梱の Arm/SWD アダプタボードが使える場合があります。

その他のターゲットの接続用にオプションのアダプタボードが購入できます。コネクタとアダプタボードのピン配置については、この後で説明します。

図 3-2. ターゲットとの接続



3.3.1 ターゲット接続のピン配置

プログラミング コネクタのピン機能はデバイスとインターフェイスに応じて異なります。表 3-1 に、デバッグおよびデータストリーム用インターフェイスのピン配置を示します。

Note: その他の情報と接続図については、お使いのデバイスのデータシートと各インターフェイス向けのアプリケーション ノートを参照してください。

表 3-1. デバッグ インターフェイスのピン配置

MPLAB® PICkit™ Basic コネクタ		デバッグ インターフェイス									ターゲット[4] コネクタ	
8 ピン SIL[1]		ICSP™ (MCHP)	MIPS EJTAG	Cortex® SWD	AVR® JTAG	AVR debugWIRE	AVR UPDI	AVR PDI	AVR ISP	AVR TPI	8 ピン SIL	6 ピン SIL
ピン番号	ピン名										ピン番号	ピン番号
1	TVPP	$\overline{\text{MCLR}}$ / VPP	$\overline{\text{MCLR}}$	$\overline{\text{RESET}}$			$\overline{\text{RESET}}$[3]				1	1
2	TVDD	VDD	VDD/ VDDIO	VDD	VTG	VTG	VTG	VTG	VTG	VTG	2	2
3	GND	GND	GND	GND	GND	GND	GND	GND	GND	GND	3	3
4	PGD	DAT	TDO	SWO[2]	TDO		DAT[3]	DAT	MISO	DAT	4	4
5	PGC	CLK	TCK	SWCLK	TCK				SCK	CLK	5	5
6	TAUX				$\overline{\text{RESET}}$	$\overline{\text{RESET}}$/ dW		CLK	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$	6	6
7	TTDI		TDI		TDI				MOSI		7	
8	TTMS		TMS	SWDIO[2]	TMS						8	

1. 6 ピンヘッダを使う場合、EJTAG、JTAG、SWD、ISP に影響するピン 7 とピン 8 の機能が失われます。
2. SWO はトレース用に使います(トレースのサポートについてはリリースノート参照)。SWDIO はデバッグ用に使います。

3. このピンは、UPDI 機能を再アクティブ化するための高電圧パルス用に使えます(デバイスに依存)。通常、少ピン数 AVR デバイスがこれに該当します(詳細はデバイス データシート参照)。しかし、RSTPINCFG コンフィグレーション ビットによって UPDI ピンが GPIO または RESET として設定されている場合、MPLAB PICkit Basic は UPDI インターフェイスを再アクティブ化するための**高電圧パルスを生成できません**。これを行うには、MPLAB PICkit™ 5 等のツールが必要です。
4. これらは、レガシーデバッガに合うように作られたターゲット側接続例です。

表 3-2. データストリーム インターフェイスのピン配置

MPLAB® PICkit™ Basic の 8 ピン SIL コネクタ	データストリーム	ターゲットの 8 ピン SIL コネクタ
ピン番号	UART / CDC[1]	ピン番号
1		1
2	VTG	2
3	GND	3
4		4
5		5
6		6
7	TX(ターゲット)	7
8	RX(ターゲット)	8

1. 詳細は「Readme for PICkit Basic」を参照してください。

3.3.2 8 ピン-10 ピン Arm SWD アダプタボード

8 ピン-10 ピン Arm SWD アダプタボードは MPLAB® PICkit™ Basic インサーキット デバッガに同梱されます。SWD を使う場合、8 ピンコネクタをデバッガに接続し、10 ピンコネクタを Arm ターゲットに接続します(同梱の 12cm フラットケーブルを使用)。

図 3-3. 8 ピン-10 ピン Arm アダプタボード

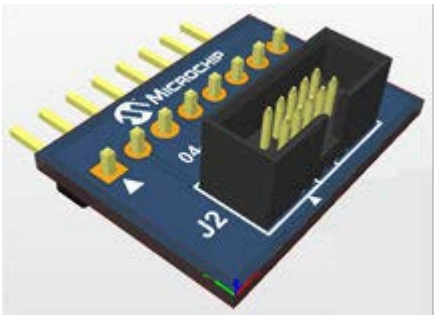


図 3-4. アダプタに同梱の 10 ピン フラットケーブル

関連リンク

9.3.2. Arm/SWD アダプタボード回路図

3.3.3 デバッガアダプタ ボード

レガシー接続用にデバッガ アダプタボード(AC102015)が利用できます。

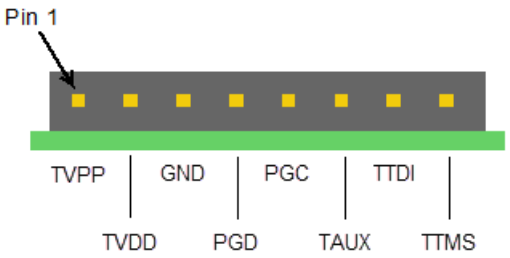
MPLAB PICkit Basic を後述の方法でこのデバッガ アダプタボードに接続し、アダプタボード上の各種コネクタを使ってターゲットに接続します。

この他に MPLAB PICkit 5、PICkit 4、PICkit Basic および Snap 向け AVR プログラミング アダプタボード(AC31S18A)も利用できます。

8 ピン インライン コネクタ

シングル インライン(SIL)コネクタを使うと、ツールを直接アダプタボードに接続できます(ツール側のピン 1 がアダプタボード側のピン 1 に対応するよう接続)。6 ピン SIL ヘッダを使う場合、ピン 7 (TMS)とピン 8 (TDI)の接続はできません。

図 3-5. デバッガ アダプタボード側の SIL コネクタ

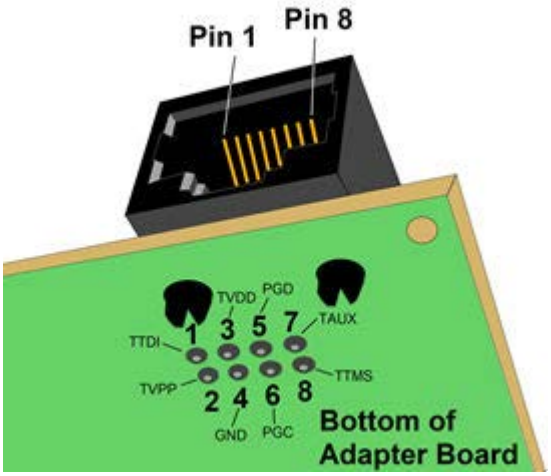


8 ピン モジュラコネクタ

MPLAB PICkit Basic には前述の 8 ピン インライン コネクタの使用を推奨します。

AC164110 - RJ-11 to ICSP アダプタと 6 ピン モジュラケーブルを使って本デバッガをこのモジュラコネクタに接続した場合、アダプタボード側のピン 8 (TMS)とピン 1 (TDI)の接続はできません。

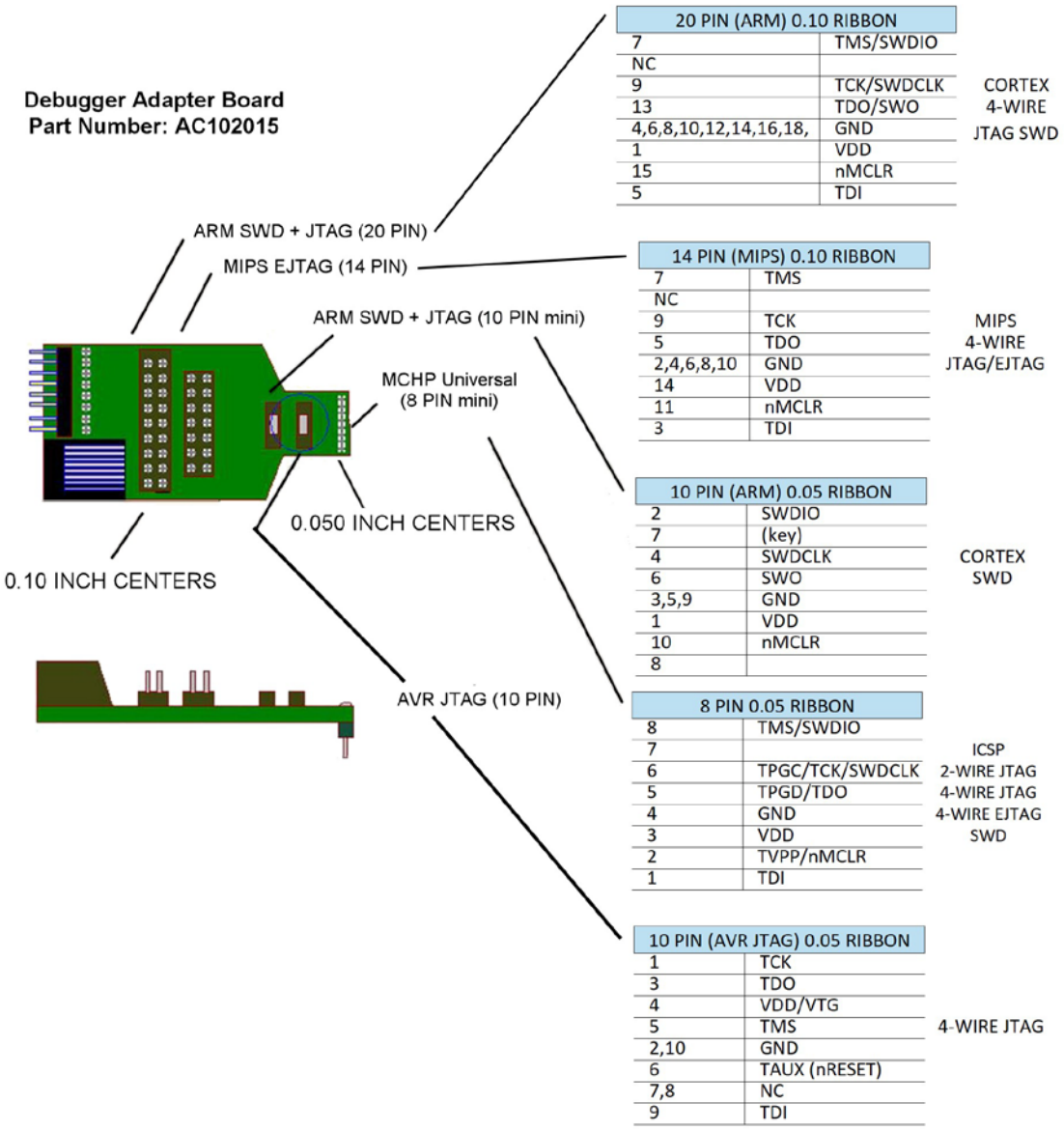
図 3-6. デバッガ アダプタボード側の RJ-45 コネクタ



3.3.3.1 アダプタボードのピン配置

図 3-7 に、JTAG/SWD/ICSP/AVR プロトコルをサポートするアダプタボード(AC102015)のピン配置を示します。

図 3-7. MPLAB PICkit Basic アダプタボード(AC102015)のピン配置



3.3.4 SAM/PIC32C MCU との JTAG/SWD 接続

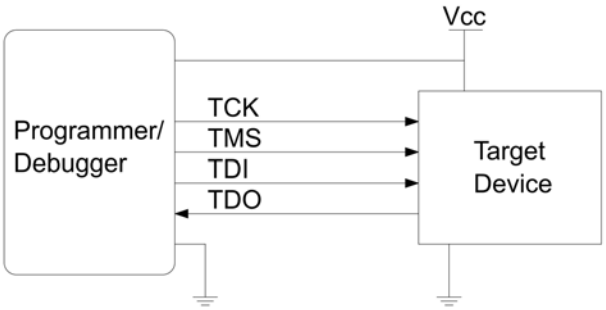
SAM/PIC32C Arm®ベースの全てのデバイスは、プログラミング/デバッグ用に SWD (Serial Wire Debug) インターフェイスを備えています。一部のデバイスは同じ機能を持つ JTAG インターフェイスも追加で備えています。各デバイスがサポートするインターフェイスについては、デバイス データシートを参照してください。

3.3.4.1 JTAG 物理インターフェイス

JTAG インターフェイスは、IEEE® 1149.1 規格に準拠する 4 線式 TAP (Test Access Port)コントローラにより構成されます。IEEE 規格は、回路基板の接続を効率的にテストするための業界標準手法(バウンダリ スキャン)を提供するために開発されました。Microchip 社の AVR および SAM デバイスは、フル プログラミングと内蔵デバッグ(機能)をサポートするためにこの機能を拡張しています。

MPLAB X IDE でこのターゲット インターフェイスを使う場合、**[Project Properties]**ウィンドウを開き、**「PICkit Basic」**カテゴリ内で**「Communications」**オプション カテゴリから**「4-wire JTAG」**を選択します。

図 3-8. JTAG インターフェイスの接続



3.3.4.1.1 SAM/PIC32C JTAG ターゲットへの接続

MPLAB PICkit Basic は最新のターゲットへ直接接続できる他、アダプタボードを使ってレガシーの 10 ピン 50 mil および 20 ピン 100 mil JTAG 接続にも対応します。

関連リンク

3.3.3.1. アダプタボードのピン配置

3.3.4.1.2 SAM/PIC32C JTAG のピン配置(Cortex®-M デバッグコネクタ)

SAM/PIC32C の JTAG ピンの名称と機能を表 3-3 に示します。この表には、MPLAB PICkit Basic をターゲットへ直接接続する場合のピン番号と、デバッガ アダプタボード(10 ピンまたは 20 ピン)を介して接続する場合のピン番号を示しています。

表 3-3. SAM/PIC32C JTAG ピンのピン番号と機能

MPLAB PICkit Basic ピン番号	10 ピン アダプタボード ピン番号	20 ピン アダプタボード ピン番号	名称	機能
8	2	7	TMS	テストモード選択(MPLAB PICkit Basic からターゲット デバイスへの制御信号)
6	7	2、3、9、11、17、19	NC/AUX	接続しない事を推奨
5	4	9	TCK	テストクロック(MPLAB PICkit Basic からターゲット デバイスへのクロック信号)
4	8	5	TDO	テストデータ出力(ターゲット デバイスから MPLAB PICkit Basic へ送信されるデータ)
3	3、5、9	4、6、8、10、12、14、16、18、20	GND	グランド - MPLAB PICkit Basic とターゲット デバイスが同じグランド基準を共有するよう、全ての GND ピンを接続する必要があります。
2	1	1	VDD\VTG*	VDD: MPLAB PICkit Basic からターゲットへの給電(オプション) またはターゲットから MPLAB PICkit Basic への給電(PTG) VTG: ターゲット参照電圧 - MPLAB PICkit Basic は、レベル コンバータに正しく給電するために、このピンでターゲット電圧をサンプリングします。このモードにおいて、MPLAB PICkit Basic はこのピンから 3mA 未満の電流を引き込みます。
1	10	15	$\overline{\text{MCLR}}$	リセット (オプション) - ターゲット デバイスをリセットするために使います。MPLAB PICkit Basic がターゲット デバイスをリセット状態に保持できるようにするため、このピンは接続する事を推奨します。特定状況のデバッグでは、このピンの接続が必須となる場合があります。
7	6	13	TDI	テストデータ入力 (MPLAB PICkit Basic からターゲット デバイスへ送信されるデータ)

* 注意: このピンと GND の間にデカップリング コンデンサが必要です。
『AN4451 - SAMA5D2 Hardware Design Considerations』を参照してください。

3.3.4.2 SAM/PIC32C SWD インターフェイス

Arm® SWD インターフェイスは JTAG インターフェイスのサブセットであり、TCK ピンと TMS ピンを使用します。

3.3.4.2.1 SAM/PIC32C SWD ターゲットへの接続

MPLAB PICkit Basic は最新のターゲットへ直接接続できる他、アダプタボードを使ってレガシーの 10 ピン 50 mil および 20 ピン 100 mil SWD 接続にも対応します。

関連リンク

3.3.3.1.アダプタボードのピン配置

3.3.4.2.2 SAM/PIC32C SWD のピン配置

SAM/PIC32C の SWD ピンの名称と機能を表 3-4 に示します。この表には、MPLAB PICkit Basic をターゲットへ直接接続する場合のピン番号と、デバッガ アダプタボード(10 ピンまたは 20 ピン)を介して接続する場合のピン番号を示しています。

表 3-4. SAM/PIC32C SWD ピンのピン番号と機能

MPLAB PICkit Basic ピン番号	10 ピン アダプタボード ピン番号	20 ピン アダプタボード ピン番号	名称	機能
8	2	7	SWDIO	シリアルワイヤ デバッグ データ入力/出力
5	4	9	SWDCLK	シリアルワイヤ デバッグ クロック
4	8	5	SWO	シリアルワイヤ出力(ITM と一緒に使用、一部のデバイスのみ実装)
3	3、5、9	4、6、8、10、12、	GND	グランド
2	1	1	VDD\VTG*	VDD: MPLAB PICkit Basic からターゲットへの給電(オプション)またはターゲットから MPLAB PICkit Basic への給電(PTG) VTG: ターゲット参照電圧 - MPLAB PICkit Basic は、レベル コンバータに正しく給電するために、このピンでターゲット電圧をサンプリングします。このモードにおいて、MPLAB PICkit Basic はこのピンから 1mA 未満の電流を引き込みます。
1	10	15	$\overline{\text{MCLR}}$	リセット (オプション) - ターゲット デバイスをリセットするために使います。MPLAB PICkit Basic がターゲット デバイスをリセット状態に保持できるようにするため、このピンは接続する事を推奨します。特定状況のデバッグでは、このピンの接続が必須となる場合があります。

* 注意: このピンと GND の間にデカップリング コンデンサが必要です。
『AN4451 - SAMA5D2 Hardware Design Considerations』を参照してください。

3.3.5 AVR MCU の各種接続方法

AVR デバイスは各種のプログラミング/デバッグ インターフェイスを備えています。各デバイスがサポートするインターフェイスについては、デバイス データシートを参照してください。

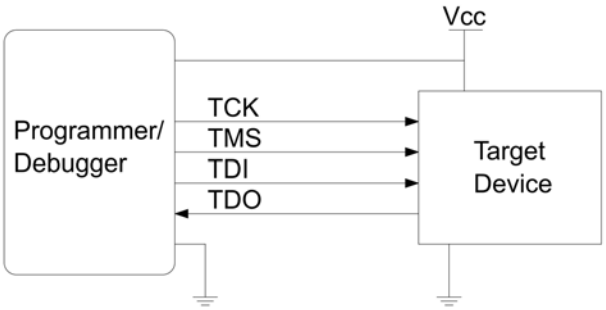
3.3.5.1 AVR MCU - JTAG 接続

一部の AVR デバイスはプログラミング/デバッグ用に JTAG インターフェイスを備えています。各デバイスがサポートするインターフェイスについては、デバイス データシートを参照してください。

3.3.5.1.1 JTAG 物理インターフェイス

JTAG インターフェイスは、IEEE® 1149.1 規格に準拠する 4 線式 TAP (Test Access Port)コントローラにより構成されます。IEEE 規格は、回路基板の接続を効率的にテストするための業界標準手法(バウンダリ スキャン)を提供するために開発されました。Microchip 社の AVR および SAM デバイスは、フル プログラミングと内蔵デバッグをサポートするためにこの機能を拡張しています。MPLAB X IDE でこのターゲット インターフェイスを使う場合、**[Project Properties]**ウィンドウを開き、**「PICkit Basic」**カテゴリ内で**「Communications」**オプション カテゴリから**「4-wire JTAG」**を選択します。

図 3-9. JTAG インターフェイスの接続



AVR JTAG ターゲットへの接続

MPLAB PICkit Basic は最新のターゲットへ直接接続できる他、アダプタボードを使ってレガシーの 10 ピン 50 mil JTAG 接続にも対応します。

関連リンク

3.3.3.1.アダプタボードのピン配置

AVR JTAG のピン配置

AVR JTAG ピンの名称と機能を表 3-5 に示します。この表には MPLAB PICkit Basic をターゲットへ直接接続する場合のピン番号と、デバッガ アダプタボード(10 ピン)を介して接続する場合のピン番号を示しています。

表 3-5. AVR JTAG ピンのビン番号と機能

MPLAB PICkit Basic ピン番号	アダプタボード AVR JTAG ピン番号	名称	機能
1	7、8	NC	未接続
2	4	VDD\VTG*	VDD: MPLAB PICkit Basic からターゲットへの給電(オプション)またはターゲットから MPLAB PICkit Basic への給電(PTG) VTG: ターゲット参照電圧 - MPLAB PICkit Basic は、レベル コンバータに正しく給電するために、このピンでターゲット電圧をサンプリングします。このモードにおいて、MPLAB PICkit Basic はこのピンから 3mA 未満の電流を引き込みます。
3	2、10	GND	グランド - MPLAB PICkit Basic とターゲット デバイスが同じグランド基準を共有するよう、全ての GND ピンを接続する必要があります。
4	3	TDO	テストデータ出力(ターゲット デバイスから MPLAB PICkit Basic へ送信されるデータ)
5	1	TCK	テストクロック(MPLAB PICkit Basic からターゲット デバイスへのクロック信号)
6	6	$\overline{\text{RESET}}$/TAUX	リセット (オプション) - ターゲット デバイスをリセットするために使います。MPLAB PICkit Basic がターゲット デバイスをリセット状態に保持できるようにするため、このピンは接続する事を推奨します。特定状況のデバッグでは、このピンの接続が必須となる場合があります。
7	9	TDI	テストデータ入力(MPLAB PICkit Basic からターゲット デバイスへ送信されるデータ)
8	5	TMS	テストモード選択(MPLAB PICkit Basic からターゲット デバイスへの制御信号)

* 注意: このピンと GND の間にデカップリング コンデンサが必要です。
『AVR042 - AVR Hardware Design Considerations』を参照してください。

3.3.5.2 AVR SPI インターフェイス

インシステム プログラミングは、ターゲット AVR デバイスの内蔵 SPI を使って、コードをフラッシュおよび EEPROM メモリへダウンロードします。これはデバッグ用インターフェイスではありません。

3.3.5.2.1 AVR SPI ターゲットへの接続

MPLAB PICkit Basic は最新のターゲットへ直接接続できる他、アダプタボードを使ってレガシーの 10 ピン 50mil JTAG 接続にも対応します。

重要:
debugWIRE Enable (DWEN)ヒューズが設定されている場合、たとえ SPIEN ヒューズが設定されていても PI インターフェイスは無効になります。SPI インターフェイスを有効にするには、debugWIRE デバッグ セッション中に「disable debugWIRE」コマンドを発行する必要があります。この方法で debugWIRE を無効にする場合、SPIEN ヒューズを予め設定しておく必要があります。MPLAB X IDE が debugWIRE の無効化に失敗する場合、SPIEN ヒューズが設定されていない可能性があります。その場合、SPIEN ヒューズを設定するために高電圧 プログラミング インターフェイスが必要となります。

Info:
SPI インターフェイスは Microchip 社の AVR 製品で初めて採用されたインシステム プログラミング インターフェイスであるため、「ISP」(In-System Programming)インターフェイスと呼ばれる事があります。現在では、インシステム プログラミング向けにその他のインターフェイスも利用可能となっています。

関連リンク
3.3.3.1.アダプタボードのピン配置

3.3.5.2.2 AVR SPI のピン配置

SPI インターフェイスを備えた AVR デバイスを使用するレガシー アプリケーションの PCB (プリント基板)では、図 3-10 に示すピン配置が使われている場合があります。

図 3-10. レガシーSPI ヘッダのピン配置

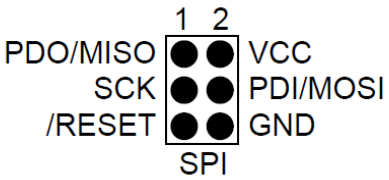


表 3-6. SPI ピンの割り当て

MPLAB PICkit Basic ピン番号	アダプタボード AVR JTAG ピン番号	レガシーSPI ピン番号	名称	機能
1	7 または 8 (NC)			
2	4 (VDD/VTG)	2	VCC/VTG	ターゲット電圧(参照電圧)
3	2 または 10 (GND)	6	GND	グランド
4	3 (TDO)	1	MISO	Host In Client Out (Main In Secondary Out)
5	1 (TCK)	3	SCK	SPI クロック
6	6 (RESET/TAUX)	5	RESET	
7	9 (TDI)	4	MOSI	Host Out Client In (Main Out Secondary In)
8	5 (TMS)			

3.3.5.3 AVR UPDI インターフェイス

UPDI (Unified Program and Debug Interface)は、デバイスの外部プログラミングと内部デバッグが可能な Microchip 社独自のインターフェイスです。UPDI は PDI 2 線式物理インターフェイスの後継技術であり、全ての AVR XMEGA デバイスに搭載されています。UPDI は、プログラミング/デバッグ用にターゲット デバイスとの双方向-半二重-非同期通信を提供する単線式インターフェイスです。

現在、UPDI には以下の 3 通りのコンフィグレーションが存在します。

1. UPDI 機能は、RESET または GPIO 向けにも使用可能な共有ピンに割り当てられます。RESET または GPIO が選択されている時に UPDI が必要になった場合、UPDI 機能を有効にするためにそのピンで HV(高電圧)パルスが必要です。

   このコンフィグレーションは旧型の AVR デバイス(ATtiny)で使われ、12V の HV パルスが必要です。
2. UPDI 機能は専用ピンに割り当てられ、常時利用可能です。RESET と GPIO はピンを共有します。

   このコンフィグレーションは新しい AVR デバイス(ATmega0、AVR DA/DB 等)で使われ、HV パルスは不要です。
3. UPDI 機能は、GPIO 向けにも使用可能な共有ピンに割り当てられます。RESET は専用ピンに割り当てられます。GPIO が選択されている時に UPDI が必要になった場合、UPDI 機能を有効にするために RESET ピンで HV パルスが必要です。

   このコンフィグレーションは新しい AVR デバイス(AVR DD)で使われ、V_{DD} + 約 2V の HV パルスが必要です。実際の電圧レンジとデバイスが使用するコンフィグレーションについては、デバイス データシートを参照してください。

3.3.5.3.1 AVR UPDI ターゲットへの接続

MPLAB PICkit Basic は最新のターゲットへ直接接続できる他、アダプタボードを使って 6 ピン UPDI インターフェイス向けにレガシーの 10 ピン 50 mil JTAG 接続にも対応します。

関連リンク

3.3.3.1.アダプタボードのピン配置

3.3.5.3.2 AVR UPDI のピン配置

UPDI インターフェイスを備えた AVR デバイスを使用するレガシー アプリケーション PCB (プリント基板)では、全てのコンフィグレーションで図 3-11 に示すピン配置が使われている可能性があります。

図 3-11. レガシーUPDI ヘッダのピン配置

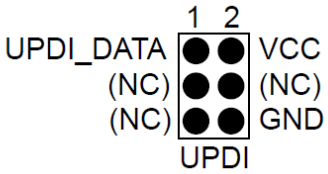


表 3-7. レガシーUPDI ピンの割り当て

MPLAB PICkit Basic ピン番号	アダプタボード AVR JTAG ピン番号	レガシーUPDI ピン番号	名称	機能
1	7 または 8 (NC)			
2	4 (VDD/VTG)	2	VCC/VTG	ターゲット電圧(参照電圧)
3	2 または 10 (GND)	6	GND	グランド
4	3 (TDO)	1	UPDI_DATA*	UPDI データデバイスによっては他のピン機能が可能
5	1 (TCK)			
6	6 (RESET/TAUX)			
7	9 (TDI)			
8	5 (TMS)			

* UPDI 機能を使うためにこのピンで高電圧(HV)パルスが必要になる場合があります。UPDI コンフィグレーションについては、デバイス データシートを参照してください。

3.3.5.4 AVR PDI インターフェイス

PDI (Program and Debug Interface)は、デバイスの外部プログラミングと内部デバッグが可能な Microchip 社独自のインターフェイスです。PDI はターゲット デバイスとの双方向-半二重-同期通信を提供する 2 ピン インターフェイスです。

3.3.5.4.1 AVR PDI ターゲットへの接続

MPLAB PICkit Basic は最新のターゲットへ直接接続できる他、アダプタボードを使って 6 ピン PDI インターフェイス向けにレガシーの 10 ピン 50 mil JTAG 接続にも対応します。

関連リンク

3.3.3.1.アダプタボードのピン配置

3.3.5.4.2 AVR PDI のピン配置

PDI インターフェイスを備えた AVR デバイスを使用するレガシー アプリケーションの PCB (プリント基板)では、図 3-12 に示すピン配置が使われている可能性があります。

図 3-12. レガシーPDI ヘッダのピン配置

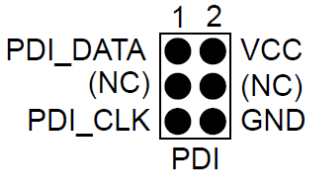


表 3-8. PDI ピンの割り当て

MPLAB PICkit Basic ピン番号	アダプタボード AVR JTAG ピン番号	レガシーPDI ピン番号	名称	機能
1	7 または 8 (NC)			
2	4 (VDD/VTG)	2	VCC/VTG	ターゲット電圧(参照電圧)
3	2 または 10 (GND)	6	GND	グランド
4	3 (TDO)	1	PDI_DATA	PDI データ
5	1 (TCK)			
6	6 (RESET/TAUX)	5	PDI_CLK	PDI クロック
7	9 (TDI)			
8	5 (TMS)			

3.3.5.5 AVR TPI インターフェイス

TPI は、一部の tinyAVR®デバイス向けのプログラミング専用インターフェイスです。TPI はデバッグ インターフェイスではなく、これらのデバイスは内蔵デバッグ(OCD)機能を備えていません。

3.3.5.5.1 AVR TPI ターゲットへの接続

MPLAB PICkit Basic は最新のターゲットへ直接接続できる他、アダプタボードを使ってレガシーの 10 ピン 50 mil JTAG 接続にも対応します。

関連リンク

3.3.3.1. アダプタボードのピン配置

3.3.5.5.2 AVR TPI のピン配置

TPI インターフェイスを備えた AVR デバイスを使用するレガシー アプリケーションの PCB (プリント基板)では、図 3-13 に示すピン配置が使われている可能性があります。

図 3-13. レガシーTPI ヘッダのピン配置

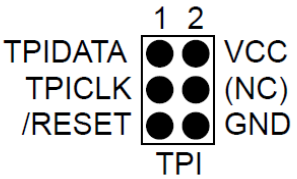


表 3-9. TPI ピンの割り当て

MPLAB PICkit Basic ピン番号	アダプタボード AVR JTAG ピン番号	レガシーTPI ピン番号	名称	機能
1	7 または 8 (NC)			
2	4 (VDD/VTG)	2	VCC/VTG	ターゲット電圧(参照電圧)
3	2 または 10 (GND)	6	GND	グランド
4	3 (TDO)	1	DATA	TPI データ
5	1 (TCK)	3	CLOCK	TPI クロック
6	6 (RESET/TAUX)	5	RESET	デバイスをリセットします。
7	9 (TDI)			
8	5 (TMS)			

3.3.5.6 AVR debugWIRE インターフェイス

debugWIRE インターフェイスは少ピン数デバイス用です。4 本のピンを使う JTAG インターフェイスとは異なり、debugWIRE はたった 1 本のピン(RESET)を使ってデバッガツールとの双方向-半二重-非同期通信を提供します。

Note:
debugWIRE インターフェイスはプログラミング インターフェイスとして使えません。このため、ターゲットをプログラミングするために SPI インターフェイスも必要です(3.3.5.2. 「AVR SPI インターフェイス」参照)。

debugWIRE を使ってデバッグ セッションを起動すると、フラッシュは debugWIRE インターフェイスを使って書き込まれます。これは工場プログラミング向けに適したオプションではありません。

ロックビットが設定されていなければ、debugWIRE イネーブル(DWEN)ヒューズが設定さた時にターゲット デバイス内の debugWIRE システムが有効になります。RESET ピンはワイヤード AND (オープンドレイン)の双方向 I/O ピン (プルアップは有効)として設定され、ターゲットとデバッガの間の通信ゲートウェイとして機能します。

3.3.5.6.1 AVR debugWIRE ターゲットへの接続

MPLAB PICkit Basic は最新のターゲットへ直接接続できる他、アダプタボードを使って 6 ピン debugWIRE/SPI インターフェイス向けにレガシーの 10 ピン 50 mil JTAG 接続にも対応します。

debugWIRE インターフェイスは 1 本の信号ライン(RESET)と VCC および GND だけで動作しますが、SPI プログラミングを使って debugWIRE インターフェイスの有効化/無効化ができるよう、SPI コネクタの全てのピンへのアクセスを可能にする事を推奨します。

DWEN ヒューズが有効である場合、OCD モジュールが RESET ピンを制御するため、SPI インターフェイスは内部でオーバーライドされます。debugWIRE OCD は自身を一時的に無効にして RESET ラインの制御を解放する事ができます。これにより SPI インターフェイスが再び利用可能になり(SPIEN ヒューズが設定されている場合のみ)、SPI インターフェイスを使って DWEN ヒューズを未設定にする事ができます。DWEN を未設定にする前に電源が切れて再投入された場合、debugWIRE モジュールは再び RESET ピンを制御します。
通常、MPLAB X IDE または Microchip Studio は自動的にインターフェイスを切り換えますが、Microchip Studio の**[Properties]**ダイアログ内で**[Debugging]**タブ上のボタンを使って手動で切り換える事もできます。

Note: DWEN ヒューズのセット/クリアは、MPLAB X IDE または Microchip Studio に処理させる事を強く推奨します。

ターゲット AVR デバイスのロックビットが設定されている場合、debugWIRE インターフェイスを使う事はできません。DWEN ヒューズを設定する前にロックビットがクリアされている事を必ず確認し、DWEN ヒューズが設定されている間はロックビットを決してセットしてはいけません。debugWIRE イネーブル(DWEN)ヒューズとロックビットの両方がセットされている場合、高電圧プログラミングを使ってチップ消去を実行する事によりロックビットをクリアできます。ロックビットがクリアされると debugWIRE インターフェイスは再び有効になります。DWEN ヒューズが未設定である場合、SPI インターフェイスはヒューズの読み出し、シグネチャの読み出し、チップ消去の実行しかできません。

関連リンク
3.3.3.1. アダプタボードのピン配置

3.3.5.6.2 AVR debugWIRE のピン配置

debugWire インターフェイスを備えた AVR デバイスを使用するレガシー アプリケーションの PCB (プリント基板)では、図 3-14 に示すピン配置が使われている可能性があります。

図 3-14. レガシーdebugWIRE (SPI)ヘッダのピン配置

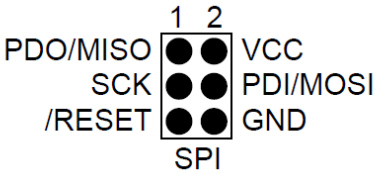


表 3-10. debugWIRE ピンの割り当て

MPLAB PICkit Basic ピン番号	アダプタボード AVR JTAG ピン番号	レガシーdebugWIRE ピン番号	名称	機能
1	7 または 8 (NC)			
2	4 (VDD/VTG)	2	VCC/TTG	ターゲット電圧(参照電圧)
3	2 または 10 (GND)	6	GND	グランド
4	3 (TDO)			
5	1 (TCK)			
6	6 ($\overline{\text{RESET}}$/TAUX)	5	$\overline{\text{RESET}}$ または debugWIRE	デバイスのリセットまたは debugWIRE データの送信用にピンを使用
7	9 (TDI)			
8	5 (TMS)			

3.3.6 PIC32M の接続

PIC32M MIPS ベース デバイスはデバッグとプログラミング用に EJTAG を使います。

3.3.6.1 PIC32M EJTAG ターゲットへの接続

MPLAB PICkit Basic は最新のターゲットへ直接接続できる他、アダプタボードを使ってレガシーの 14 ピン 10 mil JTAG/EJTAG 接続にも対応します。

3.3.6.2 PIC32M EJTAG のピン配置 - 4 線式 JTAG

PIC32M EJTAG ピンの名称と機能を表 3-11 に示します。この表には MPLAB PICkit Basic をターゲットへ直接接続する場合のピン番号と、デバッガ アダプタボード(14 ピン)を介して接続する場合のピン番号を示しています。

表 3-11. PIC32M JTAG コネクタ(14 ピン)のピン機能

MPLAB PICkit Basic ピン番号	14 ピン アダプタボード ピン番号	名称	機能
1	11	$\overline{\text{MCLR}}$	リセット (オプション) - ターゲット デバイスをリセットするために使います。MPLAB PICkit Basic がターゲット デバイスをリセット状態に保持できるようにするため、このピンは接続する事を推奨します。特定状況のデバッグでは、このピンの接続が必須となる場合があります。
2	14	VDD	MPLAB PICkit Basic からターゲットへの給電(オプション)またはターゲットから MPLAB PICkit Basic への給電(PTG)
3	2, 4, 6, 8, 10	GND	グランド - MPLAB PICkit Basic とターゲット デバイスが同じグランド基準を共有するよう、全ての GND ピンを接続する必要があります。
4	3	TDO	テストデータ出力(ターゲット デバイスから MPLAB PICkit Basic へ送信されるデータ)
5	9	TCK	テストクロック(MPLAB PICkit Basic からターゲット デバイスへのクロック信号)

表 3-11. PIC32M JTAG コネクタ(14 ピン)のピン機能

MPLAB PICkit Basic ピン番号	14 ピン アダプタボード ピン番号	名称	機能
6	1	NC	未接続
7	5	TDI	テストデータ入力(MPLAB PICkit Basic からターゲット デバイスへ送信されるデータ)
8	7	TMS	テストモード選択(MPLAB PICkit Basic からターゲット デバイスへの制御信号)

3.3.7 PIC MCU - ICSP 接続

MPLAB® PICkit™ Basic インサーキット デバッガは、ICSP™ (In-Circuit Serial Programming™)接続を介して PIC マイクロコントローラ(MCU)と dsPIC デジタルシグナル コントローラ(DSC)のデバッグおよびプログラミングをサポートします。PICkit Basic SIL コネクタは 2 本のデバイス I/O ピンとリセットラインを使ってインサーキット デバッグと ICSP を実装します。

3.3.7.1 ICSP ターゲットの接続

ICSP®モジュラコネクタを使うか、大部分の MPLAB®デバッグツールが備えるインライン コネクタを使って、デバッガを直接 PIC® MCU ターゲットに接続します。デバッガ アダプタボードへの接続は、ターゲットボードへの接続と同じです。

デバッガとターゲットでコネクタが異なる場合(モジュラ-インラインまたはインライン-モジュラ接続)、別売の小型アダプタ(RJ11 to ICSP アダプタ (AC164110))を使って正しく接続できます。

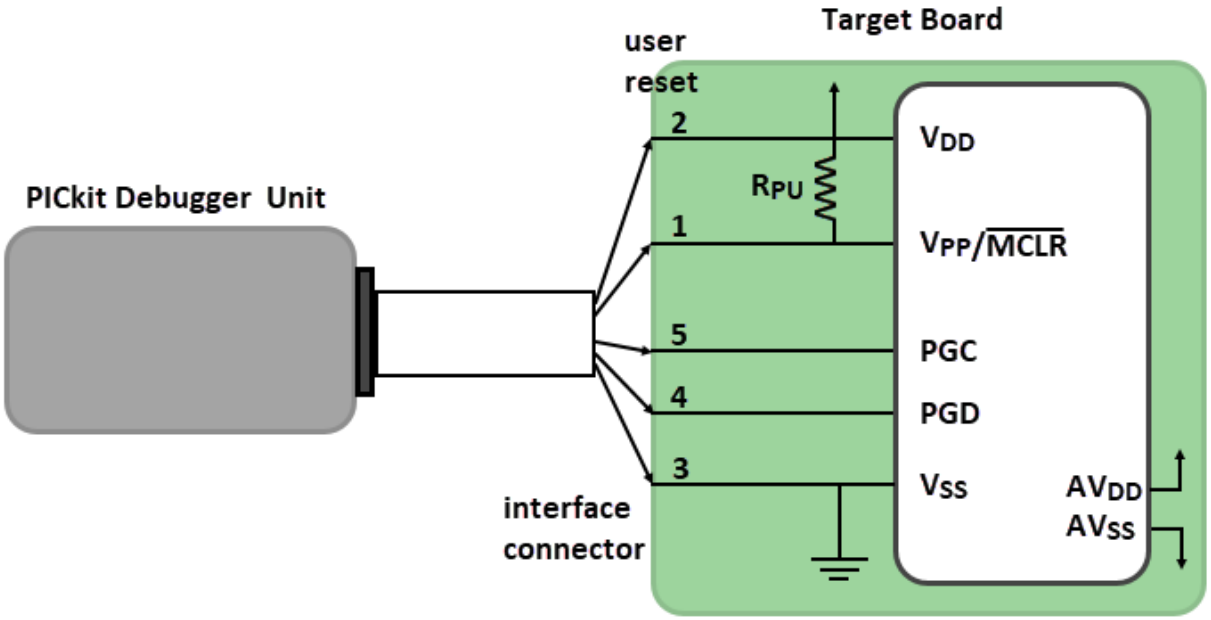
図 3-15. 6 ピン RJ11 to ICSP アダプタ

あるいは、アダプタボードを使う事で、MPLAB PICkit Basic は 6 ピンおよび 8 ピン ICSP インターフェイス向けに 8 ピン 50 mil Microchip Universal 接続を提供します。

3.3.7.2 ICSP™ターゲット接続回路

図 3-16 に、MPLAB PICkit Basic インサーキット デバッガからターゲットボード上の ICSP コネクタへの接続を示します。この図には、ICSP コネクタからターゲット PCB 上のデバイスへの配線も示しています。V_{PP}/ $\overline{MCLR}$ ラインと V_{DD} の間にプルアップ抵抗 R_{PU} を接続する事を推奨します。これにより、このラインを Low にする事によってデバイスをリセットできます。

図 3-16. ICSP ターゲット接続



Rpu=10-50 kΩ

4. 動作

以下では、MPLAB® PICkit™ Basic インサーキット デバッガシステムの動作原理を簡潔に説明します。これは、本デバッガを使ってデバッグ/プログラミングが行えるようにターゲットボードを設計するための情報を提供する事を目的としています。また、問題が発生してもすぐに解決できるよう、インサーキット デバッグ/プログラミングの基本原理についても説明します。

4.1 デバッグ/プログラミングのクイック リファレンス

表 4-1 に、MPLAB PICkit B asic インサーキット デバッガをデバッグツールまたはプログラミング ツールとして使うためのクイック リファレンスを示します。

注意: MPLAB PICkit Basic はデバッグヘッダをサポートしません。

表 4-1. デバッグ動作とプログラミング動作

項目	デバッグ	プログラミング
必要ハードウェア	コンピュータ、ターゲットボード(Microchip 社デモボードまたはお客様独自の回路)、ケーブル、必要に応じてアダプタボード (コンピュータをターゲットに接続するために使用)	
	デバッグツール、USB ケーブル、ターゲット電源	
	内蔵デバッグ回路を備えたデバイス	デバイス(内蔵デバッグ回路の有無を問わず)
必要ソフトウェア	MPLAB X IDE	MPLAB X IDE または MPLAB IPE
	アプリケーション コード (MPLAB Discover 等から入手したサンプルコードまたはお客様独自のコード)	ビルド済みコード - Hex ファイル
MPLAB X IDE	**[Project Properties]**でハードウェア ツールとして PICkit Basic を選択	
	[Debug Main Project]アイコン()	**[Make and Program Device]**アイコン()
プログラミング動作	アプリケーション コードをデバイスにプログラミングします。**[Project Properties]**ダイアログ内の選択に応じてプログラムメモリの任意の領域にプログラミングできます。加えて、小さなデバッグ実行プログラムをプログラムメモリ内に配置する事で、その他のデバッグ リソースを確保できます。	アプリケーション コードをデバイスにプログラミングします。**[Project Properties]**ダイアログ内の選択に応じてプログラムメモリの任意の領域にプログラミングできます。
利用可能なデバッグ機能	デバイス向けの全機能 - ブレークポイント等	N/A
シリアル クイックタイム プログラミング(SQTP)	N/A	MPLAB IPE を使って SQTP ファイルを生成
コマンドライン操作	MDB コマンドライン ユーティリティ(既定値パス: `C:\Program Files\Microchip\ MPLABX\vx.xx\mplab_platform\bin\mdb.ba t`)を使用	IPECMD (既定値パス: `C:\Program Files\Microchip\ MPLABX\vx.xx\mplab_platform\mplab_ipe\ ipecmd.exe`)を使用

4.2 動作の概要

Note: 下記の手順 5 を実行する前に、「3.2. PC の接続」と「3.3. ターゲットの接続」 を読んでください。

MPLAB PICkit Basic インサーキット デバッガを使ってコードをデバッグ/プログラミングする手順は以下の通りです。

1. MPLAB X IDE ウェブページから最新の MPLAB X IDE をダウンロードしてインストールします。MPLAB IPE は MPLAB X IDE インストーラに含まれます。アプリケーション コードを開発してデバッグするためにプロジェクトを作成する方法とコードをデバッグする方法は、MPLAB® X IDE User's Guide を参照してください。
2. MPLAB Discover 内でサンプルコードを見つけるか、GitHub 上で Microchip 社コンテンツ(Microchip PIC & AVR Examples 等)を検索します。

3. MPLAB XC コンパイラ ウェブページで、お客様のアプリケーション デバイスに対応するコンパイラを見つけます。
4. MPLAB PICkit Basic (必要に応じてデバッガ アダプタボード)を購入します。
5. MPLAB X IDE を起動します。USB ケーブルを使って、MPLAB PICkit Basic をコンピュータに接続します。ターゲットが正しく接続されている事を確認します。
6. MPLAB X IDE 内でお客様独自のプロジェクトまたはサンプル プロジェクトを開きます。**[Projects]**タブ内でプロジェクト名を右クリックし、**[Properties]**を選択します。**[Project Properties]**ダイアログ内の**「Connected Hardware Tool」**の下で**「PICkit Basic」**が選択されている事を確認します。次に、**「Conf(iguration) and setup options」**の下で**[Option Categories]**リストから**「PICkit Basic Category」**を選択します。
   Note: ここでは、必要に応じてターゲットへの電源も選択します。
7. お客様のアプリケーション デバイスの詳細は後述を参照してください。
8. お客様のデバイスに対するデバッガの制限事項を記載した完全なリストは、MPLAB X IDE 内でオンライン Help ファイル(**Help** > **Help Contents** > **Hardware Tool Reference Help** > **Limitations - Emulators and Debuggers**)を開いてご覧になれます。

4.3 「高電圧」について

「高電圧」という用語は各種の意味で用いられてきました。この用語の過去から現在にいたる定義について以下で説明します。

AVR®デバイス

高電圧プログラミング - HVSP と HVPP

古い AVR デバイスは、高電圧プログラミングとして知られているプログラミング インターフェイスのシリアル(HVSP)バリアントとパラレル(HVPP)バリアントの両方を備えています。一般的にこのインターフェイスは、プログラミング セッション中に 12V を $\overline{\text{RESET}}$ ピンへ印加する事を要求します。

コンフィグレーション ビット(ヒューズ)が誤ってセット/クリアされた場合、それらを修復するために高電圧プログラミングが必要です。

例:

- DWEN とロックビットがセットされている場合: debugWIRE が使えない
- DWEN がセットされ SPIEN がクリアされている場合: debugWIRE モード中のスタック時に SPI が使えない
- JTAGEN がクリアされている場合: JTAG が使えない

ツールサポート: 最新の全ての MPLAB®ハードウェア ツールはこのプログラミング方式をサポートしません。従って、上記の問題に関する全ての警告に注意し、指示に従う事が重要です。

高電圧パルス - UPDI ピン

ピンの UPDID 機能を有効にするために高電圧(HV)パルスを要求するコンフィグレーションは、デバイスにもよりますが、現在 2 つ存在します。

1. UPDI 機能は、$\overline{\text{RESET}}$ または GPIO 向けにも使用可能な共有ピンに割り当てられます。$\overline{\text{RESET}}$ または GPIO が選択されている時に UPDI が必要になった場合、UPDI 機能を有効にするためにそのピンで HV(高電圧)パルスが必要です。
   このコンフィグレーションは古い AVR デバイスで使われ、12V の HV パルスが必要です。
2. UPDI 機能は、GPIO 向けにも使用可能な共有ピンに割り当てられます。$\overline{\text{RESET}}$ は専用ピンに割り当てられます。GPIO が選択されている時に UPDI が必要になった場合、UPDI 機能を有効にするために $\overline{\text{RESET}}$ ピンで HV パルスが必要です。
   このコンフィグレーションは新しい AVR デバイスで使われ、V_{DD} + 約 2V の HV パルスが必要です。実際の電圧レンジとデバイスが使用するコンフィグレーションについては、デバイス データシートを参照してください。

ツールサポート: MPLAB PICkit Basic と MPLAB Snap (高電圧を非サポート)を除く全ての最新 MPLAB ハードウェア ツールは、どちらの HV パルスもサポートします。

PIC®デバイス

高電圧プログラミングと低電圧プログラミング

高電圧プログラミングの場合、古い PIC デバイスでは高電圧(12V)でプログラミングする必要がありますが、新しいデバイスではそれよりも低い電圧(V_{PP} ピンで 9V を上回る電圧)が使えます。

低電圧プログラミングの場合、プログラミング電圧 V_{PP} は V_{DD} を超えません。

ツールサポート: 以下を除く全ての最新 MPLAB ハードウェア ツールは、高電圧プログラミングと低電圧プログラミングをサポートします。

- MPLAB PICkit Basic と MPLAB Snap (高電圧は非サポート)
- MPLAB PICkit™ 4 (デバイスの V_{DD} 電圧が 2.8V 以上である場合にのみ高電圧プログラミングをサポート。問題は 10-10094-R6 で修正済み。ユニット背面のラベル参照)

4.4 SAM および PIC32C Arm デバイスの内蔵デバッグ機能

SAM および PIC32C マイクロコントローラはどちらも Arm® Cortex-M®コアを搭載しています。コアのタイプに応じて利用可能なデバッグ機能が異なります(表 4-2 参照)。デバッグコネクタは SWD と JTAG をサポートします。

各デバイスのコアタイプの詳細はウェブページ「32-bit PIC® and SAM Microcontrollers」またはデバイスのデータシートを参照してください。Arm が提供する「CoreSight documentation」も参照してください。

表 4-2. Cortex-M コアのデバッグ/トレースサポート

Cortex-M タイプ	デバッグサポート
Cortex-M0+	デバッグ オプション: 基本的デバッグ機能はプロセッサ停止、シングルステップ、プロセッサコア レジスタアクセス、リセットおよび HardFault ベクタキャッチ、無制限のソフトウェア ブレークポイント、フルシステム メモリアクセスを含みます。1/2/3/4 ブレークポイントと 1/2 ウォッチポイント機能も含みます。
Cortex-M23	デバッグ オプション: 基本的デバッグ機能はプロセッサ停止、シングルステップ、プロセッサコア レジスタアクセス、リセットおよび HardFault ベクタキャッチ、無制限のソフトウェア ブレークポイント、フルシステム メモリアクセスを含みます。1/2/3/4 ブレークポイントと 1/2/3/4 ウォッチポイント機能も含みます。
Cortex-M4, M4F	デバッグ オプション: 基本的デバッグ機能はプロセッサ停止、シングルステップ、プロセッサコア レジスタアクセス、ベクタキャッチ、無制限のソフトウェア ブレークポイント、フルシステム メモリアクセスを含みます。各種ブレークポイントおよび 1/4 ウォッチポイント機能も含みます。
Cortex-M7	Cortex-M7 デバッグ機能はプロセッサ停止、シングルステップ、プロセッサコア レジスタアクセス、ベクタキャッチ、無制限のソフトウェア ブレークポイント、フルシステム メモリアクセスを含みます。プロセッサは、実装中に設定された 4/8 ハードウェア ブレークポイントと 2/4 ウォッチポイントもサポートします。

4.5 AVR デバイスの内蔵デバッグ機能

内蔵デバッグ(OCD)モジュールは、一般的に**デバッガ**または**デバッグアダプタ**として知られる装置を介して外部の開発プラットフォームからデバイス上のコード実行を監視および制御する事を可能にするシステムです。

OCD システムにより、ターゲットシステム内で正確な電気的特性とタイミング特性を維持してアプリケーションを実行しながら、条件付きまたは手動で実行を停止してプログラムフローとメモリを調査できます。

Run モード

Run モード中にコードは MPLAB PICkit Basic から完全に独立して実行されます。MPLAB PICkit Basic は、ブレーク条件の発生を検出するためにターゲット デバイスを継続的に監視します。ブレーク条件が発生すると、ユーザがデバイスの内部ステートを調べる事ができるよう、OCD システムは自身のデバッグ インターフェイスを介してデバイスに問い合わせます。

Stopped モード

ブレークポイントに達した時点でプログラム実行は停止しますが、一部の I/O はブレークポイントが発生しなかったかのように動作を継続できます。例えば、USART 送信が開始された直後にブレークポイントに達した場合、コアが Stopped モード中であっても USART はフルスピードで動作を継続して送信を完了します。

ハードウェア ブレークポイント

ターゲット OCD モジュールは、ハードウェア内に複数のプログラム カウンタ コンパレータを実装しています。プログラム カウンタがいずれか 1 つのコンパレータ レジスタ内の値と一致した時に OCD は Stopped モードへ移行します。ハードウェア ブレークポイントは OCD モジュール上で専用ハードウェアを要求するため、利用可能なブレークポイントの数はターゲットに実装されている OCD モジュールのサイズによって決まります。通常、そのような 1 つのハードウェア コンパレータが内部使用向けにデバッガによって「予約」されます。

ソフトウェア ブレークポイント

ソフトウェア ブレークポイントは、ターゲット デバイス上のプログラムメモリ内に配置された BREAK 命令です。この命令がロードされた時にプログラム実行がブレークし、OCD は Stopped モードへ移行します。実行を再開するには、OCD から「start」コマンドを発行する必要があります。一部の Microchip 社製デバイスは BREAK 命令をサポートする OCD モジュールを備えています。

4.5.1 AVR デバイス インターフェイス

Note: UPDI/PDI/TPI インターフェイスを使用する AVR マイクロコントローラ デバイスでプログラミング/デバッグに問題が生じる場合、使用ツールに対応する Engineering Technical Notes (ETN) を参照してください。

AVR デバイスは各種のプログラミング/デバッグ インターフェイスを備えています。各デバイスがサポートするインターフェイスについては、デバイス データシートを参照してください。

- 全ての AVR E/D デバイスと一部の新しい tinyAVR および megaAVR デバイスは、プログラミング/デバッグ用に UPDI インターフェイスを備えています。
- 一部の tinyAVR®デバイスは TPI インターフェイスを備えています。TPI は、デバイスのプログラミング用にのみ使えます。これらのデバイスは内蔵デバッグ機能を全く備えていません。
- 一部の tinyAVR デバイスと一部の megaAVR デバイスは、tinyOCD として知られる内蔵デバッグシステムへ接続する debugWIRE インターフェイスを備えています。debugWIRE を備える全てのデバイスは、インシステム プログラミング用に SPI インターフェイスも備えています。
- 一部の megaAVR デバイスは、プログラミング/デバッグ用 JTAG インターフェイスと megaOCD としても知られる内蔵デバッグシステムを備えています。JTAG を備えた全てのデバイスは、インシステム プログラミング用の代替インターフェイスとして SPI インターフェイスも備えています。
- 全ての AVR XMEGA デバイスは、プログラミング/デバッグ用に PDI インターフェイスを備えています。一部の AVR XMEGA デバイスは、同じ機能を持つ JTAG インターフェイスも備えています。

表 4-3. AVR デバイスファミリが備えるプログラミング/デバッグ インターフェイス

AVR デバイス ファミリ	UPDI	TPI	SPI	debugWIRE	JTAG	PDI
AVR E/D	新デバイス					
tinyAVR	新デバイス	一部のデバイ	一部のデバイ	一部のデバイス		
megaAVR	新デバイス		一部のデバイ	一部のデバイス	一部のデバイス	
AVR XMEGA					一部のデバイス	全デバイス

4.5.1.1 UPDI OCD 機能

新しい tinyAVR および megaAVR デバイスと AVR E/D デバイスの UPDI OCD は、UPDI 物理インターフェイス(単ピンプログラミング/デバッグ インターフェイス)に基づきます。

tinyAVR および megaAVR デバイスのその他の特長:

- デバイスアドレス空間(NVM、RAM、I/O)へのメモリマップ アクセス
- デバイスクロック周波数の低下設定不要
- 限定なしのユーザ ブレークポイント数
- 2x ハードウェア ブレークポイント

- 先進 OCD 機能のサポート
- システムレジスタにアクセスしない非侵襲性(non-intrusive)の実行時デバイス監視
- ロックされたデバイス上でフラッシュの CRC 検査結果を読み出すためのインターフェイス

AVR E/D デバイスのその他の特長:

- 2x ハードウェア ブレークポイント
- フローの変更、割り込み、ソフトウェア ブレークポイント
- スタックポインタ(SP)レジスタ、プログラム カウンタ(PC)、ステータス レジスタ(SREG)の実行時読み出し
- Stopped モード中に読み書き可能なレジスタファイル

UPDI ピンが共有されるデバイスでは、そのピンを GPIO または/RESET ピンに再構成可能です。詳細は「Unified Program and Debug Interface (UPDI) High-Voltage Activation Information」を参照してください。

古いデバイスでは debugWIRE OCD が利用できます。OCD 機能の詳細は「debugWIRE OCD Features」を参照してください。

4.5.1.1.1 UPDI OCD に関する特別な配慮

UPDI データピン(UPDI_DATA)は、ターゲット AVR®デバイスに応じて専用ピンまたは共有ピンに割り当てられます。共有 UPDI ピンは、UPDI ピンまたは $\overline{\text{RESET}}$ ピン(デバイスに依存)での高電圧(HV)パルスによる有効化を必要とします。詳細は使用するデバイスのデータシートを参照してください。

CRCSCAN モジュール(CRC メモリスキャン)を備えたデバイスでは、デバッグ中にこのモジュールを連続バックグラウンド モードで使ってはいけません。OCD モジュールのハードウェア ブレークポイント コンパレータのリソースは限られているため、それより多数のブレークポイントが必要な場合は BREAK 命令(ソフトウェア ブレークポイント)をフラッシュ内に挿入できます。あるいはソースレベル コードのステップ実行中でも、これは可能です。ただし、CRC モジュールはブレークポイントをフラッシュメモリ内容の破損として誤検出する可能性があります。

CRCSCAN モジュールは、ブート前に CRC スキャンを実行するように設定されている場合があります。CRC 不一致が検出された場合、デバイスはブートせずにロック状態になる可能性があります。この状態からデバイスを回復させるには、完全なチップ消去を実行した後に有効なフラッシュイメージを書き込むか、ブート前 CRCSCAN を無効にする以外に方法はありません(チップ消去だけではフラッシュがブランク状態(CRC は不正)となるため、デバイスはブートしません)。チップ消去によってデバイスがこの状態になると、ソフトウェア フロントエンドは自動的に CRCSCAN ヒューズを無効にします。

UPDI インターフェイスを使用するターゲット アプリケーション PCB を設計する際は、適正動作を得るために以下の配慮が必要です。

- UPDI ラインのプルアップ抵抗を 10 kΩ 以上にする。プルダウン抵抗は使わない(UPDI を使う時にプルアップ抵抗を取り外す)。UPDI 物理インターフェイスはプッシュ-プル回路形式のため、ラインのアイドル時に Start ビットが誤ってトリガされる事を防ぐために弱いプルアップ抵抗で十分です。
- UPDI ピンが $\overline{\text{RESET}}$ ピンとして使われる場合、UPDI の使用時にノイズ対策用のコンデンサを切り離す必要があります(それらの容量性負荷はインターフェイスの適正動作を妨げるため)。
- UPDI ピンが $\overline{\text{RESET}}$ または GPIO ピンとして使われる場合、プログラミング/デバッグ中にライン上の全ての外部ドライバを切り離す必要があります(それらのドライバはインターフェイスの適正動作を妨げる可能性があるため)。

4.5.1.2 debugWIRE OCD の特長

debugWIRE OCD は少ピン数 AVR デバイス専用に設計された特別な OCD モジュールであり、機能は制限されています。このモジュールは以下の機能をサポートします。

- 完全なプログラムフロー制御
- 全レジスタおよびメモリ領域へのフルアクセス

- 無制限のユーザプログラム ブレークポイント(BREAK 命令を使用)
- ターゲット クロックに基づく自動 baud レート設定

4.5.1.2.1 debugWIRE に関する特別な配慮

debugWIRE 通信ピン(dW)は、外部リセット(RESET)と同じピンに物理的に配置されています。従って、debugWIRE インターフェイスの有効時に外部リセット要因はサポートされません。

debugWIRE インターフェイスが機能するには、ターゲット デバイス上で debugWIRE イネーブル(DWEN)ヒューズがセットされている必要があります。Microchip AVR デバイスは、このヒューズが未設定(既定値)の状態で出荷されます。debugWIRE インターフェイスを使ってこのヒューズをセットする事はできません。DWEN ヒューズをセットするには、SPI モードを使う必要があります。必要な SPI ピンが接続されている場合、ソフトウェア フロントエンドはこれを自動的に処理します。ソフトウェア フロントエンド内で SPI プログラミングを使って手動でセットする事もできます。

自動設定: debugWIRE を備えたデバイス上でデバッグ セッションの開始を試みます。debugWIRE インターフェイスが有効ではない場合、ソフトウェア フロントエンドは SPI プログラミングを使って debugWIRE の有効化を試みます。SPI ヘッダの全てのピンが接続されていれば debugWIRE が有効になり、ユーザはターゲットの電源を再投入するよう指示されます。電源の再投入は、変更されたヒューズの効果を得るために必要です。

手動設定: SPI モード中に Microchip Studio 内のプログラミング ダイアログを開き、シグネチャがデバイスと一致している事を確認します。DWEN ヒューズにチェックマークを付ける事により debugWIRE を有効にします。

重要:
SPIEN ヒューズは設定したまま、RSTDISBL ヒューズを未設定にする事が重要です。そうしないと、debugWIRE モード中にデバイスがスタックした時に、DWEN 設定を復元するために高電圧プログラミングが必要になります。

debugWIRE インターフェイスを無効にするには、高電圧プログラミングを使って DWEN ヒューズを未設定にします。あるいは、debugWIRE インターフェイスを使って一時的に debugWIRE インターフェイス自身を無効にする事で、SPI プログラミングの実行を可能にします(SPIEN ヒューズがセットされている場合)。

重要:
SPIEN ヒューズの設定を忘れた場合、SPI からのアクセスができず、操作が完了できません。結果として高電圧プログラミングが必要となります。

MPLAB X IDE の場合、ターゲット デバイス上で debugWIRE が有効にされている状態で SPI プログラミング セッションが試みられると、IDE は最初に debugWIRE の無効化を試みます。Microchip Studio の場合、手動でこれを行う必要があります(デバッグ セッション中に「Debug」メニューからオプション「Disable debugWIRE and Close」を選択)。debugWIRE は一時的に無効になり、ソフトウェア フロントエンドは SPI プログラミングを使って DWE ヒューズを未設定にします。

DWEN ヒューズが設定されていると、クロックシステムの一部は全てのスリープモード中に動作可能となります。これによりスリープモード中の AVR の消費電力は増加します。従って、debugWIRE を使用しない時は常に DWEN ヒューズを無効にしておくべきです。

debugWIRE を使用するターゲット アプリケーション PCB を設計する際は、適正動作を得るために以下の配慮が必要です。

- dW/(RESET)ラインのプルアップ抵抗を 10 kΩ 以上にする。ただし、デバッガツールがプルアップ抵抗を提供するため、debugWIRE 機能のためのプルアップ抵抗は不要です。
- debugWIRE の使用時は、RESET ピンへ接続しているノイズ対策用のコンデンサを切り離す必要があります(それらの容量性負荷はインターフェイスの適正動作を妨げるため)。

- RESET ライン上の全ての外部リセット要因またはその他のアクティブ ドライバは切り離す必要があります(それらはインターフェイスの適正動作を妨げる可能性があるため)。

ターゲットデバイス上のロックビットを設定してはいけません。debugWIRE インターフェイスが正しく機能するには、ロックビットがクリアされている必要があります。

4.5.1.2.2 debugWIRE のソフトウェア ブレークポイント

debugWIRE OCD は、Microchip 社の megaAVR (JTAG) OCD に比べて大幅に縮小されており、ユーザがデバッグ用に使えるプログラム カウンタ ブレークポイントを一切備えていません。そのようなコンパレータはカーソル位置までの実行(run-to-cursor)とシングルステップ動作向けに 1 つだけ存在しますが、追加のユーザ ブレークポイントはハードウェアでサポートされません。

その代わりにデバッガは AVR BREAK 命令を使う必要があります。この命令はフラッシュ内に配置できます。実行時にこの命令がロードされると AVR CPU はスリープモードへ移行します。デバッグ中のブレークポイントをサポートするため、デバッガはユーザがブレークポイントを要求するフラッシュ内の位置に BREAK 命令を挿入する必要があります。オリジナルの命令は、後で置き換えるためにキャッシュされる必要があります。BREAK 命令をまたぐシングルステップ実行時にプログラムの挙動を保つため、デバッガはキャッシュされたオリジナルの命令を実行する必要があります。極端なケースでは、BREAK をフラッシュから削除した後に置き換える必要があります。これらの全ての処理により、ブレークポイントからのシングルステップ実行時に明らかな遅延が生じる可能性があります。ターゲット クロック周波数が非常に低い場合、この状況はさらに悪化します。

従って、可能であれば以下のガイドラインに従う事を推奨します。

- デバッグ中は常にターゲットをできるだけ高い周波数で動作させる(debugWIRE 物理インターフェイスはターゲットからのクロックで動作する)。
- ブレークポイントの追加/削除の数を最小化する(そのたびにターゲット上のフラッシュページの書き換えが要求されるため)。
- フラッシュページの書き込み回数を最小化するため、一度に追加または削除するブレークポイントを少数に抑える。
- 可能であれば、2 ワード命令にブレークポイントを置かない。

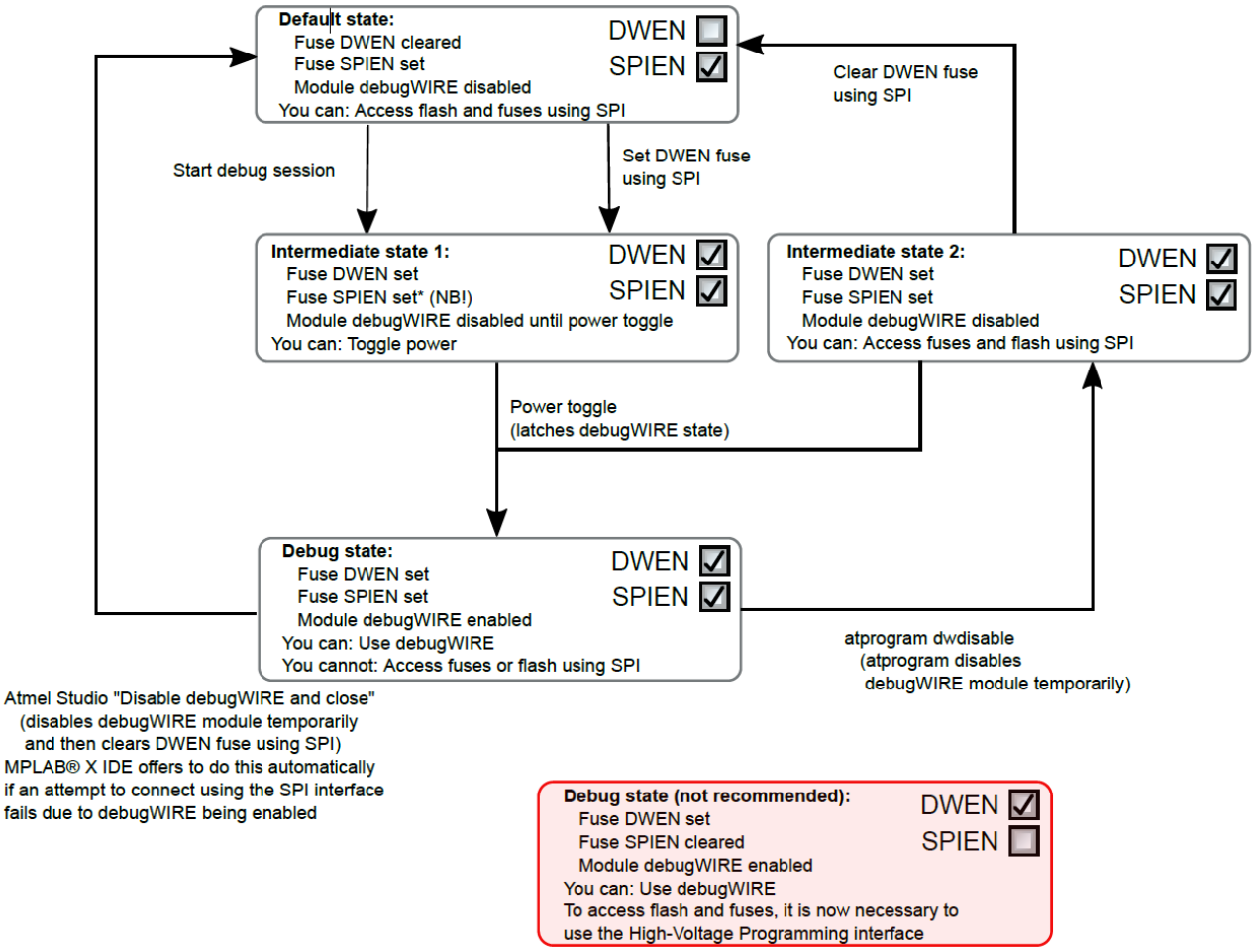
4.5.1.2.3 debugWIRE と DWEN ヒューズを理解する

有効にされた debugWIRE インターフェイスはデバイスの/RESET ピンを制御します。SPI インターフェイスもこのピンを必要とするため、debugWIRE インターフェイスと SPI インターフェイスは相互排他的です(同時に使えません)。debugWIRE モジュールの有効化/無効化は以下の 2 通りの方法で行えます。

- ソフトウェア フロントエンドに処理をまかせる(推奨)
- DWEN を手動でセットまたはクリアする(注意を要するため熟練ユーザ向け)

重要: DWEN を手動で扱う場合、高電圧プログラミングが必要になる事態を避けるため、SPIEN ヒューズはセットしたままにする必要があります。

図 4-1. debugWIRE と DWEN ヒューズの関係



4.5.1.3 megaAVR OCD の特長

megaAVR OCD は JTAG 物理インターフェイスに基づきます。このモジュールは以下の機能をサポートします。

- 完全なプログラムフロー制御
- 全レジスタおよびメモリ領域へのフルアクセス
- 4x プログラムメモリ (ハードウェア) ブレークポイント(1 つは予約済み)
- ハードウェア ブレークポイントを結合してデータ ブレークポイントを形成可能
- BREAK 命令による無制限のプログラム ブレークポイント(ATmega128[A]を除く)

4.5.1.4 AVR XMEGA OCD の特長

AVR XMEGA OCD は別名 PDI (Program and Debug Interface)としても知られます。2 つの物理インターフェイス(JTAG と PDI)はデバイス内の同一 OCD 実装へのアクセスを提供します。このモジュールは以下の機能をサポートします。

- 完全なプログラムフロー制御
- 全レジスタおよびメモリ領域へのフルアクセス
- 1x 専用プログラム アドレス コンパレータまたはシンボリック ブレークポイント(予約済み)
- 4x ハードウェア コンパレータ
- BREAK 命令による無制限のユーザプログラム ブレークポイント
- 無制限のシステムクロック周波数

4.5.1.4.1 AVR® XMEGA®に関する特別な配慮

OCD とクロック

MCU が Stopped モードへ移行すると、OCD クロックが MCU クロックとして使われます。JTAG インターフェイス使用時の OCD クロックは JTAG TCK であり、PDI インターフェイス使用時の OCD クロックは PDI_CLK です。

Stopped モード中の I/O モジュール

以前の Microchip megaAVR デバイスとは異なり、XMEGA では Stopped モード中に I/O モジュールが停止します。このため、USART 送信は中断され、タイマ(および PWM)は停止します。

ハードウェア ブレークポイント

4 個のハードウェア コンパレータ(2x アドレス コンパレータと 2x 値コンパレータ)が存在します。これらには以下の制約があります。

- 全てのブレークポイントは同一タイプ(プログラムまたはデータ)である事
- 全てのデータ ブレークポイントは同一メモリ領域(I/O、SRAM、XRAM)内に配置される事
- アドレスレンジを使う場合、ブレークポイントは 1 つだけ配置可能

設定可能なブレークポイントの組み合わせは以下の通りです。

- 2x データまたはプログラム アドレス ブレークポイント(単一のアドレスを指定)
- 1x データまたはプログラム アドレスレンジ ブレークポイント(アドレスの範囲を指定)
- 2x データ アドレス/値 ブレークポイント(単一のアドレスと値を指定)
- 1x データ ブレークポイント (アドレスまたは値もしくはその両方の範囲を指定)

MPLAB X IDE と Microchip Studio は、ブレークポイントが設定できなかった場合にその理由をユーザに示します。ソフトウェア ブレークポイントが利用可能である場合、データ ブレークポイントはプログラム ブレークポイントよりも優先されます。

外部リセットと PDI 物理インターフェイス

PDI 物理インターフェイスは、リセットラインをクロックとして使います。デバッグ時は、10 kΩ 以上のリセット プルアップ抵抗を使うか、このプルアップ抵抗を取り除く必要があります。リセット コンデンサは全て取り除く必要があります。その他の外部リセット要因は切り離す必要があります。

JTAGEN ヒューズ

JTAG インターフェイスは、JTAGEN ヒューズを使って有効にします。このヒューズは既定値では設定状態にあり、JTAG プログラミング インターフェイスへのアクセスが可能となっています。

重要: JTAGEN ヒューズが誤って無効にされた場合、再度有効にするには PDI 物理インターフェイスを使う以外に方法はありません。

JTAGEN ヒューズが設定されていても、ファームウェア内で MCU コントローラ レジスタ内の JTAG ディセーブル ビットをセットする事により、JTAG インターフェイスを無効にできます。この操作によってコードはデバッグ不可能となるため、デバッグ セッションを試みる時にこれを行ってはいけません。デバッグ セッションの開始時に Microchip AVR デバイス上でそのようなコードが既に実行されている場合、MPLAB PICkit Basic は接続中に $\overline{\text{RESET}}$ ラインをアサートします。$\overline{\text{RESET}}$ ラインが正しく配線されていればターゲット デバイスは強制的にリセットされ、JTAG 接続が可能になります。

JTAG インターフェイスが有効である場合、JTAG ピンを代替ピン機能向けに使う事はできません。プログラムコードから JTAG ディセーブル ビットをセットするか、プログラミング インターフェイスを介して JTAGEN ヒューズをクリアする事によって JGAG インターフェイスを無効にしない限り、それらのピンは専用 JTAG ピンのままです。

ヒント:
デバイス上で JTAG ディセーブル ビットをセットする(従って JTAG インターフェイスを無効にする)コードが実行された場合に MPLAB PICkit Basic から RESET ラインをアサートする事によって JTAG インターフェイスを再度有効にできるようにするには、Microchip Studio 内の**[Programming]**ダイアログと**[Debug options]**ダイアログの両方で**[Use external reset]**チェックボックスにチェックを入れておく必要があります。

ATxmegaA1 (rev H 以前)におけるスリープ中のデバッグに関するバグ

ATxmegaA1 デバイスの初期バージョンでは、デバイスが特定スリープモード中である時に OCD の有効化が妨げられるというバグが存在しました。OCD を再有効化するための対処方法には以下の 2 通りがあります。

- MPLAB PICkit Basic を選択し、「Tool」メニュー内のオプション「Always activate external Reset when reprogramming device」を有効にする。
- チップ消去を実行する。

このバグは以下のスリープモードで発生します。

- パワーダウン
- パワーセーブ
- スタンバイ
- 拡張スタンバイ

4.5.1.5 先進デバッグ(AVR® JTAG/debugWIRE デバイス)における I/O 周辺モジュール

ブレークポイントによってプログラム実行が停止しても、大部分の I/O 周辺モジュールは動作を継続します。例えば、UART 送信中にブレークポイントに達した場合、送信は完全に実行されて対応するビットがセットされます。コードの次のシングルステップで TXC (送信完了)フラグがセットされ利用可能になります(実際のデバイスでは、通常これは遅れて発生します)。

以下の 2 つのモジュールを除く全ての I/O モジュールは、Stopped モード中も動作を継続します。

- タイマ/カウンタ(ソフトウェア フロントエンドを使って設定可能)
- ウォッチドッグ タイマ(デバッグ中のリセットを防ぐために常に停止)

シングルステップ中の I/O アクセス

Stopped モード中に I/O は動作を継続するため、特定のタイミング問題を防ぐための注意が必要です。例:

```
OUT PORTB, 0xAA
IN TEMP, PINB
```

このコードを普通に実行した場合、TEMP レジスタは `0xAA` を読み戻しません。パイプライン処理の都合で、IN 動作によってサンプリングされた時にデータはまだピンへ物理的にラッチされていないからです。PIN レジスタに正しい値が格納されるようにするには、`OUT` 命令と `IN` 命令の間に `NOP` 命令を置く必要があります。

しかし、OCD を通してこの関数をシングルステップ実行した場合、シングルステップ中にコアが停止しても I/O はフルスピードで動作するため、このコードは常に PIN レジスタに値 `0xAA` を与えます。

シングルステップとタイミング

特定のレジスタは、制御信号を有効にした後、決められたサイクル数以内に読み書きする必要があります。I/O クロックと周辺モジュールは Stopped モード中もフルスピードで動作し続けるため、そのようなコードのシングルステップ実行はタイミング要件を満たしません。2 つのシングルステップの間に、I/O クロックサイクルは数 100 万を数えるかもしれません。そのようなタイミング要件でレジスタを正しく読み書きするには、デバイスをフルスピードで動作させて読み書きシーケンス全体を 1 つのアトミック動作で実行する必要があります。

これは、マクロまたは関数呼び出しを使ってコードを実行するか、デバッグ環境内で run-to-cursor 機能を使う事により実現します。

16 ビットレジスタへのアクセス

Microchip AVR の周辺モジュールは通常複数の 16 ビットレジスタ(例: 16 ビットタイマの TCNTn)を備えており、それらには 8 ビット データバスを介してアクセスします。16 ビットレジスタには、2 回の読み出しまたは書き込み動作を使ってバイト単位でアクセスする必要があります。16 ビットアクセスの中間でのブレークまたは 16 ビット アクセスのシングルステップ実行では正しい値が得られない可能性があります。

I/O レジスタへのアクセスに関する制約

特定のレジスタでは、その値に影響を与えずに読み出す事はできません。そのようなレジスタには、読み出し時にクリアされるフラグを含むレジスタや、バッファリングされるデータレジスタ(例: UDR)が含まれます。ソフトウェア フロントエンドは、OCD デバッグの意図する非侵襲性を維持するために、Stopped モード中にこれらのレジスタが読み出される事を防ぎます。また、副作用を生じる事なく安全に書き込む事ができないレジスタも存在します。これらのレジスタは読み出し専用です。

例:

- フラグレジスタの場合、任意ビットに「1」を書き込む事によりフラグがクリアされます。これらのレジスタは読み出し専用です。
- UDR レジスタと SPDR レジスタは、モジュールの状態に影響を与えずに読み出す事はできません。これらのレジスタはアクセス不可です。

4.6 PIC32M MCU の内蔵デバッグ機能

PIC32M MCU デバイスは、以下の 2 種類のデバッグをサポートします。

(1) PGECx および PGEDx ピンを使うインサーキット シリアル プログラミング™ (ICSP™)およびデバッグ

(2) 4 線式 MIPS®拡張 JTAG

MIPS32 M4K プロセッサコアは、アプリケーションおよびカーネルコードのソフトウェア デバッグ向けに拡張 JTAG (EJTAG)インターフェイスを提供します。EJTAG 仕様では、標準 JTAG 命令に加えて、どのレジスタをどのように使うか指定する特殊な命令が定義されています。このインターフェイスの詳細については、使用するデバイスのデータシートを参照してください。

加えて、プログラム ブレークポイントと複合データ ブレークポイントが存在します。PIC32M デバイスのデバッグ機能の詳細は、対応するデバイスデータシートを参照してください。

4.7 PIC MCU/dsPIC DSC の内蔵デバッグ機能

内蔵デバッグ(OCD)モジュールは、外部の開発プラットフォームから一般的に**デバッガ**または**デバッグアダプタ**として知られるツールを介してデバイス上の実行を監視および制御可能にするシステムです。OCD システムを使うと、ターゲット システム内で正確な電気的特性とタイミング特性を維持してアプリケーションを実行できます。システムは条件付きまたは手動で実行を停止してプログラムフローとメモリを調査できます。

PIC®マイクロコントローラ (MCU)または dsPIC®デジタルシグナル コントローラ(DSC)の場合、一部のデバイスリソースをデバッグ用に予約する事が必要になる場合があります。

4.7.1 エミュレータの基本機能

MPLAB PICkit Basic インサーキット デバッガは、以下の基本的デバッグ機能を備えています。

4.7.1.1 エミュレーションの開始/停止

MPLAB X IDE 内でアプリケーションをデバッグするには、ソースコードを含むプロジェクトを作成する必要があります。これにより、ソースコードをビルドしてターゲット デバイスにプログラミングし、以下の操作が可能になります。

	デバッグモードでプロジェクトコードをデバッグまたは実行します。
	コード実行を一時停止(Pause)または停止(Halt)します。

	一時停止または停止後にコード実行を再開します。
	一時停止または停止したコードに対して１命令をステップインまたは実行します。スリープ命令をステップインしないよう注意が必要です。さもないとエミュレーションを再開するためにプロセッサ リセットが必要になります。
	一時停止または停止したコードに対して１命令をステップオーバーします。
	デバッグ セッションを終了します(コード実行は終了)。
	プロセッサ リセットを実行します。デバイスによってはその他のリセット(POR/BOR、MCLR、システムリセット等)が利用できる場合があります。

4.7.1.2 プロセッサ メモリおよびファイルの表示

MPLAB X IDE はデバッグ情報や各種プロセッサメモリ情報を表示する複数のウィンドウを備えています。これらのウィンドウは「Window」メニューから選択できます。これらのウィンドウの使用方法については MPLAB X IDE オンラインヘルプを参照してください。

- **Window** > **Target Memory Views** – 各種デバイスメモリを表示します。選択したデバイスに応じて各種のメモリタイプ (プログラムメモリ、ファイルレジスタ、コンフィグレーション メモリ等)が表示されます。
- **Window** > **Debugging** – デバッグ情報を表示します(変数、ウォッチ、コールスタック、ブレークポイント、ストップウォッチ、トレースから選択)。

ソースコードを表示するには、**[Projects]**ウィンドウ内でいずれかのソースコード ファイルをダブルクリックします。ソースコードは**[Files]**ウィンドウに表示されます。このウィンドウには、選択したプロセッサとビルドツールに応じて、色分けされたコードが表示されます。色分けのスタイルを変更するには、**Tools** > **Options** > **Fonts & Colors** > **Syntax** を選択します。

エディタの詳細は MPLAB X IDE オンラインヘルプの「Editor」セクションを参照してください。

4.7.1.3 ブレークポイントの使用法

ブレークポイントは、コード内の指定した行でコード実行を停止するために使います。

4.7.1.3.1 ブレークポイントの数

PIC 16 ビットデバイスと dsPIC DSC デバイスの場合、ブレークポイント、データキャプチャ、ランタイムウォッチは同じリソースを使います。従って、実際に使えるブレークポイントの数は、ブレークポイントとトリガの合計数によって決まります。

PIC32M 32 ビットデバイスの場合、ブレークポイントはデータキャプチャ、ランタイム ウォッチとは別のリソースを使います。従って、使用可能なブレークポイントの数はトリガの数とは無関係です。

使用可能または使用中のハードウェア/ソフトウェア ブレークポイントの数は、**[Dashboard]**ウィンドウ(**Window** > **Dashboard**)に表示されます。詳細は MPLAB X IDE ユーザガイドを参照してください。一部のデバイスではソフトウェア ブレークポイントが使えません。

デバイスのハードウェア ブレークポイントの数、ハードウェア ブレークポイントのスキッド等、ブレークポイント動作に関する制約は、「Debug Limitations - PIC® MCUs」を参照してください。

4.7.1.3.2 ハードウェア/ソフトウェア ブレークポイントの選択

ハードウェアまたはソフトウェア ブレークポイントを選択する手順は以下の通りです。

1. **[Projects]**ウィンドウ内で、使用するプロジェクトを選択してから右クリックして**「Properties」**を選択します。
2. **[Project Properties]**内で、**ICE4** > **Debug Options** を選択します。
3. ソフトウェア ブレークポイントを使う場合、**[Use software breakpoints]**にチェックを入れます。ハードウェア ブレークポイントを使う場合、このチェックを外します。

Note: デバッグにソフトウェア ブレークポイントを使うと、デバイスの書き込み耐性に影響します。従って、このデバッグ方法を使ったデバイスは量産製品に使わない事を推奨します。

表 4-4 に、ハードウェア ブレークポイントとソフトウェア ブレークポイントの比較を示します。どちらのブレークポイントを使うべきか判断する際の参考にしてください。

表 4-4. ハードウェア ブレークポイントとソフトウェア ブレークポイント

特長	ハードウェア ブレークポイント	ソフトウェア ブレークポイント
ブレークポイントの数	制限あり	制限なし
ブレークポイントの書き込み先*	内部デバッグレジスタ	フラッシュ プログラムメモリ
ブレークポイントの適用先**	プログラムメモリとデータメモリ	プログラムメモリのみ
ブレークポイントの設定に要する時間	最小限	オシレータ速度、フラッシュメモリへの書き込み時間、ページサイズによって決まる
ブレークポイントのスキッド	大部分のデバイスで発生 詳細はオンラインヘルプの「Limitations」セクション参照	発生しない
* ブレークポイントに関する情報が書き込まれるデバイス内の場所		
** ブレークポイントが適用されるデバイス機能の種類(ブレークポイントが設定される場所)		

4.7.1.4 ストップウォッチの使用

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

Note: ストップウォッチはブレークポイントのリソースを使います。

ストップウォッチの使い方

1. ストップウォッチを開始するブレークポイントを追加します。
2. ストップウォッチを停止するブレークポイントを追加します。
3. **Window** > **Debugging** > **Stopwatch** を選択します。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. プログラムを再度デバッグすると、ストップウォッチの結果が表示されます。

図 4-2. ストップウォッチの設定

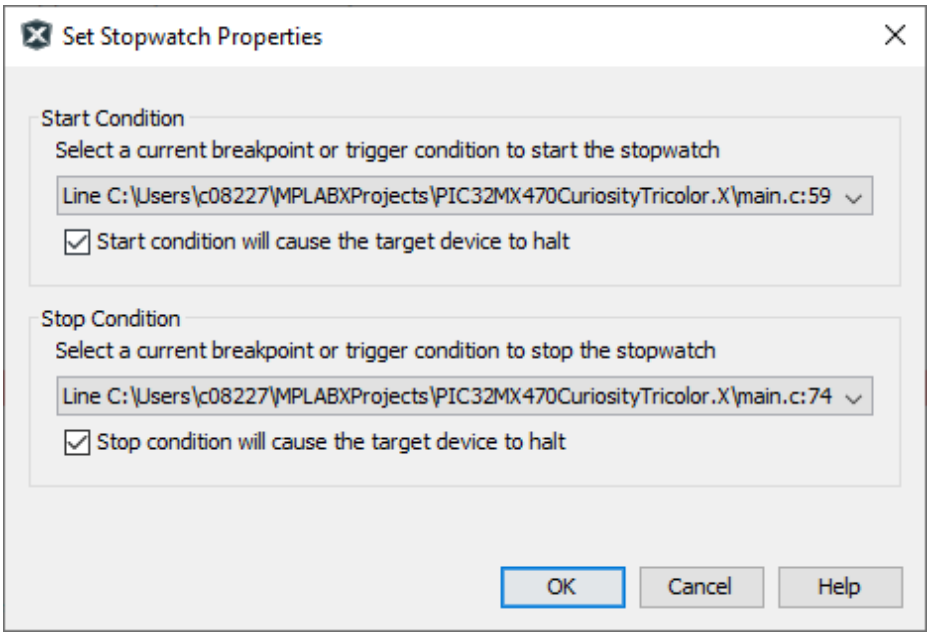
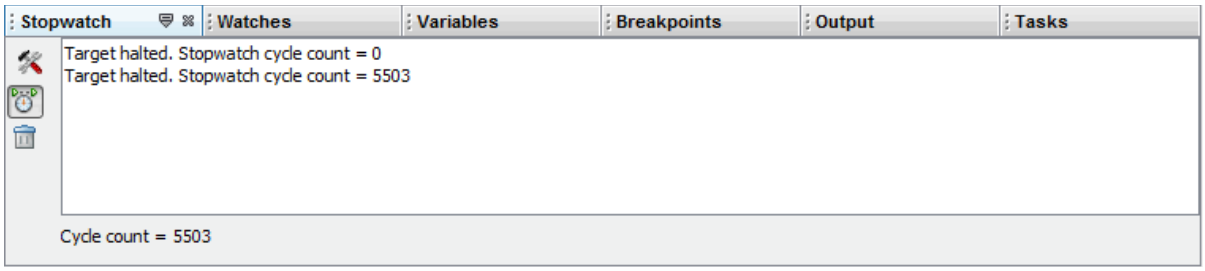


図 4-3 [Stopwatch]ウィンドウの内容



[Stopwatch]ウィンドウの左側に以下のアイコンが表示されます。

表 4-5. ストップウォッチ アイコン

	アイコンテキスト	概要
	Properties	ストップウォッチのプロパティを設定します。現在のブレークポイントまたはトリガの 1 つを始点、もう 1 つを終点として選択します。
	Reset Stopwatch on Run	実行開始時にストップウォッチの時間を 0 にリセットします。
	Clear History	**[Stopwatch]**ウィンドウをクリアします。
	Clear Stopwatch	(シミュレータのみ) デバイスのリセット後にストップウォッチをリセットします。

4.7.1.5 周辺モジュールのフリーズ設定

「Freeze on Halt」を選択すると、実行停止時にその周辺モジュールをフリーズさせる事ができます。これらの機能の詳細は「周辺モジュールのフリーズ」を参照してください。

4.7.2 ICSP デバッグ

MPLAB® PICkit™ Basic インサーキット デバッガは、2 段階の手順によりデバッガとして使う事ができます。最初にアプリケーションをターゲット デバイスにプログラミングします(MPLAB PICkit Basic はプログラミング用に使えます)。次にターゲットのフラッシュ デバイスに内蔵されたインサーキット デバッグ ハードウェアを使って、アプリケーション プログラムを実行/テストします。これらの手順は、MPLAB X IDE 内での以下の操作に直接対応します。

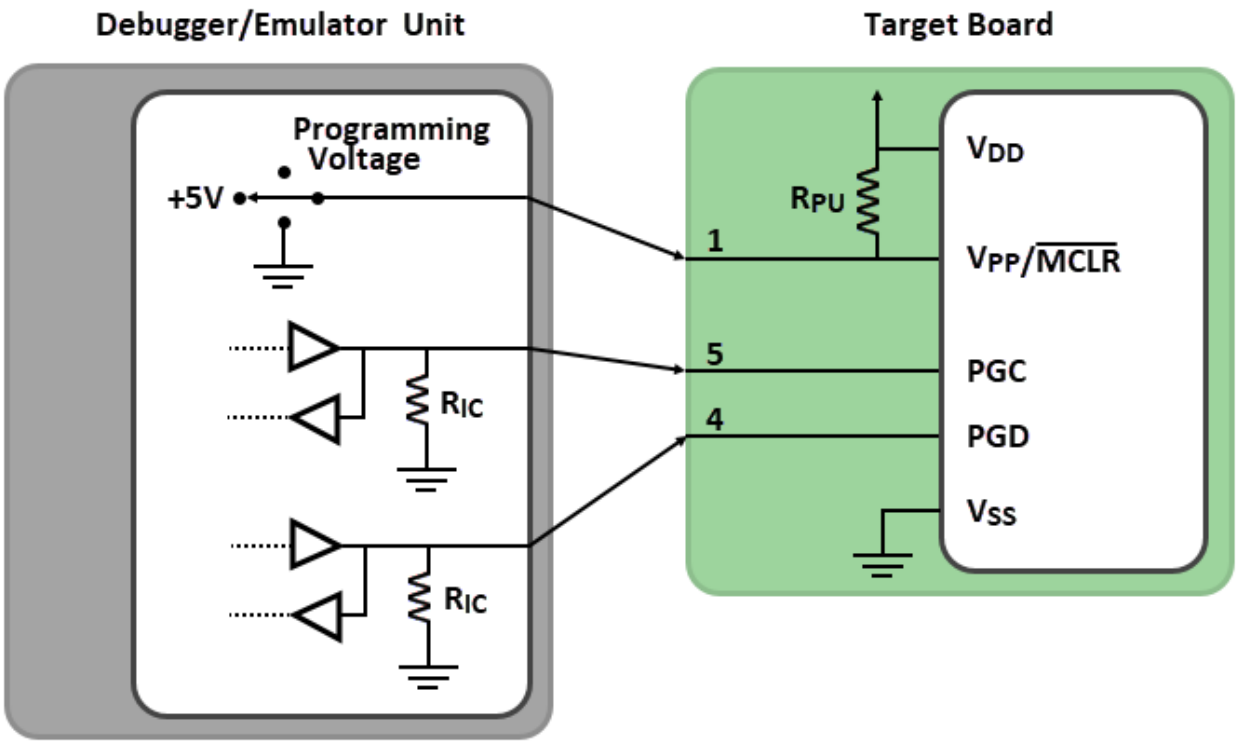
1. コードをターゲットにプログラミングし、特別なデバッグ機能を有効にします(詳細は後述)。
2. デバッガを使ってブレークポイントを設定し、コードを実行します。

詳細は MPLAB X IDE オンラインヘルプを参照してください。

ターゲット デバイスが正しくプログラミングできない場合、MPLAB PICkit Basic によるデバッグはできません。

図 4-4 に、MPLAB PICkit Basic の内部インターフェイス回路の概略図を示します。

図 4-4. ICSP プログラミング向けの接続



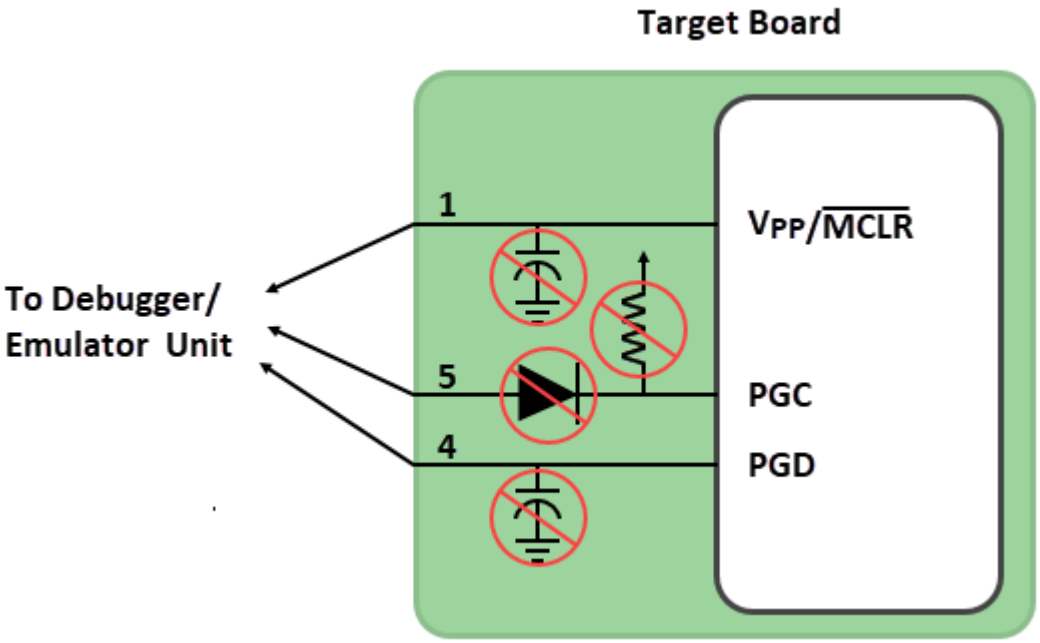
Ric=4.7 kΩ and Rpu=10 kΩ typical

プログラミングの場合、ターゲット デバイスにクロックを供給する必要はありませんが、電源は供給する必要があります。プログラミング時にデバッガは VPP/MCLR ラインをプログラミング レベルに設定し、PGC にクロックパルス、PGD にシリアルデータを出力します。デバイスが正しくプログラミングされた事を確認するため、デバッガは PGC へクロックを供給し、PGD からデータを読み戻します。このシーケンスにより、デバッガとデバイスが正しく通信している事を確認します。

4.7.2.1 エミュレータの動作を妨げる回路

図 4-5 は、デバッグ用ラインに MPLAB PICkit Basic インサーキット デバッガの正常動作を妨げる部品が取り付けられた状態を示しています。

図 4-5. 不適切な回路部品



特に以下のガイドラインに従う必要があります。

- **PGC/PGD にプルアップ抵抗を使わない** – 電圧レベルが低下します。
- **PGC/PGD にコンデンサを使わない** – プログラミング/デバッグ中のデータラインとクロックラインの高速な遷移が妨げられ、プログラミング時間が長くなります。
- **MCLR にコンデンサを使わない** – VPP の高速な遷移が妨げられます。通常は単純なプルアップ抵抗で十分です。
- **PGC/PGD にダイオードを使わない** – デバッガとターゲット デバイスの間の双方向通信が妨げられます。

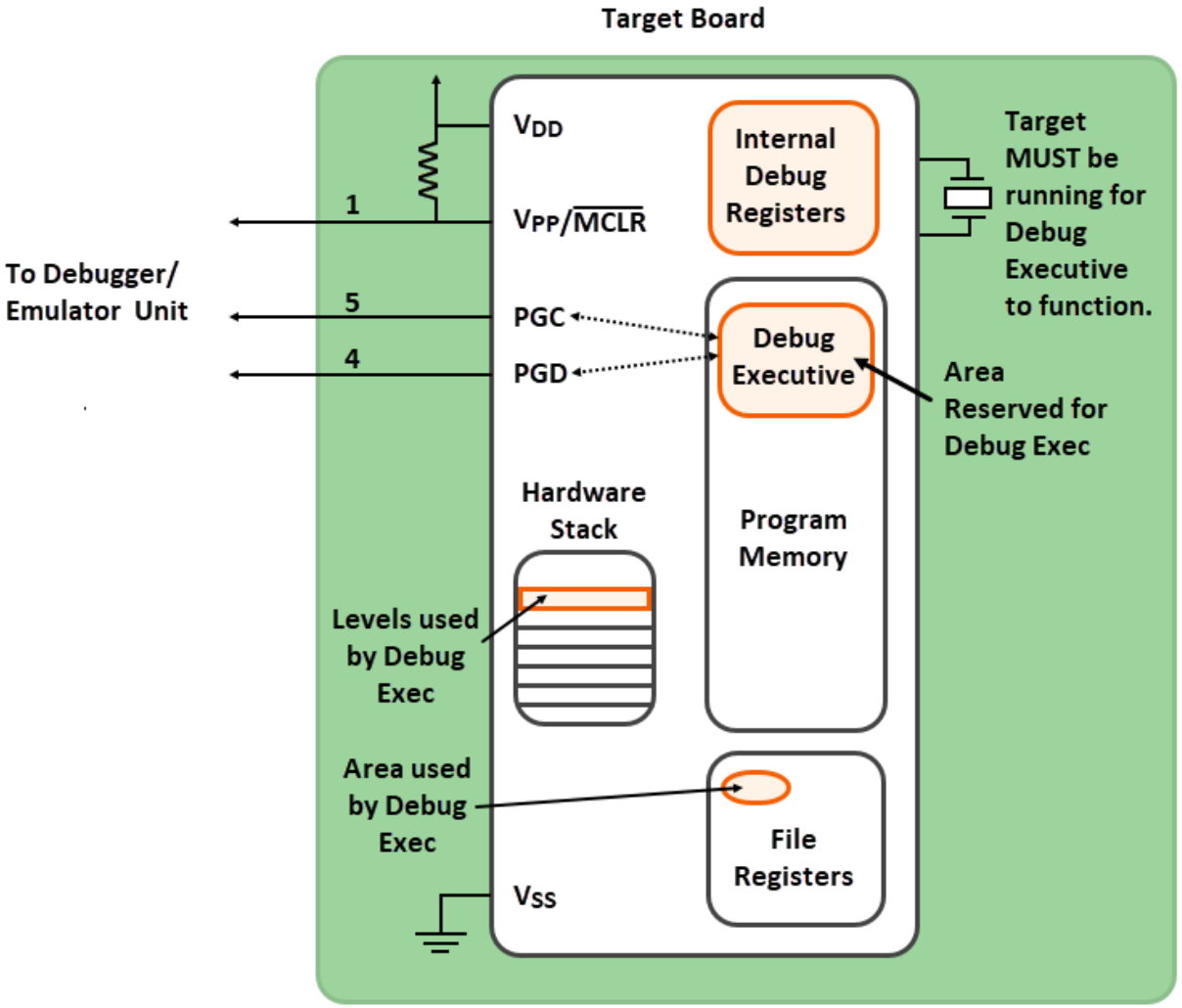
4.7.2.2 デバッグ開始前の準備手順

「4.7.2.4. デバッグに関する要件」を満たしていれば、MPLAB X IDE 内で MPLAB PICkit Basic インサーキット デバッガをツールとして設定します。使用するプロジェクトの名前を右クリックし、**「Properties」**を選択して**[Project Properties]**ダイアログを開き、**「Connected Hardware Tool」**の下で**「MPLAB PICkit Basic」**を選択します。複数のツールを接続している場合、ツール名の横のシリアル番号で識別して選択します。

この時点で、以下の操作が可能になります。

- **Debug** > **Debug Project** を選択すると、前述の ICSP プロトコルに従ってアプリケーション コードがデバイスのメモリに書き込まれます。
- ターゲット デバイスのプログラムメモリの上位領域には、小さな「デバッグ実行プログラム」コードが書き込まれます。一部のアーキテクチャでは、デバッグ実行プログラムはプログラムメモリに常駐する必要があるため、アプリケーション コードからこの予約済み空間を使ってはいけません。デバイスによっては、デバッグ実行プログラム専用の特別なメモリ空間を備えています。詳細はデバイスのデータシートで確認してください。
- MPLAB X IDE は、ターゲット デバイスの「インサーキット デバッグ」専用レジスタを有効にします。これにより、デバッガからデバッグ実行プログラムを有効にする事ができます。デバイスの予約済みリソースについては「4.7.2.5. デバッガが使うリソース」を参照してください。
- ターゲット デバイスはデバッグモードで動作します。ターゲットを確実に動作させるため「5.2. デバッグに失敗する主な理由」を参照してください。

図 4-6. PIC® MCU が使用するデバッグ向け予約済みリソース



4.7.2.3 デバッグの詳細

MPLAB PICkit Basic とターゲットが正しく接続され、かつ回路に問題がなければ、デバッガはデバッグを開始できます。

アプリケーション コードが正しく動作するかどうかを確認するため、通常はコードの最初の方にブレークポイントを設定します。MPLAB X IDE のユーザ インターフェイスからブレークポイントを設定すると、そのアドレスがターゲット デバイス内部のデバッグ専用レジスタに書き込まれます。ブレークポイントのアドレスは、PGC および PGD ピンを介してこれらのレジスタへ直接書き込まれます。

次に、MPLAB X IDE 内で**[Debug]**ボタン()を選択します。デバッガは、デバッグ実行プログラムの実行を指令します。ターゲットはリセットベクタから実行を開始し、プログラム カウンタが内部デバッグレジスタに書き込まれているブレークポイント アドレスに達すると停止します。

ブレークポイント アドレスの命令が実行されるとターゲット デバイスのインサーキット デバッグ メカニズムがトリガされ、デバイスのプログラム カウンタがデバッグ実行プログラムに渡されます(割り込みに似た動作)。この時点で、ユーザ アプリケーションの実行は停止します。デバッガは PGC と PGD を介してデバッグ実行プログラムと通信し、ブレークポイントのステータス情報を取得して MPLAB X IDE に返します。次に MPLAB X IDE は、ターゲット デバイスに関する情報(ファイルレジスタの内容、CPU の状態等)を取得するために、一連の問い合わせをデバッガに送信します。これらの問い合わせは、最終的にデバッグ実行プログラムによって実行されます。

デバッグ実行プログラムはプログラムメモリ内のアプリケーションと同じように動作し、一時変数を保存するためにスタックの一部を使います。何らかの理由(オシレータが動作しない、電源接続の不良、ターゲットボードの短絡等)でデバイスが動作しない場合、デバッグ実行プログラムは MPLAB PICkit Basic インサーキット デバッガと通信できません。この場合、MPLAB X IDE はエラーメッセージを表示します。

[Pause]ボタン()を使ってブレークポイントを設定する事もできます。この場合、PGC ラインと PGD ラインのトグルによってターゲット デバイスのインサーキット デバッグ メカニズムがトリガされ、プログラム カウンタはプログラムメモリ内のユーザコードからデバッグ実行プログラムへ切り換わります。この場合もターゲット アプリケーションのコード実行が停止し、MPLAB X IDE はデバッガとデバッグ実行プログラム間の通信を使ってターゲット デバイスの状態を取得します。

4.7.2.4 デバッグに関する要件

MPLAB PICkit Basic インサーキットデバッグシステムによるデバッグ (ブレークポイントの設定、レジスタの参照等)では、以下を正しく行う必要があります。

- デバッガは給電され、コンピュータへ接続され、MPLAB X IDE ソフトウェアと通信している必要があります。
- ターゲット デバイスは給電されて動作可能(オシレータが動作中)である事が必要です。いかなる場合も、ターゲット デバイスが動作しない限り MPLAB PICkit Basic インサーキット デバッガによるデバッグは行えません。
- ターゲット デバイスのコンフィグレーション ワードは、以下の通りに正しく設定しておく必要があります。これらは、コードを使って設定するか、MPLAB X IDE 内の**[Configuration Bits]**ウィンドウで設定できます。
  - オシレータ コンフィグレーション ビットはターゲット上で利用可能なオシレータ タイプに対応している必要があります。
  - 既定値でウォッチドッグ タイマが有効になるデバイスでは、ウォッチドッグ タイマを無効にする必要があります。
  - ターゲット デバイスの全てのタイプのコード保護を無効にしておく必要があります。
  - ターゲット デバイスのテーブル読み出し保護は無効にしておく必要があります。
- PGC/PGD ペアを複数個備えるデバイスでは、デバイスのコンフィグレーション ワードの設定で正しいペアを選択する必要があります。これはデバッグに対してのみ適用されます。プログラミングはどの PGC/PGD ペアでも正常に動作します。

4.7.2.5 デバッガが使うリソース

デバイスによっては、デバッグ用にデバイスのリソースを使う必要があります。それらのリソースの詳細を見るには、MPLAB X IDE 内で **Help** > **Release Notes** を選択します。「Release Notes/Readmes」のセクションに加えて「Reserved Resources」のセクションがあります。「Reserved Resources by Device Family and Tool」または「Reserved Resources by Device for All Tools」のどちらかを選択します。

4.7.2.6 プログラミング

デバッグと同様に、MPLAB X IDE 内で MPLAB PICkit Basic インサーキット デバッガをツールとして選択します。使用するプロジェクトの名前を右クリックし、**「Properties」**を選択して**[Project Properties]**ダイアログを開き、**「Connected Hardware Tool」**の下で**「PICkit Basic」**を選択します。複数のツールを接続している場合、ツール名の横のシリアル番号で識別して選択します。

- **[Run Project]**アイコン()を選択します。アプリケーション コードは、ICSP プロトコルによってデバイスのメモリに書き込まれます。プログラミング中にクロックは不要です。また、コード保護、ウォッチドッグ タイマの有効化、テーブル読み出し保護等、プロセッサの全てのモードを設定できます。
- 一部のターゲット デバイスでは、プログラムメモリの上位空間に小さな「プログラム実行処理」コードが書き込まれます。

- MPLAB X IDE はターゲット デバイスの「インサーキット デバッグ」専用レジスタと全てのデバッグ機能を無効にします。これは、ブレークポイントの設定とレジスタの内容の参照または変更ができなくなる事を意味します。
- ターゲット デバイスはリリースモードで動作します。プログラマとして使う場合、本デバッガはMCLR ラインをトグルする(ターゲット デバイスをリセット/起動する)事しかできません。

5. トラブルシュート

MPLAB® PICkit™ Basic インサーキット デバッガの動作に問題が生じた場合、最初に以下を確認します。

5.1 最初に確認する項目

1. **使用デバイスを確認します。**
   最近リリースされたデバイスを使う場合、MPLAB X IDE または MPLAB IPE を新しいバージョンに更新する事が必要になる場合があります。
2. **Microchip 社のデモボードを使っていますか。それとも自作のボードを使っていますか。また、通信接続に関する抵抗とコンデンサのガイドラインに従っていますか。**
   「Development Tools Design Advisory」を参照してください。
3. **ターゲットは給電されていますか。**
   本デバッガからターゲットに給電する事はできません。外部電源からターゲットボードに給電してください。
4. **USB ハブを使っていますか。ハブはセルフパワー型ですか。**
   問題が解決しない場合、ハブを使わずにデバッガをコンピュータに直接接続してください。
5. **デバッガに付属する USB ケーブルを使っていますか。**
   その他の USB ケーブルを使うと、品質が劣る、長過ぎる、USB 通信をサポートしていないといった問題が生じる可能性があります。

5.2 デバッグに失敗する主な理由

1. **オシレータが動作していない:** オシレータに関するコンフィグレーション ビットの設定を確認してください。外部オシレータを使っている場合、内部オシレータを試してください。内部 PLL を使っている場合、PLL の設定が正しい事を確認してください。
2. **ターゲットボードが給電されていない:** . 電源ケーブルの接続を確認してください。
3. **V_{DD} 電圧が不適切:** V_{DD} 電圧がデバイスの仕様範囲内であるか確認してください。詳細はデバイスのプログラミング仕様を参照してください。
4. **物理的に接続されていない:** デバッガとコンピュータ間またはデバッガとターゲットボード間が物理的に接続されていません。通信ケーブルの接続を確認してください。
5. **通信が遮断された:** 何らかの理由でデバッガと PC の通信が中断しました。MPLAB X IDE または MPLAB IPE でデバッガを再接続してください。
6. **デバイスが取り付けられていない:** デバイスがターゲットボードに正しく挿入されていません。デバッガが正しく接続されておりターゲットボードに電源が供給されていても、デバイスが存在しないか正常に挿入されていない場合、以下のメッセージが表示されます。
   `Target Device ID (0x0) does not match expected Device ID (0x%x)`
   (ターゲット デバイスの ID (0x0)がデバイス ID (0x%x)と一致しません)
   %x はデバイス ID です。
7. **デバイスがコード保護されている:** コード保護向けのコンフィグレーション ビット設定を確認してください。
8. **アプリケーション コードが破損している:** ターゲット アプリケーションに破損またはエラーがあります。ターゲット アプリケーションを再ビルドしてプログラミングし直してください。その後にターゲットのパワーオン リセットを実行します。
9. **プログラミング ピンの設定が不適切である:** PGC/PGD ピンペアがコンフィグレーション ビットで正しく設定されていません(複数の PGC/PGD ピンペアを備えたデバイスの場合)。
10. **追加設定が必要である:** 他の設定がデバッグと競合していないか確認してください。ターゲットのコード実行を妨げるようなコンフィグレーションが設定されている場合、デバッガによるデバッグモードでのコード実行もできません。
11. **ブラウンアウト電圧が不適切である:** ブラウンアウト検出電圧が動作電圧 V_{DD} よりも高く設定されている場合、デバイスはリセット状態になるためデバッグはできません。
12. **接続が不適切である:** 通信接続に関する 3. 「接続」内のガイドラインを参照してください。

13. **要求が無効である:**デバッガは、要求された動作を常に実行できる訳ではありません。例えばターゲット アプリケーションが実行中の場合、デバッガはブレークポイントを設定できません。

5.3 全般的な対応法

1. 一過性エラーの可能性があります。同じ操作を繰り返してみてください。
2. プログラミングに関する一般的な問題の可能性もあります。  アイコンを使って実行モードに切り換え、なるべくシンプルなアプリケーション(例: LED 点滅プログラム)でターゲットへの書き込みテストを行います。このアプリケーションが動作しない場合、ターゲットの設定に何らかの問題があると考えられます。
3. ターゲット デバイスが何らかの損傷(例: 過電流)を受けた可能性があります。開発環境では電子部品に悪影響が及ぶ事が少なくありません。ターゲットボードを交換して再試行してください。Microchip 社は、ほとんどの MCU をサポートするデモボードを提供しています。MPLAB® PICkit™ Basic インサーキット デバッガの動作を検証するには、正常動作が確認済みのこれらのアプリケーションを使うのも 1 つの方法です。
4. デバッガの動作を確認し、アプリケーションを正しくセットアップしてください。詳細は本書の 3.「接続」と 4.「動作」を参照してください。
5. 回路に対してプログラム速度の設定が速過ぎる可能性があります。MPLAB X IDE 内で **File** > **Project Properties** と操作し、**「PICkit Basic」**カテゴリ、**「Program Options」**オプション カテゴリを選択します。**「Program Speed 」**の横で、ドロップダウン メニューからより低い速度を選択します。既定値は「Normal」です。
6. デバッガが正常に動作しておらず、ファームウェアのダウンロードまたはデバッガの再プログラミングが必要な場合があります。以下のセクションを参照して適切な操作を判断してください。

5.4 ハードウェア ツール エマージェンシ ブート ファームウェア リカバリ ユーティリティの使い方

NOTICE **このユーティリティは、ハードウェア ツール ブート ファームウェアを工場出荷時状態に戻すために使います。お客様のハードウェア ツールがどのマシンでも動作しない場合にのみ使ってください。**

まれにデバッガをリカバリブート モード(再プログラミング)に移行させる事が必要になる場合があります。例えば、デバッガがコンピュータに接続されている時に以下の状況が発生した場合です。

- デバッガの LED が点灯しない
- 前のセクションで説明した手順が正常に完了しなかった

MPLAB PICkit Basic 向けにエマージェンシ リカバリ ユーティリティを使うには MPLAB X IDE V6.25 以上が必要です。

MPLAB X IDE 内でメインメニュー オプション **Debug** > **Hardware Tool Emergency Boot Firmware Recovery** を選択し、表示される指示に注意深く従ってください。

図 5-1. エマージェンシ ユーティリティの選択

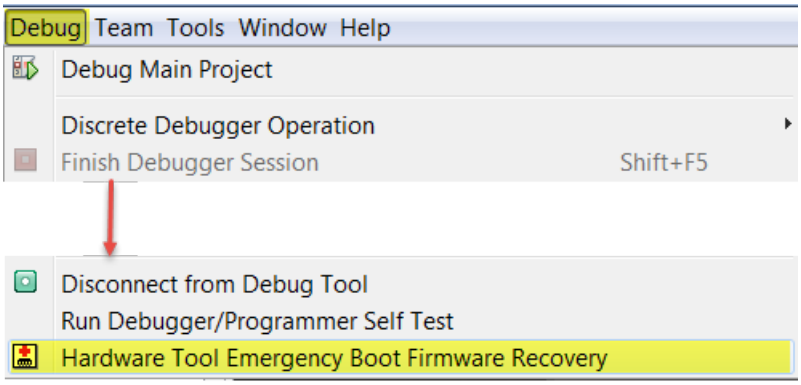


図 5-2 に、MPLAB PICkit Basic インサーキット デバッガのエマージェンシ リカバリボタンの位置を示します。ペーパークリップまたは小型のスクリュー ドライバを使ってこのボタンを押す事ができます。

図 5-2. エマージェンシ リカバリボタン

リカバリに成功すると、リカバリ ウィザードは成功した事を示します。MPLAB PICkit Basic は動作可能となり、MPLAB X IDE と通信できます。

リカバリに失敗した場合、再試行してください。それでも失敗する場合、Microchip 社サポート (support.microchip.com)にお問い合わせください。

6. よく寄せられる質問

以下には、MPLAB® PICkit™ Basic インサーキット デバッガシステムについてよく寄せられる質問とその回答を記載します。

6.1 動作に関する FAQ

MPLAB PICkit Basic インサーキット デバッガは通信用にデバイス内のどの機能を使いますか。

デバイス内のモジュールにより各種の通信が可能です。MPLAB PICkit Basic インサーキット デバッガは、ICSP™インターフェイス(またはアダプタボードを使ってその他のインターフェイス)を介してフラッシュ デバイスと通信できます。通信モジュールがデバイスの内蔵デバッグ回路へのアクセスを許容する場合、デバッガはこのデバッグ回路へアクセスしてデバッグ機能を実行できます。

デバッグ実行プログラムを実行するとプロセッサのスループットはどのように影響を受けますか。

PIC MCU デバイスの場合、デバッグ セッション中にデバッグ実行プログラムがプログラムメモリまたは専用メモリに書き込まれます。デバッグ実行プログラムは実行モードでは動作しないため、コード実行時にスループットは低下しません。つまり、デバッガがターゲット デバイスの実行サイクルを「奪う」事はありません。

MPLAB PICkit Basic インサーキット デバッガには、他のインサーキット エミュレータ/デバッガのような複雑なブレークポイントがありますか。

いいえ、ありません。しかし、特定のデータメモリ アドレスに格納された値またはプログラム アドレスに基づいてブレークを実行する事はできます。

MPLAB PICkit Basic インサーキット デバッガでは複合ブレークポイントが使えますか。

はい使えます。特定のデータメモリ アドレスに格納された値に基づいてブレークする事ができます。複数のイベントが発生した後にブレークするシーケンス化ブレークポイントも使えます。しかし、シーケンスは 2 つしか使えません。AND 条件と PASS カウントも使えます。

MPLAB PICkit Basic インサーキット デバッガは光学的または電気的に絶縁されていますか。

いいえ。このデバッガは絶縁されていないため、非絶縁のライン電圧(100V/120V)回路系との接続はできません。

MPLAB PICkit Basic インサーキット デバッガによってコードの実行速度は低下しますか。

いいえ。デバイスはデータシートに記載されている速度で動作します。

dsPIC DSC デバイスをどの速度で動作させてもデバッグはできますか。

MPLAB PICkit Basic インサーキット デバッガは、デバイスのデータシートに記載されている全ての速度でデバイスを動作させてデバッグを実行できます。

6.2 トラブルに関する FAQ

以下を確認してください。

デバイスのプログラミング後にベリファイで失敗します。これはプログラミングの問題ですか。

[Run Main Project]アイコン()を選択した場合、デバイスはプログラミングの直後に自動的にコードを実行します。従って、コード実行によってフラッシュメモリが書き換えられる場合、ベリファイに失敗する事があります。プログラミング後のコード実行を防ぐには、**「Hold in Reset」**を選択します。

コンピュータが省電力/休止モードに移行した後、デバッガが動作しなくなりました。どうしたのでしょうか。

本デバッガを(特にデバッガとして)長時間使う場合、コンピュータのオペレーティング システムの電源オプション設定画面で休止モードを無効にしておいてください。**[休止状態]**タブを開き、**[休止状態を有効にする]**チェックボックスのチェックを外します。これにより、全ての USB サブシステム コンポーネントで全ての通信を維持できます。

「Freeze on Halt」を設定していない周辺モジュールが突然フリーズしてしまいます。なぜでしょうか。

dsPIC30F/33F と PIC24F/H の場合、デバッガは周辺モジュール制御レジスタの予約済みビット(通常は bit 14 または 5)を Freeze ビットとして使います。レジスタ全体に書き込みを実行した時に、このビットが上書きされた可能性があります。このビットはデバッグモード中にユーザアクセス可能です。

この問題を防ぐには、レジスタ全体を書き換える命令(MOV)ではなく、アプリケーション向けに変更が必要なビットだけを書き換える命令(BTS、BTC)を使います。

16 ビットデバイスを使用中に予期しないリセットが発生しました。どのようにすれば原因を特定できますか。

以下を確認してください。

- RCON レジスタを確認してリセット要因を特定する。
- 割り込みサービスルーチン(ISR)でトラップ/割り込みを処理する(例: 以下のような `trap.c` コードを挿入する)。

```
void  attribute (( interrupt  )) _OscillatorFail(void);
        :
void  attribute (( interrupt  )) _AltOscillatorFail(void);
        :
void  attribute (( interrupt  )) _OscillatorFail(void)
    {
        INTCON1bits.OSCFAIL = 0;        //Clear the trap flag
        while (1);
    }
        :
    void   attribute  ((  interrupt  )) _AltOscillatorFail(void)
    {
        INTCON1bits.OSCFAIL = 0;
        while (1);
    }
    :
```

- ASSERT を使う(例: ASSERT (IPL==7))。

7. エラーメッセージ

MPLAB PICkit Basic インサーキット デバッガは各種のエラーメッセージを生成します。エラーメッセージには特別な対処が必要なものと一般的な対処法で解決できるものがあります。通常エラーメッセージの下に説明が表示されます。説明を読んでも問題を解決できない場合、または説明がない場合、以下のセクションを参照してください。

7.1 エラーメッセージのタイプ

7.1.1 デバッガとターゲット間の通信エラー

Failed to send database (データベースの送信に失敗しました)

対処法:

- もう一度ダウンロードする(一過性エラーの可能性があるため)。
- 最も番号の大きい.jam ファイルの手動ダウンロードを試みる。

これらの対処法では問題が解決しない場合、または説明がない場合、「7.2.3. デバッガとコンピュータ間の通信エラーの対処方法」を参照してください。

7.1.2 インストール ファイルの破損/期限切れエラー

Failed to download firmware (ファームウェアのダウンロードに失敗しました)

HEX ファイルが存在する場合の対処法:

- 再接続してもう一度試してみる。
- それでも解決しない場合、ファイルが破損している可能性があるため MPLAB X IDE または MPLAB IPE を再インストールする。

HEX ファイルが存在しない場合の対処法:

- MPLAB X IDE または MPLAB IPE を再インストールする。

Unable to download debug executive (デバッグ実行プログラムをダウンロードできません)

デバッグ時にこのエラーが表示された場合の対処法:

1. MPLAB PICkit Basic のデバッグツールとしての選択を解除する。
2. プロジェクトを閉じて MPLAB X IDE または MPLAB IPE を終了する。
3. MPLAB X IDE または MPLAB IPE を再起動して再度プロジェクトを開く。
4. MPLAB PICkit Basic をデバッグツールとして再度選択し、ターゲット デバイスのプログラミングを再試行する。

Unable to download program executive (プログラム実行プログラムをダウンロードできません)

プログラミング時にこのエラーが表示された場合の対処法:

1. MPLAB PICkit Basic のプログラマとしての選択を解除する。
2. プロジェクトを閉じて MPLAB X IDE または MPLAB IPE を終了する。
3. MPLAB X IDE または MPLAB IPE を再起動して再度プロジェクトを開く。
4. MPLAB PICkit Basic をプログラマとして再度選択し、ターゲット デバイスのプログラミングを再試行する。

説明を読んでも問題を解決できない場合、または説明がない場合、「インストール ファイルが破損している場合の対処方法」を参照してください。

7.1.3 デバッグ障害エラー

The target device is not ready for debugging. Please check your configuration bit settings and program the device before proceeding (ターゲット デバイスのデバッグの準備が完了していません。コンフィグレーション ビットの設定を確認し、デバイスをプログラミングしてからデバッグしてください)

このメッセージが表示されるのは、デバイスを一度もプログラミングせずに実行しようとした場合です。このような場合、あるいはデバイスをプログラミングした直後にこのメッセージが表示される場合、「7.2.6. デバッグ障害の対処方法」を参照してください。

The device is code protected.(デバイスはコード保護されています)

デバイスのコード保護が有効になっている場合、コードの読み書きと変更(読み出し、書き込み、ブランクチェック、ベリファイ)はできません。コード保護のコンフィグレーション ビット設定を確認してください(**Windows** > **Target Memory Views** > **Configuration Bits**)。

コード保護を無効にするには、該当するコンフィグレーション ビット (デバイスのデータシート参照)をコード内または**[Configuration Bits]**ウィンドウ内でセットまたはクリアします。その後で、デバイス全体を消去してから再プログラミングします。

これらの対処法では問題が解決しない場合、「デバッガとターゲット間の通信エラーの対処方法」と「7.2.6. デバッグ障害の対処方法」を参照してください。

7.1.4 その他のエラー

MPLAB PICkit Basic is busy. Please wait for the current operation to finish. (MPLAB PICkit Basic がビジーです。実行中の動作が終了するまで待機してください)

MPLAB PICkit Basic をデバッガまたはプログラマの選択から解除しようとしてこのエラーが表示された場合の対処法は以下の通りです。

1. 待機して、全てのアプリケーション タスクを終了するための時間をデバッガに与えた後に、MPLAB PICkit Basic の選択解除を再度試みる。
2. **[Finish Debugger Session]**()を選択して実行中のアプリケーションを全て停止した後に、デバッガの選択解除を再度試みる。
3. コンピュータからデバッガの接続ケーブルを抜いた後に、デバッガの選択解除を再度試みる。
4. MPLAB X IDE を終了する。

7.1.5 エラーメッセージの一覧

表 7-1. エラーメッセージ (アルファベット順)

AP_VER=(アルゴリズム プラグイン バージョン)
AREAS_TO_PROGRAM=(以下のメモリ領域をプログラミングします:)
AREAS_TO_READ=(以下のメモリ領域を読み出します:)
AREAS_TO_VERIFY=(以下のメモリ領域をベリファイします:)
BLANK_CHECK_COMPLETE=(ブランクチェック完了、デバイスはブランクです)
BLANK_CHECK_FAILED=(ブランクチェックに失敗しました。デバイスはブランクではありません)
BLANK_CHECKING=(ブランクチェック実行中...)
BOOT_CONFIG_MEMORY=(ブート コンフィグレーション メモリ
BOOT_VER=(ブートバージョン)
BOOTFLASH=(ブートフラッシュ)
BP_CANT_B_DELETED_WHEN_RUNNING=(ターゲット動作中にソフトウェア ブレークポイントを削除する事はできません。選択したブレークポイントは、次のターゲット停止時に削除されます)
CANT_CREATE_CONTROLLER=(ツール コントローラ クラスが見つかりません)
CANT_FIND_FILE=(ファイル%s が見つかりません)
CANT_OP_BELOW_LVPTHRESH=(選択した電圧レベル%f は最低消去電圧%f を下回っています。この電圧レベルでは動作を継続できません)
CANT_PGM_USEROTP=(ユーザ OTP メモリがブランクではないため、デバッグツールはそこに書き込めません。メモリからユーザ OTP メモリを除外するか、ユーザ OTP メモリがブランクのデバイスに切り換えてください)
CANT_PRESERVE_PGM_MEM=(プログラムメモリを保存できません。無効なレンジ: 開始値 = %08x、終了値 = %08x)
CANT_READ_REGISTERS=(ターゲット レジスタを読み出せません)
CANT_READ_SERIALNUM=(デバイスシリアル番号を読み出せません)

CANT_REGISTER_ALTERNATE_PNP=(複数の USB 製品 ID の PNP イベントを登録できません)
CANT_REMOVE_SWPS_BUSY=(PICKit Basic は現在ビジーです。ソフトウェア ブレークポイントは削除できません)
CHECK_4_HIGH_VOLTAGE_VPP=(注意: MPLAB IDE (%s)で選択したデバイスとデバッグツールに物理的に接続したデバイスが同じである事を確認してください。3.3V デバイスを接続した状態で 5V デバイスを選択すると、デバッガがデバイス ID をチェックする際にデバイスが損傷する可能性があります)
CHECK_PGM_SPEED=(プログラム速度は%s に設定されています。ボード上の回路では速度を下げる必要があるようです。ツールプロパティで設定を Low に変更し、再試行してください)
CHECK_SLAVE_DEBUG=(マスタプロジェクトの[Slave Core]設定で[Debug]にチェックが入っていないためにデバッグに失敗した可能性があります。この設定にチェックが入っているか確認してください)
COMM_PROTOCOL_ERROR=(デバッグツールとの通信エラーが発生しました。ツールはリセットされ、すぐにエニュメレートし直す必要があります)
COMMAND_TIME_OUT=(PICkit Basic は、コマンド%02x への応答を待機中にタイムアウトしました)
CONFIGURATION=(コンフィグレーション)
CONFIGURATION_MEMORY=(コンフィグレーション メモリ)
CONNECTION_FAILED=(接続に失敗しました)
CORRUPTED_STREAMING_DATA=(無効なストリーミング データが検出されました。ランタイム ウォッチまたはトレースデータは無効になった可能性があります。デバッグ セッションをリスタートする事を推奨します)
CPM_TO_TARGET_FAILED=(ControlPointMediator.ToTarget()の実行中に例外が発生しました)
DATA_FLASH_MEMORY=(データ フラッシュメモリ)
DATA_FLASH=(データフラッシュ)
DEBUG_INFO_PGM_FAILED=(デバッグ情報のプログラミングに失敗したためデバッグモードに移行できませんでした。コンフィグレーション ビットの無効な組み合わせが原因である可能性があります)
DEBUG_READ_INFO=(ターゲットのオシレータ速度のせいでデバッグモード中のデバイスの読み出しに時間がかかる可能性があります。(PICkit Basic のプロジェクト プロパティで)読み出すレンジを狭めると時間を短縮できます。必要であれば動作を中止する事で読み出し動作を終了できます)
DEVICE_ID_REVISION=(デバイス ID リビジョン)
DEVICE_ID=(デバイス ID)
DEVICE_INFO_CONFIG_BITS_MASK=Address = %08x, Mask = %08x
DEVICE_INFO_MEMBERS=DeviceInfo: pcAddress = %08x, Vpp = %.2f, useRowEraseIfVoltageIsLow = %s, voltageBelowWhichUseRowErase = %.2f, deviceName = %s, programmerType = %s
DEVICE_INFO_MEMINFO_MEMBERS= DeviceInfo: mask = %04x, exists = %s, startAddr = %08x, endAddr = %08x, rowSize = %04x, rowEraseSize = %04x, addrInc = %04x, widthProgram = %04x
DEVICE_INFO=DeviceInfo:Values:
DEVID_MISMATCH=(ターゲット デバイス ID (0x%x)は無効なデバイス ID です。ターゲット デバイスとの接続を確認してください)
DFU_NOT_SUPPORTED=(接続されたツールにサポートしていない機能がある事を MPLAB X が検出しました。このツールを使うには、MPLAB X の最新バージョンをダウンロードしてください)
DISCONNECT_WHILE_BUSY=(ツールが動作中に切断されました)
EEDATA_MEMORY=(EEData メモリ)
EEDATA=EEData
EMPTY_PROGRAM_RANGES=(メモリ領域が選択されていないため、プログラミング動作は完了しませんでした)
EMULATION_MEMORY_READ_WRITE_ERROR=(MPLAB のエミュレーション メモリの読み書き中にエラーが発生しました: アドレス=%08x)
END=(終了)
ENSURE_SELF_TEST_READY=(続行する前に、RJ-11 ケーブルがテストボードに接続されている事を確認してください。続行しますか)
ENV_ID_GROUP=(デバイス ID)
ERASE_COMPLETE=(消去に成功しました)
ERASING=(消去中...)
FAILED_2_PGM_DEVICE=(デバイスのプログラミングに失敗しました)
FAILED_CREATING_COM=(通信オブジェクトを作成できません)

FAILED_CREATING_DEBUGGER_MODULES=(初期化に失敗しました: デバッガ モジュールの作成に失敗)
FAILED_ERASING=(デバイスの消去に失敗しました)
FAILED_ESTABLISHING_COMMUNICATION=(ツールの通信を確立できません)
FAILED_GETTING_DBG_EXEC=(デバッグ実行プログラムの読み込み中に問題が発生しました)
FAILED_GETTING_DEVICE_INFO=(初期化に失敗しました: デバイス データベース(.pic)情報の取得に失敗)
FAILED_GETTING_EMU_INFO=(初期化に失敗しました: エミュレーション データベース情報の取得に失敗)
FAILED_GETTING_HEADER_INFO=(初期化に失敗しました: ヘッダデータベース情報の取得に失敗)
FAILED_GETTING_PGM_EXEC=(プログラム実行プログラムの読み込み中に問題が発生しました)
FAILED_GETTING_TEX=(ToolExecMediator を取得できません)
FAILED_GETTING_TOOL_INFO=(初期化に失敗しました: ツールデータベース(.ri4)情報の取得に失敗)
FAILED_INITING_DATABASE=(初期化に失敗しました: ツールデータベース オブジェクトを初期化できません)
FAILED_INITING_DEBUGHANDLER=(初期化に失敗しました: DebugHandler オブジェクトを初期化できません)
FAILED_PARSING_FILE=(ファームウェア ファイル: %s の構文解析に失敗しました)
FAILED_READING_EMULATION_REGS=(エミュレーション メモリの読み出しに失敗しました)
FAILED_READING_MPLAB_MEMORY=(%0x08 から%0x08 に%s メモリを読み出せません)
FAILED_READING_SECURE_SEGMENT=(セキュア セグメント コンフィグレーション ビット読み出し中にエラーが発生しました)
FAILED_SETTING_PC=(PC を設定できません)
FAILED_SETTING_SHADOWS=(シャドーレジスタの適切な設定に失敗しました)
FAILED_SETTING_XMIT_EVENTS=(ランタイムデータ セマフォを同期できません)
FAILED_STEPPING=Failed while stepping the target. (ターゲットのステップ動作に失敗しました)
FAILED_TO_GET_DEVID=(デバイス ID の取得に失敗しました。ターゲット デバイスが接続されている事を確認し、操作を再試行してください)
FAILED_TO_INIT_TOOL=(PICkit Basic の初期化に失敗しました)
FAILED_UPDATING_BP=(ブレークポイントの更新に失敗しました:\n ファイル: %s\naddress: %08x)
FAILED_UPDATING_FIRMWARE=(ファームウェアの更新に失敗しました)
FILE_REGISTER=(ファイルレジスタ)
FIRMWARE_DOWNLOAD_TIMEOUT=(ファームウェア ダウンロード処理中に PICkit Basic がタイムアウトしました)
FLASH_DATA_MEMORY=(フラッシュデータ メモリ)
FLASH_DATA=(フラッシュデータ)
FRCINDEBUG_NEEDS_CLOCKSWITCHING=(デバッグモードで FRC を使うには、クロック切り換えコンフィグレーション ビットの設定を有効にする必要があります。クロック切り換えを有効にし、要求動作を再試行してください)
FW_DOESNT_SUPPORT_DYNBP=(現在の PICkit Basic ファームウェアは、選択されたデバイスに対するランタイム ブレークポイントの設定をサポートしていません。ファームウェア バージョン%02x.%02x.%02x 以上をダウンロードしてください)
GOOD_ID_MISMATCH=(ターゲット デバイス ID (0x%x)は有効なデバイス ID ですが、選択された要求デバイス ID (0x%x)と一致しません)
HALTING=(停止中...)
HIGH=High
HOLDMCLR_FAILED=(リセット維持に失敗しました)
IDS_SELF_TEST_BOARD_PASSED=(PICkit Basic は正常に動作しています。ターゲット回路の問題が解決しない場合、オンラインヘルプの「Target Board Considerations」セクションを確認してください)
IDS_ST_CLKREAD_ERR=(テストインターフェイス PGC クロックラインの読み出しエラーです)
IDS_ST_CLKREAD_NO_TEST=(テストインターフェイス PGC クロックラインの読み出しテストはしていません)
IDS_ST_CLKREAD_SUCCESS=(テストインターフェイス PGC クロックラインの読み出しに成功しました)
IDS_ST_CLKWRITE_ERR=(テストインターフェイス PGC クロックラインの書き込みエラーです。テスタが正しく接続されている事を確認してください)
IDS_ST_CLKWRITE_NO_TEST=(テストインターフェイス PGC クロックラインの書き込みテストはしていません)
IDS_ST_CLKWRITE_SUCCESS=(テストインターフェイス PGC クロックラインの書き込みに成功しました)
IDS_ST_DATREAD_ERR=(テストインターフェイス PGD データラインの読み出しエラーです)
IDS_ST_DATREAD_NO_TEST=(テストインターフェイス PGD データラインの読み出しテストはしていません)
IDS_ST_DATREAD_SUCCESS=(テストインターフェイス PGD データラインの読み出しに成功しました)

IDS_ST_DATWRITE_ERR=(テストインターフェイス PGD データラインの書き込みエラーです)
IDS_ST_DATWRITE_NO_TEST=(テストインターフェイス PGD データラインの書き込みテストはしていません)
IDS_ST_DATWRITE_SUCCESS=(テストインターフェイス PGD データラインの書き込みに成功しました)
IDS_ST_LVP_ERR=(テストインターフェイス LVP 制御ラインのエラーです)
IDS_ST_LVP_NO_TEST=(テストインターフェイス LVP 制御ラインのテストはしていません)
IDS_ST_LVP_SUCCESS=(テストインターフェイス LVP 制御ラインのテストに成功しました)
IDS_ST_MCLR_ERR=(テストインターフェイス MCLR レベルエラーです)
IDS_ST_MCLR_NO_TEST=(テストインターフェイス MCLR レベルのテストはしていません)
IDS_ST_MCLR_SUCCESS=(テストインターフェイス MCLR レベルテストに成功しました)
IDS_TEST_NOT_COMPLETED=(インターフェイス テストを完了できませんでした。修理については Microchip 社正規代理店にお問い合わせください)
INCOMPATIBLE_FW=(PICkit Basic ファームウェアが MPLAB X ソフトウェアの現バージョンと互換ではありません)
INVALID_ADDRESS=(%s アドレスがデバイスアドレス レンジ 0x%08x - 0x%08x を外れているため、動作を実行できません)
JTAG_NEEDS_JTAGEN=(JTAG アダプタは、JTAG イネーブル コンフィグレーション ビットの有効化を必要とします。続行する前に、このコンフィグレーション ビットを有効にしてください)
MCLR_HOLD_RESET_NO_MAINTAIN_POWER=(警告: PICkit Basic からからターゲット デバイスへ給電していますが、PICkit Basic の[Power]プロパティページで「Maintain active power」オプションが選択されていません。このオプションを選択しないと、現在のセッション終了後の MCLR のステート(ホールド/リセットからのリリース)を保証できません)
MCLR_OFF_ID_WARNING=(デバイス ID が正しくないとの警告が出た場合、低電圧プログラミングを使っているのにターゲット デバイスの MCLRE コンフィグレーション ビットを OFF にしている可能性が考えられます。この場合、[PICkit Basic Program Options]プロパティページでプログラムモード エントリを\"Use high voltage programming mode entry\"に切り換えてから再試行してください)
MCLR_OFF_WARNING=(MCLRE コンフィグレーション ビットを OFF にしたまま続行する場合、[PICkit Basic Program Options]プロパティページでプログラムモード エントリの設定を\"Use high voltage programming mode entry\"に切り換えてください)
MEM_INFO=DeviceInfo: MemInfo values:
MEM_RANGE_ERROR_BAD_END_ADDR=(無効なプログラムレンジ終了アドレス%s を受信しました。デバッグツールの[Memories to Program]プロパティページで手動プログラムレンジを確認してください)
MEM_RANGE_ERROR_BAD_START_ADDR=(無効なプログラムレンジ開始アドレス%s を受信しました。デバッグツールの[Memories to Program]プロパティページで手動プログラムレンジを確認してください)
MEM_RANGE_ERROR_END_LESSTHAN_START=(無効なプログラムレンジを受信しました: 終了アドレス%s < 開始アドレス%s。デバッグツールの[Memories to Program]プロパティページで手動プログラムレンジを確認してください)
MEM_RANGE_ERROR_ENDADDR_NOT_ALIGNED=(無効なプログラムレンジを受信しました。終了アドレス%s が正しい 0x%x アドレス境界にアラインメントされていません。デバッグツールの[Memories to Program]プロパティページで手動プログラムレンジを確認してください)
MEM_RANGE_ERROR_STARTADDR_NOT_ALIGNED=(無効なプログラムレンジを受信しました。開始アドレス%s が正しい 0x%x アドレス境界にアラインメントされていません。デバッグツールの[Memories to Program]プロパティページで手動プログラムレンジを確認してください)
MEM_RANGE_ERROR_UNKNOWN=(ユーザが入力したメモリ領域検証中に不明なエラーが発生しました)
MEM_RANGE_ERROR_WRONG_DATABASE=(ユーザが入力したメモリ領域検証中にデータオブジェクトにアクセスできませ
MEM_RANGE_OUT_OF_BOUNDS=(選択されたプログラムレンジ%s は、選択されたメモリ領域の正しいレンジから外れています。デバッグツールの[Memories to Program]プロパティページで手動プログラムレンジを確認してください)
MEM_RANGE_STRING_MALFORMED=([Memories to Program]プロパティページで入力したメモリ領域(%s)が正しくフォーマットされていません)
MISSING_BOOT_CONFIG_PARAMETER=(データベース内にブート コンフィグレーションの開始/終了アドレスが見つかりま
MUST_NOT_USE_LVP_WHEN_LVPCFG_OFF=(MPLAB はデバイスの低電圧コンフィグレーション ビットが OFF である事を検出しましたが、デバッグツールのプロパティページでは低電圧プログラミング オプションが選択されています。低電圧プログラミング オプションを使用する場合、まず以下の操作を実行する必要があります。\n* デバッグツールの[Program Options]プロパティページで低電圧プログラミング オプションを OFF にする\n* 低電圧コンフィグレーション ビットを ON に設定する\n* デバッグツールの[Program Options]プロパティページで低電圧プログラミング オプションを ON にする)
MUST_SET_LVPBIT_WITH_LVP=(低電圧プログラミング機能では、ターゲット デバイスで LVP コンフィグレーション ビットを有効にする必要があります。このコンフィグレーション ビットを有効にしてから再試行してください)
NEW_FIRMWARE_NO_DEVICE=(ファームウェア ダウンロード中)
NEW_FIRMWARE=(ターゲット デバイス: %s の新しいファームウェアをダウンロードしています)
NMMR=NMMR

NO_DYNAMIC_BP_SUPPORT_AT_ALL=(ご使用中のデバイスは、動作中のブレークポイント設定機能をサポートしていません。ブレークポイントは次のデバイス動作前に適用されます)
NO_PGM_HANDLER=(ソフトウェア ブレークポイントをプログラムできません。プログラムハンドラが初期化されていません)
NO_PROGRAMMING_ATTEMPTED=(MPLAB のメモリはブランクであるため、プログラミング動作は何も試行されませんでした)
NORMAL=Normal
OP_FAILED_FROM_CP=(デバイスがコード保護されているため動作は失敗しました)
OpenIDE-Module-Name=PICkit Basic
OPERATION_INFO_MEMBERS=OperationInfo:Type = %s, Mask = %08x, Erase = %s, Production Mode = %s.
OPERATION_INFO_TRANSFER_INFO_MEMBERS=OperationInfo: Start = %x, End = %x, Buffer Length = %d, Type = %s, Mask =
OPERATION_INFO=OperationInfo:Values:
OPERATION_NOT_SUPPORTED=(この動作は選択したデバイスではサポートされていません)
OUTPUTWIN_TITLE=PICkit Basic
PERIPHERAL=Peripheral (周辺モジュール)
POWER_ERROR_NO_POWER_SRC=(ターゲットボードは自己給電するように設定されていますが、VDD で電圧が検出されません。ターゲットに給電されている事を確認してから再試行してください)
POWER_ERROR_POWER_SRC_CONFLICT=(ツールからターゲットに給電するように設定されていますが、VDD で既に電圧が検出されています。ターゲットからツールに給電していない事を確認してから再試行してください)
POWER_ERROR_SLOW_DISCHARGE=(VDD の静電容量が大き過ぎるためにシステムの放電とシャットダウンに遅延が生じているようです。放電の遅延を防ぐため、総静電容量負荷を抑えるかターゲットから給電する事を検討してください)
POWER_ERROR_UNKNOWN=(不明な電源エラーが発生しました)
POWER_ERROR_VDD_TOO_HIGH=(要求 VDD 電圧がレンジ外です。最大電圧 5.5V を上回っています)
POWER_ERROR_VDD_TOO_LOW=(要求 VDD 電圧がレンジ外です。最小電圧 1.5V を下回っています)
POWER_ERROR_VPP_TOO_HIGH=(要求 VPP 電圧がレンジ外です。最大電圧 14.2V を上回っています)
POWER_ERROR_VPP_TOO_LOW=(要求 VDD 電圧がレンジ外です。最小電圧 1.5V を下回っています)
PRESERVE_MEM_RANGE_ERROR_BAD_END_ADDR=(無効な保存領域終了アドレス%s を受信しました。デバッグツールの[Memories to Program]プロパティページで手動プログラムレンジを確認してください)
PRESERVE_MEM_RANGE_ERROR_BAD_START_ADDR=(無効な保存領域開始アドレス%s を受信しました。デバッグツールの[Memories to Program]プロパティページで手動プログラムレンジを確認してください)
PRESERVE_MEM_RANGE_ERROR_END_LESSTHAN_START=(無効な保存領域を受信しました: 終了アドレス%s < 開始アドレス%s。デバッグツールの[Memories to Program]プロパティページで手動プログラムレンジを確認してください)
PRESERVE_MEM_RANGE_ERROR_ENDADDR_NOT_ALIGNED=(無効な保存領域を受信しました: 終了アドレス%s が正しい 0x%x アドレス境界にアラインメントされていません。デバッグツールの[Memories to Program]プロパティページで手動プログラムレンジを確認してください)
PRESERVE_MEM_RANGE_ERROR_STARTADDR_NOT_ALIGNED=(無効な保存領域を受信しました: 開始アドレス%s が正しい 0x%x アドレス境界にアラインメントされていません。デバッグツールの[Memories to Program]プロパティページで手動プログラムレンジを確認してください)
PRESERVE_MEM_RANGE_ERROR_UNKNOWN=(ユーザが入力した保存領域検証中に不明なエラーが発生しました)
PRESERVE_MEM_RANGE_ERROR_WRONG_DATABASE=(ユーザが入力したメモリ領域検証中にデータオブジェクトにアクセスできません)
PRESERVE_MEM_RANGE_MEM_NOT_SELECTED=(ユーザは 1 つのメモリ領域を保存するよう選択しましたが、その領域へのプログラミングは選択していません。デバッグツールの[Memories to Program]プロパティページで保存領域を確認し、保存メモリのプログラミングも指定しているか確認してください)
PRESERVE_MEM_RANGE_OUT_OF_BOUNDS=(選択されている保存領域%s は、選択されたメモリ領域の正しいレンジから外れています。デバッグツールの[Memories to Program]プロパティページで手動プログラムレンジを確認してください)
PRESERVE_MEM_RANGE_STRING_MALFORMED=([Memories to Program]プロパティページで入力した保存メモリ領域(%s)が正しくフォーマットされていません)
PRESERVE_MEM_RANGE_WONT_BE_PROGRAMMED_AUTO_SELECT=([Memories to Program]プロパティページで入力された保存メモリ領域(%s)の一部または全部が、選択されたメモリの指定されたプログラム領域(%s)から外れています。[Memories to Program]プロパティページの[Auto select memories and ranges]オプションの選択を外して手動モードに変更し、適切なレンジに調整してください)

PRESERVE_MEM_RANGE_WONT_BE_PROGRAMMED=([Memories to Program]プロパティページで入力された保存メモリ領域(%s)の一部または全部が、選択されたメモリの指定されたプログラム領域(%s)から外れています。デバッグツールの[Memories to Program]プロパティページで保存レンジを確認してください)
PROGRAM_CFG_WARNING=(警告: ユーザはコンフィグレーション メモリのプログラミングを選択しました。コンフィグレーション フィールドに無効な値をプログラミングすると予期せぬ結果が生じる可能性があります。全てのコンフィグレーション フィールドに有効な値が書き込まれている事を確認してください。確信が持てない場合、まずデバイスからコンフィグレーション値を読み出し、必要なフィールドのみ変更してください。プログラミングを続行しますか)
PROGRAM_COMPLETE=(プログラミング/ベリファイが完了しました)
PROGRAM_MEMORY=(プログラムメモリ)
PROGRAM=(プログラム)
PROGRAMMING_DID_NOT_COMPLETE=(プログラミングは完了しませんでした)
READ_COMPLETE=(読み出しは完了しました)
READ_DID_NOT_COMPLETE=(読み出しは完了しませんでした)
RELEASEMCLR_FAILED=(リセットからのリリースに失敗しました)
REMOVING_SWBPS_COMPLETE=(ソフトウェア ブレークポイントの消去が完了しました)
REMOVING_SWBPS=(ソフトウェア ブレークポイントを消去しています)
RESET_FAILED=(デバイスのリセットに失敗しました)
RESETTING=(リセット中...)
RISKY_CFG_RANGE_REMOVED=([Exclude configuration memory from programming]オプションが設定されているため、コンフィグレーション メモリはプログラミング動作に含まれません。これを変更するには、[Memories to Program]プロパティページでこの設定のチェックを外します。警告: このデバイスでコンフィグレーション値をプログラミングした場合、全てのコンフィグレーション値が正しく設定されていないと予期せぬ結果が生じる可能性があります)
RUN_INTERRUPT_THREAD_SYNCH_ERROR=(内部実行エラーが発生しました。デバッグ セッションをリスタートする事を推奨します。動作は続行可能ですが、一部のランタイム機能が正常に機能しなくなる可能性があります)
RUN_TARGET_FAILED=(ターゲット デバイスを動作させる事ができません)
RUNNING=(動作中)
SERIAL_NUM=(シリアル番号:)
SETTING_SWBPS=(ソフトウェア ブレークポイントを設定中.....)
STACK=(スタック)
START_AND_END_ADDR=(開始アドレス = 0x%x、終了アドレス = 0x%x)
START=(開始)
TARGET_DETECTED=(ターゲット電圧を検出しました)
TARGET_FOUND=(ターゲット デバイス%s が見つかりました)
TARGET_HALTED=(ターゲットが停止しました)
TARGET_NOT_READY_4_DEBUG=(ターゲット デバイスのデバッグの準備が完了していません。コンフィグレーション ビットの設定を確認し、デバイスをプログラミングしてからデバッグしてください) 一般的に、このエラーはオシレータとPGC/PGD の両方またはどちらか一方の設定が原因で発生します)
TARGET_VDD=(ターゲット VDD:)
TEST=test
TOOL_INFO_MEMBERS=ToolInfo: speedLevel = %d, PGCResistance = %d, PGDResistance = %d, PGCPullDir = %s, PGDPullDir = %s, ICSPSelected = %s.
TOOL_INFO=ToolInfo:Values:
TOOL_IS_BUSY=(PICkit Basic がビジーです。実行中の動作が終了するまで待機してください)
TOOL_SUPPLYING_POWER=(PICKit Basic がターゲットに給電しています(%.2f V))
TOOL_VDD=VDD:
TOOL_VPP=VPP:
UNABLE_TO_OBTAIN_RESET_VECTOR=(PICKit Basic はリセット ベクタアドレスを取得できませんでした。_reset シンボルが未定義であり、デバイスの正常な起動が妨げられている可能性があります)
UNKNOWN_MEMTYPE=(不明なメモリタイプ)

UNLOAD_WHILE_BUSY=(PICKit Basic がまだビジーである間にアンロードされました。USB ケーブルを一度抜いて挿し直してください)
UPDATING_APP=(ファームウェア アプリケーションを更新中...)
UPDATING_BOOTLOADER=(ファームウェア ブートローダを更新中)
USE_LVP_PROGRAMMING=(低電圧プログラミングを使ってこのデバイスをプログラミングする場合、このダイアログで「Cancel」を選択します。次に、プロジェクト プロパティの PICkit Basic ノードで[Program Options Option Category]ペインの[Enable Low Voltage Programming]チェックボックスにチェックを入れます(低電圧プログラミンがデバッグ動作向けに有効になっていません))
USERID_MEMORY=(ユーザ ID メモリ)
USERID=(ユーザ ID)
VERIFY_COMPLETE=(ベリファイが成功しました)
VERIFY_FAILED=(ベリファイに失敗しました)
VERSIONS=(バージョン)
VOLTAGE_LEVEL_BAD_VALUE_EX=(PICkit Basic の[Power]プロパティページで[Voltage Level]に無効な値%s が入力されました。続行する前にこれを修正してください)
VOLTAGE_LEVEL_BAD_VALUE=(電圧レベル%s を解析できません。有効な電圧を入力してください)
VOLTAGE_LEVEL_OUT_OF_RANGE=(入力されたターゲット電圧レベル%.3f は、デバイスのレンジ%.3f～%.3f から外れています)
VOLTAGES=(電圧)
WOULD_YOU_LIKE_TO_CONTINUE=(続行しますか)
WRONG_PICkit Basic_FLAVOR=(ご使用の PICkit Basic ハードウェアは更新が必要です。PICkit Basic_Update@microchip.com にお問い合わせください)

7.2 一般的な対処方法

7.2.1 読み書きエラーの対処方法

読み書きエラーが発生した場合の対処方法:

1. **Debug** > **Reset** を選択しませんでしたか。この操作によって読み書きエラーが発生する事があります。
2. 同じ操作を繰り返してみてください(一過性エラーの可能性があるため)。
3. ターゲットに電源が供給されており、デバイスの電圧レベルが適正である事を確認します。デバイスに必要な電圧レベルはデバイスのデータシートで確認してください。
4. デバッガとターゲット間が正しく接続されている事(PGC と PGD が接続されている事)を確認します。
5. 書き込みエラーの場合、**[Project Properties]**ウィンドウ内で、デバッガ向けの**[Program Options]**で**[Erase all before Program]**にチェックが入っている事を確認します。
6. ケーブルの長さが適切である事を確認します。

関連リンク

8.4. Debug オプション カテゴリ

7.2.2 デバッガとターゲット間の通信エラーの対処方法

MPLAB PICkit Basic インサーキット デバッガとターゲット デバイスが通信していない可能性があります。

1. **Debug** > **Reset** を選択した後に同じ操作を繰り返します。
2. ケーブルの長さが適切である事を確認します。

7.2.3 デバッガとコンピュータ間の通信エラーの対処方法

MPLAB PICkit Basic インサーキット デバッガと MPLAB X IDE または MPLAB IPE が通信していない可能性があります。

1. デバッガとコンピュータ間の接続ケーブルを一度抜いて挿し直します。
2. デバッガに再接続します。

3. 同じ操作を繰り返します(一過性エラーの可能性があるため)。
4. インストールされている MPLAB X IDE または MPLAB IPE のバージョンが、MPLAB PICkit Basic インサーキット デバッガに読み込まれているファームウェアのバージョンに対応していない可能性があります。「7.2.4. インストール ファイルが破損している場合の対処方法」の手順に従ってください。
5. コンピュータの USB ポートに問題があるかもしれません。「7.2.5. USB ポート通信エラーの対処方法」を参照してください。

7.2.4 インストール ファイルが破損している場合の対処方法

ほとんどの場合、この問題は MPLAB X IDE または MPLAB IPE のインストールが不完全であるか、インストールしたファイルが破損した事が原因で発生します。

1. 全てのバージョンの MPLAB X IDE または MPLAB IPE をコンピュータからアンインストールします。
2. 最新バージョンの MPLAB X IDE または MPLAB IPE を再インストールします。
3. それでも解決しない場合は Microchip 社にお問い合わせください。

7.2.5 USB ポート通信エラーの対処方法

ほとんどの場合、このエラーは通信ポートに問題があるか、存在しない通信ポートを指定している事が原因で発生します。

1. MPLAB PICkit Basic インサーキット デバッガを再接続します。
2. デバッガがコンピュータの適切な USB ポートに物理的に接続されている事を確認します。
3. **[Project Properties]**ウィンドウ内のデバッガ オプションで適切な USB ポートが選択されている事を確認します。
4. 指定した USB ポートを他のデバイスが使っていない事を確認します。
5. USB ハブを使う場合、電源が供給されている事を確認します。
6. USB ドライバがインストールされている事を確認します。

7.2.6 デバッグ障害の対処方法

各種の状況で「MPLAB PICkit Basic インサーキット デバッガはデバッグを実行できませんでした」と言うエラーメッセージが表示されます。そのような状況のいくつかを以下に示します。

1. デバイスがクロック源を備えていないか、コンフィグレーション ビットで選択されたクロック源が動作していない。
2. デバイスの PGD(データ)ピンと PGC(クロック)ピンがアプリケーションによって使われている。デバッガはこれらのピンを必要とするため、デバッグモード中にこれらのピンがアプリケーションによって制御されてはいけません。
3. 多くのデバイスは複数の PGD/PGC ペアを備えています。デバッガ用に使われる PGD/PGC は、コンフィグレーション ビットの設定により選択されます。プログラミング モードの場合、デバッグツールに接続された任意の PGD/PGC ペアが使えます。デバッグモードの場合、コンフィグレーション ビットの設定は MCU に物理的に接続されている PGD/PGC ペアと一致する必要があります。
4. デバイスが未プログラミングである場合に MPLAB X IDE から「Run Main Project」を選択すると、このメッセージが表示されます。代わりにデバッグモード()を使ってください。

この情報は弊社ウェブサイト内の知識ベース記事「Why do I get the following error while trying to debug: "ICD3Err0040: The target device is not ready for debugging"?」に基づきます。

7.2.7 内部エラーの対処方法

内部エラーは想定外のエラーであり、通常は発生しません。これらは主に Microchip 社内の開発用に使われます。

ほとんどの場合、インストールファイルの破損が原因で発生します(「7.2.4. インストール ファイルが破損している場合の対処方法」参照)。また、システムリソースの一時的な不足によって発生する場合もあります。

1. システムを再起動してメモリを解放します。
2. HDD に十分な空き容量がある事と、過度なフラグメンテーションが発生していない事を確認します。

それでも解決しない場合は Microchip 社にお問い合わせください。

8. デバッガ機能のまとめ

以下には、MPLAB PICkit Basic インサーキット デバッガの機能をまとめて記載しています。

8.1 デバッガの選択と切り換え

プロジェクトで使うデバッガの選択と切り換えには**[Project Properties]**ダイアログを使います。コンピュータに複数のデバッガを接続している場合にのみ、デバッガの切り換えが可能です。MPLAB X IDE はシリアル番号でデバッガを区別します。

プロジェクトで使うデバッガを選択または切り換えるための手順は以下の通りです。

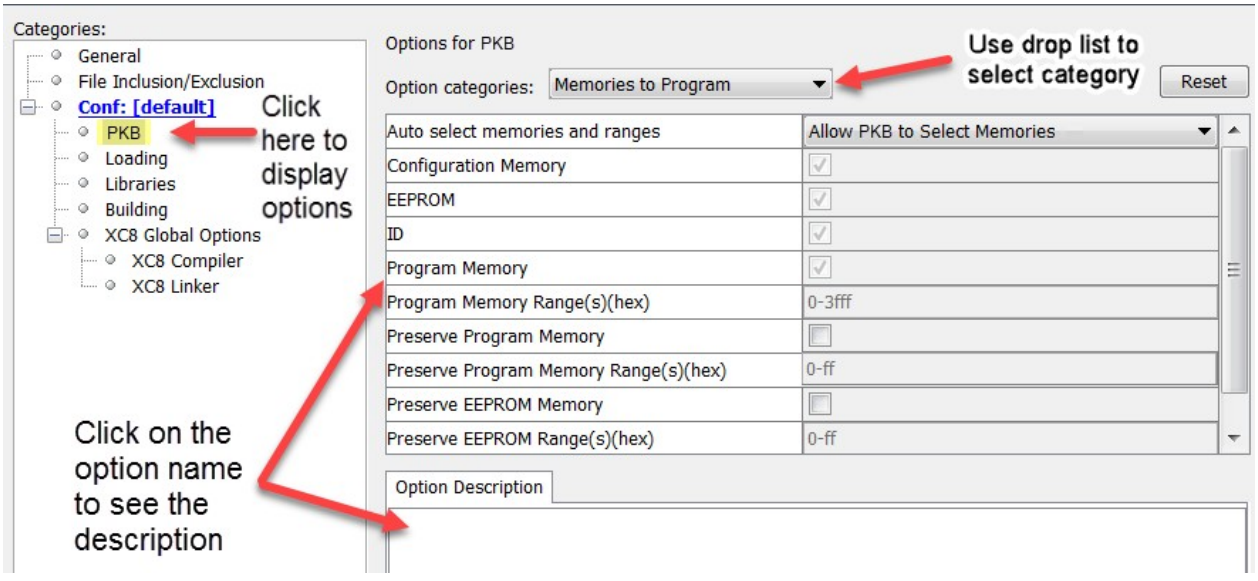
1. 以下のどちらかの方法で**[Project Properties]**ダイアログを開きます。
   a. **[Projects]**ウィンドウでプロジェクト名をクリックし、**File** > **Project Properties** を選択する
   または
   b. **[Projects]**ウィンドウでプロジェクト名を右クリックし、**「Properties」**を選択する
2. 左側の**「Categories:」**の下に表示される **Conf:[default]** を展開して PICkit Basic を表示させます。
3. **[Hardware Tools]**に表示される**「PICkit Basic」**の中からプロジェクトで使うデバッガのシリアル番号(SN)をクリックし、**[Apply]**をクリックします。

8.2 デバッガ オプションの選択

デバッガのオプションは MPLAB® X IDE 内の**[Project Properties]**で設定します。**「Categories:」**の下で**「PKB」**をクリックすると**「Options for PKB」**が表示されます(図 8-1 参照)。**[Options categories]** ドロップダウン リストを使って各種オプションを選択します。オプション名をクリックすると、下の**[Option Description]**ボックスに説明が表示されます。オプションを選択または変更するには、オプション名の右側をクリックします。

Note: 利用可能なオプション カテゴリと、それらのカテゴリ内のオプションは、選択したデバイスによって異なります。

図 8-1. MPLAB PICkit Basic 向けの MPLAB® X IDE オプション



オプションの設定が完了したら、**[Apply]**または**[OK]**をクリックします。MPLAB X IDE ダッシュボード画面内で**[Refresh Debug Tool]**アイコン( )をクリックする事により変更を適用する事もできます。

MPLAB IPE の場合、MPLAB PICkit Basic 向けオプションは **Settings** > **Advance Mode** > **Settings** にあります。詳細は MPLAB IPE オンラインヘルプを参照してください。

8.3 「Memories to Program」オプション カテゴリ

プログラミングするターゲットのメモリを選択します。下表に全てのオプションを示します。ただし、MPLAB X IDE には選択したデバイスで使えるオプションのみ表示されます。
Note: 「Erase All Before Program」が選択されていると、プログラミングの前に全てのデバイスメモリが消去されます(**8.5 「Program」オプション カテゴリ** 参照)。

表 8-1. 「Memories to Program」オプション カテゴリ

Auto select memories and ranges	**Allow PICkit Basic to Select Memories** - デバッガはユーザが選択したデバイスと既定値設定に基づいてプログラミングする内容を決定します。 **Manually select memories and ranges** - プログラミングするメモリのタイプとレンジをユーザが選択します(以下のオプションを指定)。
Configuration Memory	**コンフィグレーション メモリ**をプログラミング領域に含める場合にチェックを入れます。デバッグモードでは常にプログラミングされます。
Boot Flash	**ブートフラッシュ** メモリをプログラミング領域に含める場合にチェックを入れます。デバッグモードでは常にプログラミングされます。
EEPROM	**EEPROM** メモリをプログラミング領域に含める場合にチェックを入れます。
ID	ユーザ ID をプログラミングする場合にチェックを入れます。
Program Memory	下の「Program Memory Range(s)(hex)」オプションによって指定されたターゲット プログラムメモリ領域をプログラミングする場合にチェックを入れます。
Program Memory Range(s)(hex)	プログラミングするプログラムメモリのレンジを指定します(複数レンジを指定可能)。各レンジは書き込み、読み出し、ベリファイを行うプログラムメモリ領域の開始アドレスと終了アドレス(16 進値)により指定します。開始アドレスと終了アドレスを表す 2 つの 16 進数をダッシュ文字「-」で区切る必要があります。複数のレンジを指定する場合、各レンジをコンマで区切ります(例: 0-ff, 200-2ff)。各レンジは 0x800 アドレス境界にアラインメントする必要があります。 **Note:** ここで指定したアドレスレンジは消去機能には適用されません。消去機能はデバイス上の全てのデータを消去します。
Preserve Program Memory	このオプションを有効にすると、デバイス上の現在のプログラムメモリが MPLAB X IDE のメモリへ読み出され、プログラミング完了後にターゲット デバイスへ書き戻されます。保存するプログラムメモリのレンジは、下の「Preserve Program Memory Range(s)」オプションで設定します。コード保護が有効になっていない事が必要です。
Preserve Program Memory Range(s) (hex)	保存するプログラムメモリのレンジを指定します。各レンジは、開始アドレスと終了アドレスを表す 2 つの 16 進数をダッシュ文字「-」で区切って指定します。複数のレンジを指定する場合、各レンジをコンマで区切ります(例: 0-ff, 200-2ff)。指定されたレンジは MPLAB X IDE 内へ読み出す事により保存され、プログラミング動作が発生した時に書き戻されます。このため、保存領域はプログラミング対象のメモリレンジ内にある事が必要です。
Preserve (Type of) Memory	このオプションを有効にすると、デバイス上のこのタイプ(Type of)のメモリを MPLAB X IDE のメモリに読み出し、プログラミング完了時にターゲット デバイスへ書き戻します。チェックを入れると、下の「Preserve (Type of) Memory Range(s) (hex)」で指定されたターゲット メモリ領域が再プログラミングから保護されます。メモリのタイプには EEPROM、ID、ブートフラッシュ、補助メモリが含まれます。コードの保護が有効になっていない事が必要です。
Preserve (Type of) Memory Range(s) (hex)*	このタイプ(Type) のメモリを保存するアドレスレンジを指定します。各レンジは、開始アドレスと終了アドレスを表す 2 つの 16 進数をダッシュ文字「-」で区切って指定します。複数のレンジを指定する場合、各レンジをコンマで区切ります(例: 0-ff, 200-2ff)。指定されたレンジは MPLAB X IDE 内へ読み出す事により保存され、プログラミング動作が発生した時に書き戻されます。このため、保存領域はプログラミング対象のメモリレンジ内にある事が必要です。メモリのタイプには EEPROM、ID、ブートフラッシュ、補助メモリが含まれます。コードの保護が有効になっていない事が必要です。
* レンジが正しくないためにプログラミング エラーが発生する場合、指定したレンジがデバイスメモリ内で利用可能な範囲を超えていないか確認してください。	

8.4 「Debug」オプション カテゴリ

このオプションがプロジェクト デバイスに対して使える場合、このオプションを選択する事でソフトウェア ブレークポイントが使えます。

表 8-2. 「Debug」オプション カテゴリ

Debug startup	デバイスの起動後にデバッグ セッションを開始します。
Debug reset	リセット後にデバッグ セッションを開始します。
Use Software Breakpoints	ソフトウェア ブレークポイントを使います。

表 8-3. ソフトウェア ブレークポイントとハードウェア ブレークポイントの比較

項目	ソフトウェア ブレークポイント	ハードウェア ブレークポイント
ブレークポイントの数	制限なし	制限あり
ブレークポイントの書き込み先	プログラムメモリ	デバッグレジスタ
ブレークポイントの設定に要する時間	オシレータ速度に依存 数分かかる場合があります	最小限
スキッド	No	Yes

Note: デバッグでソフトウェア ブレークポイントを使うと、デバイスの書き込み耐性に影響します。従って、ソフトウェア ブレークポイントを使ったデバイスは量産製品に使わない事を推奨します。

8.5 「Program」オプション カテゴリ

プログラミング前にメモリ全体を消去するかどうかを選択します。消去しない場合、コードがマージされます。

表 8-4. 「Program」オプション カテゴリ

Erase All Before Program	チェックを入れると、デバイス全体を消去してから MPLAB X IDE からのデータをプログラミングします。保存するように指定されたメモリ領域はデバイス消去前に読み出され、プログラミング時に書き戻されます。新しいデバイスまたは消去済みデバイスにプログラミングする場合を除き、このボックスには必ずチェックを入れます。チェックを入れないとデバイスは消去されず、デバイス上で既存のコードと新しいコードがマージされます。

8.6 「PICkit Basic Tool」オプション カテゴリ

PICkit Basic ツール専用のオプションです。

表 8-5. 「PICkit Basic Tool」オプション カテゴリ

Programming mode entry	低電圧プログラミング モードエントリを使用します(常に選択)。PICkit Basic は低電圧方式のみサポートし、デバイスをプログラミング モードに設定するための高電圧 Vpp オプションをサポートしません(Vpp は Vdd 電源電圧を超えません。代わりに Vpp でテストパターンを使います)。
Program Speed	ターゲットをプログラミングする速度を「Low」、「Normal」、「High」から選択します。既定値は「Normal」です。プログラミングが失敗する場合、速度を下げると問題が解決する可能性があります。

8.7 周辺モジュールのフリーズ

周辺モジュールのリストから、コード実行停止時にフリーズさせるモジュールを選択します。利用可能な周辺モジュールはデバイスごとに異なります。

PIC12/16/18 MCU デバイスの場合

[Freeze on Halt]チェックボックスにチェックを入れると、実行停止時に全てのデバイス周辺モジュールがフリーズします。一部の周辺モジュールは freeze-on-halt 機能をサポートしておらず、デバッガから制御する事はできません。そのような周辺モジュールはチェックを入れても停止しません。

dsPIC、PIC24、PIC32 デバイスの場合

「Peripherals to Freeze on Halt」リスト内で、停止時にフリーズさせる周辺モジュールにチェックを入れます。プログラム実行が停止しても動作を継続させる周辺モジュールは、チェックを外します。停止させたい周辺モジュールがリスト内に存在しない場合、**[All Other Peripherals]**にチェックを入れます。一部の周辺モジュールは freeze-on-halt 機能をサポートしておらず、デバッガから制御する事はできません。そのような周辺モジュールはチェックを入れても停止しません。

[All Other Peripherals]を含む全ての周辺モジュールを選択するには、**[Check All]**をクリックします。**[All Other Peripherals]**を含む全ての周辺モジュールの選択を解除するには、**[Uncheck All]**をクリックします。

8.8 「Secure Segment」オプション カテゴリ

デバッガ ファームウェアを選択してロードします。

表 8-6. 「Secure Segment」オプション カテゴリ

Segments to be Programmed	プログラミングするセグメントを以下の中から 1 つを選択します。 1. Full Chip Programming (既定値) 2. Boot, Secure and General Segments 3. Secure and General Segments 4. General Segment Only

8.9 「Clock」オプション カテゴリ

このオプションは、ターゲット デバイスの高速内部 RC (FRC)クロックを使うために設定します。

表 8-7. 「Clock」オプション カテゴリ

Use FRC in Debug mode (dsPIC33F/E と PIC24F/H/E のみ)	デバッグ時に、アプリケーション向けに指定されたオシレータの代わりにデバイスの高速内部 RC (FRC)を使います。これはアプリケーション クロックが低速の場合に便利な機能です。このボックスにチェックを入れると、アプリケーションは低速で動作しますが、デバッグはより高速な FRC 速度で実行できます。 この設定を変更した後は、再プログラミングが必要です。 **Note:** フリーズされない周辺モジュールはデバッグ中に FRC 速度で動作します。

8.10 「Tool Pack Selection」オプション カテゴリ

デバッガ ファームウェアを選択してロードします。

表 8-8. 「Tool Pack Selection」オプション カテゴリ

Tool pack update options	「Use latest installed tool pack (recommended)」または「Use specific tool pack」を選択します。
Specifically selected version	使用するツールパックをクリックして選択します。クリックすると、バージョンを選択するための**[Select Tool pack]**ダイアログが開きます。

8.11 「Communication」オプション カテゴリ

デバイスとターゲットの通信に使うオプションを設定します。

表 8-9. 「Communication」オプション カテゴリ

Interface	インターフェイスを選択します。利用可能なオプションはプロジェクト デバイスに応じて異なります。
Speed (MHz)	選択したインターフェイスが対応可能な速度を入力します。

重要: UPDI を備えた少ピン数 AVR デバイスにおいて、RSTPINCFG コンフィグレーション ビットによって UPDI ピンが GPIO または RESET として設定されている場合、MPLAB PICkit Basic は UPDI インターフェイスを再有効化するための高電圧パルスを生成できません。これを行うには、MPLAB PICkit 5 等のツールが必要です。

8.12 「Event Recorder」オプション カテゴリ

イベントレコーダ向けのオプションを指定します。イベントレコーダの詳細は『MPLAB® X IDE User's Guide』(DS-50002027)または MPLAB X IDE ウェブヘルプを参照してください。

表 8-10. 「Event Recorder」オプション カテゴリ

Enable	チェックを入れるとイベントレコーダが有効になります。
SCVD Files	プロジェクト内で使用する SCVD ファイルを指定します。

9. ハードウェア仕様

以下では、MPLAB PICkit Basic インサーキット デバッガシステムのハードウェア仕様と電気的仕様について詳しく説明します。

9.1 USB コネクタ仕様

MPLAB PICkit Basic インサーキット デバッガは、USB Type-C®コネクタ(バージョン 2.0 に準拠)を介してホストコンピュータに接続します。USB Type-C®コネクタは本デバッガの上部にあります。

この USB インターフェイスを介してファームウェアをリロードできます。

システム電源は USB インターフェイスから供給されます。本デバッガは USB 仕様により「ハイパワーシステム」に分類されます。

Note: MPLAB PICkit Basic インサーキット デバッガは、USB Type-C®コネクタを介して給電されます。ターゲットボードには別の電源が必要です。MPLAB PICkit Basic からターゲットに給電する事はできません。

ケーブル長 – コンピュータとデバッガ間の接続用に、正常動作が得られる適切な長さのケーブルがデバッガキットに同梱されます。

セルフパワード ハブ - USB ハブを使う場合、バスパワードではなくセルフパワード タイプを使う必要があります。コンピュータ キーボード上の USB ポートでは、本デバッガに十分な電力を供給できません。

コンピュータの休止/省電力モード – USB 接続によるコンピュータとデバッガ間の通信を維持するため、コンピュータの省電力モード(休止モード等)は無効にする必要があります。

9.2 MPLAB PICkit Basic インサーキット デバッガの構成

本デバッガユニットは以下により構成されます。

1. 以下を備えた内部メイン基板:
   a. USB Type-C®コネクタ
   b. ターゲット接続用の 8 ピン SIL ヘッダ(0.100”ピッチ)
   c. デバッガの動作モードを示す 2 個の LED
   d. エマージェンシ リカバリ ユーティリティ専用のリカバリボタン

色分け信号ラベルを表記した基板ケース

デバッガユニットには以下も同梱されます。

1. 高品質 USB Type-C®**ハイスピード** ケーブル(長さ 1.5 m) - デバッガをコンピュータへ接続するために使用
2. 8 ピン SIL コネクタ - ケースの信号ラベルに対応した色分けワイヤ(17 cm)付き
3. 8 ピン-10 ピン ARM SWD アダプタボード - SWD 使用時に ARM ターゲットへの接続用に使用

9.2.1 回路基板の仕様

回路基板は以下の機能を備えています。

- Arm® Cortex®-M7 コアを備えた 32 ビット マイクロコントローラ
  - プログラムコード イメージを保持するためのメモリを内蔵(このイメージは内蔵フラッシュ デバイスをプログラミングするために使われる)
- USB Type-C®インターフェイス - 最大 480 Mbps に対応
- 2 個の LED (アクティブ、ステータス)

9.2.2 LED

MPLAB PICkit Basic は 2 個の単色 LED を備えています。アクティブ LED は緑色、ステータス LED は黄色です。PICkit Basic デバッガの起動時に緑 LED は点灯し黄 LED は消灯します。下表に、正常動作時とエラー時の LED の状態を示します。

表 9-1. 正常動作時の LED

LED	色	デバッガの状態
アクティブ LED 点灯	緑	電源接続済み、スタンバイ状態
ステータス LED 点灯(または点滅)	黄	デバッガはビジー、動作中

表 9-2. エラー時の LED

エラー	意味
ステータス LED が 3 秒間点灯	シリアル EEPROM へのアクセス中にブートローダに問題発生した
ステータス LED が 10 秒間点灯	ブートローダが API コマンドを処理できない
アクティブ LED とステータス LED が交互に点滅	ツール ファームウェアで実行時例外が発生
アクティブ LED とステータス LED が同期して点滅	ブートローダで実行時例外が発生

9.2.3 色分け信号ラベル

MIPS EJTAG	Cortex SWD	debugWIRE	PDI UPDI	AVR ISP	ICSP	番号	色
MCLR	MLCR				MLCR	1	橙
VIO_REF	VTG	VTG	VTG	VTG	VDD	2	赤
GND	GND	GND	GND	GND	GND	3	茶
TDO	SWO		DAT	MISO	DAT	4	黄
TCK	SWCLK	dW		SCK	CLK	5	緑
			CLK	RESET	AUX	6	青
TDI				MOSI		7	紫
TMS	SWDIO					8	灰

詳細は「3.3.1. ターゲット接続のピン配置」 を参照してください。

9.3 通信ハードウェア

デバッガとターゲット間の標準的な接続では、デバッガを直接ターゲットに接続します(「3.3. ターゲット接続」参照)。本デバッガは 8 ピン SIL ヘッダを備えています。ターゲットが 6 ピンコネクタを備えている場合、ピン 1 同士を合わせて差し込みます。

9.3.1 標準通信

標準通信は、デバッガとターゲット プロセッサ間のメイン インターフェイスを介して行います。このインターフェイスはターゲット デバイスのプログラミングと通信に必要な V_{DD} リセットライン、クロックライン、データラインを含みます。

クロックラインとデータラインは、下記の特性を備えたインターフェイスを介して接続する必要があります。

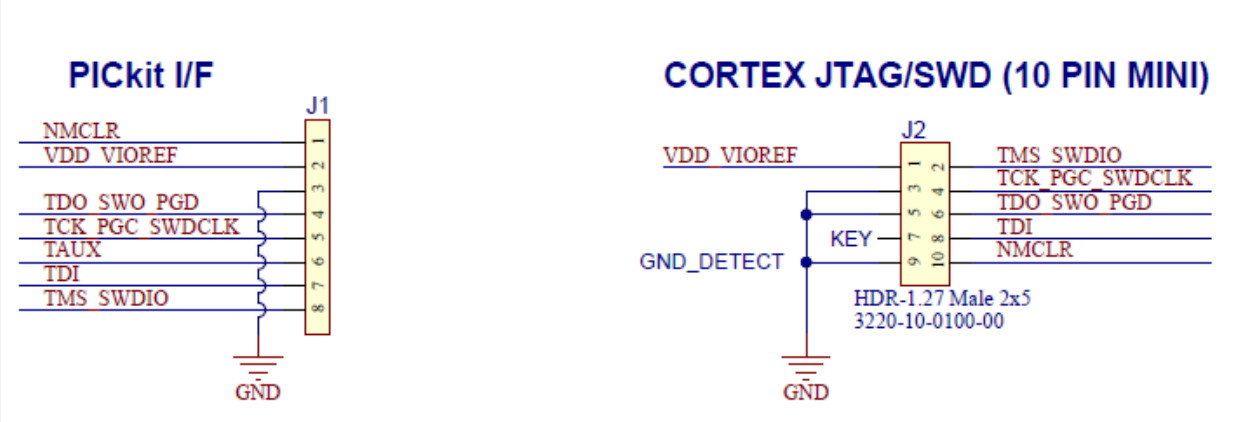
- クロックおよびデータ信号ラインは高インピーダンス モードである(たとえ MPLAB PICkit Basic インサーキット デバッガシステムが電源 OFF 中であっても高インピーダンスに維持)

表 9-3. 論理信号の電気的仕様

論理入力	$V_{IH} = V_{DD}$ x 0.7V (min.)			
	$V_{IL} = V_{DD}$ x 0.3V (max.)			
論理出力	V_{DD} = 5V	V_{DD} = 3V	V_{DD} = 2.3V	V_{DD} = 1.4V
	V_{OH} = 3.8V min.	V_{OH} = 2.4V min.	V_{OH} = 1.9V min.	V_{OH} = 1.0V min.
	V_{OL} = 0.55V max.	V_{OL} = 0.55V max.	V_{OL} = 0.3V max.	V_{OL} = 0.1V max.

9.3.2 Arm®/SWD アダプタボードの回路図

図 9-1. アダプタボードのピン配置



9.4 ターゲットボードに関する注意事項

ターゲットボードには、デバイスとアプリケーションの要件を満たす電源を供給する必要があります。

Note: デバイス データシートの電気的特性に記載している「絶対最大定格」を超える条件は、デバイスに恒久的な損傷を生じる可能性があります。これはストレス定格です。本書に示す動作条件外でのデバイス運用は想定していません。絶対最大定格条件を超えて長期間曝露させるとデバイスの信頼性に影響を及ぼす可能性があります。

本デバッガはターゲットの電圧を検出します。

デバッガとターゲット間の通信方法によっては、ターゲットボードの回路に関していくつかの注意点があります。詳細は以下を参照してください。

関連リンク

3.3.7.2. ICSP ターゲット接続回路
4.7.2.1. エミュレータの動作を妨げる回路

10. 改訂履歴

本書の各リビジョンでの改訂内容を以下に記載します。

Note: 文字「I」と「O」はリビジョンの表記に使われません(一部のフォントでは、これらの文字が数字「1」および「0」と見分けにくいため)。

10.1 リビジョン A (2025 年 2 月)

本書は初版です。

11. 用語集

絶対セクション(Absolute Section)
リンカで変更されない固定(絶対)アドレスを持つ GCC コンパイラのセクション。

絶対変数/関数(Absolute Variable/Function)
OCG コンパイラの@ address 構文を使って絶対アドレスに配置される変数または関数。

アクセスメモリ(Access Memory)
PIC18 のみ - PIC18 でバンクセレクト レジスタ(BSR)の設定に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を提供する。

アドレス(Address)
メモリ内の位置を一意に特定する値。

アルファベット文字(Alphabetic Character)
アルファベットの小文字と大文字の総称(a,b,･･･,z,A,B,･･･,Z)。

英数字(Alphanumeric)
アルファベット文字と 0～9 の 10 進数(0,1, …, 9)の数字の総称。

AND 条件ブレークポイント(ANDed Breakpoint)
プログラムの実行を停止するために設定する AND 条件(ブレークポイント 1 とブレークポイント 2 が同時に発生した場合のみプログラム実行を停止する)。AND 条件で実行が停止するのは、データメモリのブレークポイントとプログラムメモリのブレークポイントが同時に発生した場合のみ。

匿名構造体(Anonymous Structure)
16 ビット C コンパイラ – 無名の構造体。

PIC18 C コンパイラ - C 共用体のメンバーである無名の構造体。
匿名構造体のメンバーは、その構造体を包含している共用体のメンバーと同じようにアクセスできる。例えば以下のサンプルコードでは、hi と lo は共用体 caster に含まれる匿名構造体のメンバーである。

```
union castaway
int intval;
struct {
char lo; //accessible as caster.lo
char hi; //accessible as caster.hi
};
} caster;
```

ANSI
American National Standards Institute(米国規格協会)の略。米国における標準規格の策定と承認を行う団体。

アプリケーション(Application)
PIC®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/ライブラリ(Archive/Library)
アーカイブ/ライブラリは、再配置可能なオブジェクト モジュールの集まり。複数のソースファイルをオブジェクト ファイルにアセンブルした後、アーカイバ/ライブラリアンを使ってこれらオブジェクト ファイルを 1 つのアーカイブ/ライブラリ ファイルにまとめると生成される。

アーカイブ/ライブラリをオブジェクト モジュールや他のアーカイブ/ライブラリとリンクすると、実行コードが生成される。

ASCII

American Standard Code for Information Interchange の略。7 桁の 2 進数で 1 つの文字を表現する文字セットエンコード方式。大文字、小文字、数字、記号、制御文字等を含む。

アセンブリ/アセンブラ(Assembly/Assembler)

アセンブリとは、2 進数のマシンコードをシンボル表現で記述したプログラミング言語。アセンブラとは、アセンブリ言語のソースコードをマシンコードに変換する言語ツール。

割り当てセクション(Assigned Section)

リンカのコマンドファイルで特定のターゲットメモリ ブロックに割り当てられた GCC コンパイラのセクション。

非同期(Asynchronously)

複数のイベントが同時には発生しない事。一般に、プロセッサ実行中の任意の時点で発生する割り込みに言及する際に使う。

非同期スティミュラス(Asynchronous Stimulus)

シミュレータ デバイスへの外部入力をシミュレートするために生成されるデータ。

属性(Attribute)

GCC の C 言語プログラムの変数または関数の特徴を表す情報で、マシン固有の特性を記述する目的で使う。

属性(セクション属性) (Attribute, Section)

「executable」、「readonly」、「data」等、セクションの特徴を表す情報。アセンブラの.section ディレクティブでフラグとして指定できる。

AVR MCU

Microchip 社の全ての AVR 8 ビット マイクロコントローラ ファミリの総称

2 進数(Binary)

0 と 1 の数字を使う、2 を底とした記数法。一番右の桁が 1 の位、次の桁が 2 の位、その次の桁が 2^2 = 4 の位を表す。

ブックマーク(Bookmark)

ファイル内の特定の行に簡単な操作でアクセスできるようにする機能。

[Editor]ツールバーの[Toggle Bookmarks]を選択してブックマークを追加または削除する。このツールバーの他のアイコンをクリックすると、次または前のブックマークに移動する。

C/C++

C 言語は、簡潔な表現、現代的な制御フローとデータ構造、豊富に用意された演算子等を特長とする汎用プログラミング言語。C++とは、C 言語のオブジェクト指向バージョン。

校正メモリ(Calibration Memory)

PIC MCU の内蔵 RC オシレータやその他の周辺モジュールの校正値を格納するための特殊機能レジスタ。

中央演算処理装置(Central Processing Unit)

デバイス内で、実行する正しい命令をフェッチし、デコードして実行する装置。

必要に応じて、算術論理演算装置(ALU)と組み合わせて命令実行を完了する。プログラムメモリのアドレスバス、データメモリのアドレスバス、スタックへのアクセスを制御する。

クリーン(Clean)
クリーンする事により、アクティブなプロジェクトのオブジェクト ファイル、HEX ファイル、デバッグファイル等、全ての中間ファイルが削除される。これらのファイルは、プロジェクトのビルド時に他のファイルから再構築される。

COFF
Common Object File Format の略。このフォーマットのオブジェクト ファイルは、マシンコードの他、デバッグ等に関する情報を含む。

コマンドライン インターフェイス(Command Line Interface)
プログラムとユーザのやり取りをテキストの入出力だけで行う方法。

コンパイルド スタック(Compiled Stack)
コンパイラが管理するメモリの領域で、この領域内で変数に静的に空間を割り当てる。ターゲット デバイス上にソフトウェア スタックまたはハードウェア スタックのメカニズムを効率的に実装できない場合、コンパイルド スタックがソフトウェア スタックまたはハードウェア スタックに置き換わる。

コンパイラ(Compiler)
高級言語で記述されたソースファイルをマシンコードに変換するプログラム。

条件付きアセンブリ(Conditional Assembly)
アセンブリ言語で、ある特定の式のアセンブル時の値に基づいて含まれたり除外されたりするコード。

条件付きコンパイル(Conditional Compilation)
プログラムの一部を、プリプロセッサ ディレクティブで指定した特定の定数式が真の場合のみコンパイルする事。

コンフィグレーション ビット(Configuration Bit)
PIC MCU と dsPIC DSC の動作モードを設定するために書き込む専用ビット。コンフィグレーション ビットは事前プログラミングされている場合とされていない場合がある。

定数(Constant)
C コードの#define ディレクティブまたはアセンブリの.equ ディレクティブによる定義等、即値を表す。

制御ディレクティブ(Control Directive)
アセンブリ言語コード内で使うディレクティブで、指定した式のアセンブル時の値に基づいてコードを含めるか除外するかを決定する。

CPU
「中央演算処理装置」参照。

相互参照ファイル(Cross Reference File)
シンボルテーブルとそのシンボルを参照するファイルリストを参照するファイル。シンボルが定義されている場合、リストの最初のファイルがシンボル定義の位置となる。残りのファイルはシンボルへの参照を含む。

データ ディレクティブ(Data Directive)
アセンブラが行うプログラムメモリまたはデータメモリの割り当てを制御するディレクティブ。データ項目をシンボル(意味のある名前)で参照する手段として使う。

データメモリ(Data Memory)
Microchip 社の MCU と DSC では、データメモリ(RAM)は汎用レジスタ(GPR)と特殊機能レジスタ(SFR)で構成される。EEPROM データメモリを内蔵したデバイスもある。

デバッグ/デバッガ(Debug/Debugger)
「ICE/ICD」参照。

デバッグ情報(Debugging Information)
コンパイラとアセンブラでこのオプションを選択すると、アプリケーション コードのデバッグに使える各種レベルの情報を出力できる。デバッグ オプションの選択の詳細はコンパイラまたはアセンブラのマニュアルを参照。

非推奨の機能(Deprecated Feature)
後方互換性確保のためにサポートしているだけで現在は使っておらず、いずれ廃止になる事が決まっている機能。

デバイス プログラマ(Device Programmer)
マイクロ コントローラ等、電気的に書き込み可能な半導体デバイスをプログラミングするためのツール。

デジタル信号処理/デジタルシグナル プロセッサ(Digital Signal Processing/Digital Signal Processor)
デジタル信号処理(DSP)とは、デジタル信号をコンピュータで処理する事。通常は、アナログ信号(音声または画像)をデジタル形式に変換(サンプリング)して処理する事をいう。デジタルシグナル プロセッサとは、信号処理用に設計されたマイクロプロセッサの事。

ディレクティブ(Directive)
言語ツールの動作を制御するためにソースコードに記述する命令文。

ダウンロード(Download)
ホストから別のデバイス(例: エミュレータ、プログラマ、ターゲットボード)にデータを送信する事。

dsPIC DSC
Microchip 社のデジタル信号処理能力を備えたマイクロコントローラ(dsPIC®デジタルシグナル コントローラ(DSC))ファミリの総称。

DWARF
Debug With Arbitrary Record Format の略。ELF ファイルのデバッグ情報フォーマット。

EEPROM
Electrically Erasable Programmable Read Only Memory の略。電気的に消去可能なタイプの PROM。データの書き込みと消去をバイト単位で行う。EEPROM は電源を OFF にしても内容を保持する。

ELF
Executable and Linking Format の略。この形式のオブジェクト ファイルはマシンコードを含む。デバッグその他の情報は DWARF で指定する。ELF/DWARF の方が COFF よりも最適化したコードのデバッグに適している。

エミュレーション/エミュレータ(Emulation/Emulator)
「ICE/ICD」参照。

エンディアン(Endianness)
マルチバイト オブジェクトにおけるバイトの並び順。

環境(Environment)
MPLAB PM3 - デバイスのプログラミングに関する設定ファイルを保存したフォルダ。このフォルダを SD/MMC カードに転送できる。

エピローグ(Epilogue)
コンパイラで生成したコードのうち、スタック領域の割り当て解除、レジスタの復帰、ランタイムモデルで指定したその他のマシン固有の要件を実行するコード部分。関数のユーザコードの後、関数リターンの直前にエピローグを実行する。

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

エラー/エラーファイル(Error/Error File)
プログラムの処理を継続できない問題が発生するとエラーとして報告される。可能な場合、エラーは問題が発生したソースファイル名と行番号を特定する。エラーファイルは、言語ツールから出力されたエラーメッセージと診断結果を保存する。

イベント(Event)
アドレス、データ、パスカウント、外部入力、サイクルタイプ (フェッチ、R/W)、タイムスタンプ等、バスサイクルを記述したもの。トリガ、ブレークポイント、割り込みを記述するために使う。

実行可能コード(Executable Code)
読み込んで実行できる形式のソフトウェア。

エクスポート(Export)
MPLAB X IDE から標準フォーマットでデータを外部に出力する事。

式(Expression)
算術演算子または論理演算子で区切った定数または記号の組み合わせ。

拡張マイクロ コントローラ モード(Extended Microcontroller Mode)
拡張マイクロコントローラ モードでは、内蔵プログラムメモリと外部メモリの両方が利用できる。プログラムメモリのアドレスが PIC18 の内部メモリ空間より大きい場合、自動的に外部メモリの実行に切り換わる。

拡張モード(Extended Mode) (PIC18 MCU)
コンパイラの動作モードの 1 つ。拡張命令(ADDFSR、ADDULNK、CALLW、MOVSF、MOVSS、PUSHL、SUBFSR、SUBULNK)とリテラル オフセットによるインデックス アドレス指定を利用できる。

外部ラベル(External Label)
外部リンケージを持つラベル。

外部リンケージ(External Linkage)
関数または変数が、それを定義したモジュールの外部から参照できる場合、外部リンケージを持つという。

外部シンボル(External Symbol)
外部リンケージを持つ識別子のシンボル。参照の場合と定義の場合がある。

外部シンボル解決(External Symbol Resolution)
リンカが全ての入力モジュールの外部シンボル定義を 1 つにまとめ、全ての外部シンボル参照を解決しようとするプロセス。外部シンボル参照に対応する定義が存在しない場合、リンカエラーとなる。

外部入力ライン(External Input Line)
外部信号に基づいてイベントを設定するための外部入力信号ロジックプローブ ライン(TRIGIN)。

外部 RAM (External RAM)
デバイス外部にある、読み書き可能なメモリ。

致命的エラー(Fatal Error)
コンパイルがただちに停止するようなエラー。エラーの発生後はメッセージも出力されない。

ファイルレジスタ(File Register)
汎用レジスタ(GPR)と特殊機能レジスタ(SFR)からなる内蔵のデータメモリ。

フィルタ(Filter)
トレース ディスプレイまたはデータファイルにどのデータを含めるか/除外するかを選択するもの。

フィックスアップ(Fixup)
リンカによる再配置後にオブジェクト ファイルのシンボル参照を絶対アドレスに置き換える処理。

フラッシュ (Flash)
データの書き込みと消去をバイト単位ではなくブロック単位で行えるタイプの EEPROM。

FNOP
Forced No Operation の略。Forced NOP サイクルは、2 サイクル命令の 2 サイクル目で発生する。PIC マイクロコントローラのアーキテクチャはパイプライン構造となっており、現在の命令を実行中に物理アドレス空間の次の命令をプリフェッチする。しかし、現在の命令でプログラム カウンタが変化した場合、プリフェッチした命令は明示的に無視され、Forced NOP サイクルが発生する。

フレームポインタ(Frame Pointer)
スタックベースの引数とスタックベースのローカル変数の境界となるスタック番地を指し示すポインタ。ここを基準にすると、現在の関数のローカル変数やその他の値に容易にアクセスできる。

フリー スタンディング(Free-Standing)
複素数型を使っておらず、ライブラリ(ANSI C89 規格第 7 節)で規定する機能の使用が標準ヘッダ(`<float.h>`、`<iso646.h>`、`<limits.h>`、`<stdarg.h>`、`<stdbool.h>`、`<stddef.h>`、`<stdint.h>`)の内容のみに限定されている厳密な規格合致プログラムを受理する処理系。

GPR
General Purpose Register(汎用レジスタ)の略。デバイスのデータメモリ(RAM)のうち、汎用目的に使える部分。

Halt
プログラム実行を停止する事。Halt を実行する事は、ブレークポイントで停止する事と同じ。

ヒープ(Heap)
動的メモリ割り当てに使うメモリ空間。メモリブロックの割り当てと解放は実行時に任意の順序で行う。

HEX コード/HEX ファイル(Hex Code/Hex File)
HEX コードは、実行可能な命令を 16 進数形式のコードで保存したもの。Hex コードは Hex ファイルに保存される。

16 進数(Hexadecimal)
0～9 の数字と A～F(または a～f)のアルファベットを使った、16 を底とした記数法。16 進数の A～F は、10 進数の 10～15 を表す。一番右の桁が 1 の位、次の桁が 16 の位、その次の桁が 16^2 = 256 の位を表す。

高級言語(High Level Language)
プログラムを記述するための言語で、プロセッサから見てアセンブリよりも遠い位置関係にあるもの。

ICE/ICD
インサーキット エミュレータ/インサーキット デバッガの略。ターゲット デバイス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基板にマウントし、インサーキット エミュレータまたはデバッガによるデバッグ用に使う。

ICSP
In-Circuit Serial Programming の略。Microchip 社製の組み込みデバイスを、シリアル通信を利用して最小限のデバイスピンでプログラミングする方法。

IDE
Integrated Development Environment の略。MPLAB X 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 X IDE にデータを取り込む事。

初期化済みデータ(Initialized Data)
初期値を指定して定義されたデータ。C では、

```
int myVar=5;
```

として定義した変数は初期化済みデータセクションに格納する。

命令セット(Instruction Set)
特定のプロセッサが理解できるマシン語命令の集合。

命令(Instruction)
CPU に対して特定の演算を実行するように指示するビット列。演算の対象となるデータを含める事もできる。

内部リンケージ(Internal Linkage)
関数または変数が、それを定義したモジュールの外部から参照できない場合、内部リンケージを持つという。

国際標準化機構(International Organization for Standardization)
コンピューティングや通信を始めとする、多くのテクノロジとビジネス関連の標準規格の策定を行っている団体。一般的に ISO と呼ぶ。

割り込み(Interrpt)
CPU に対する信号の一種。この信号が発生すると、現在動作中のアプリケーションの実行を一時停止し、制御を割り込みサービスルーチン(ISR)に渡してイベントを処理する。ISR の実行が完了すると、通常のアプリケーションの実行を再開する。

割り込みハンドラ(Interrupt Handler)
割り込み発生時に専用のコードを実行するルーチン。

割り込みサービス要求(IRQ: Interrupt Service Request)
プロセッサの通常の命令実行を一時的に停止し、割り込みハンドラルーチンの実行開始を要求するイベント。プロセッサによっては複数の割り込み要求イベントを持ち、優先度の異なる割り込みを処理できるものもある。

割り込みサービスルーチン(ISR: Interrupt Service Routine)
言語ツールの場合: 割り込みを処理する関数。

MPLAB X IDE の場合: 割り込みが発生すると実行されるユーザ作成コード。通常、発生した割り込みの種類によってプログラムメモリ内の異なる位置のコードを実行する。

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

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

レイテンシ(Latency)
イベントが発生してからその応答までの時間の長さ。

ライブラリ/ライブラリアン(Library/Librarian)
「アーカイブ/ライブラリ」参照。

リンカ(Linker)
オブジェクト ファイルとライブラリを結合し、モジュール間の参照を解決して実行可能コードを生成する言語ツール。

リンカスクリプト ファイル(Linker Script File)
リンカのコマンドファイル。リンカのオプションを定義し、ターゲット プラットフォームで利用可能なメモリを記述する。

リスティング ディレクティブ(Listing Directive)
アセンブラのリスティング ファイルのフォーマットを制御するディレクティブ。タイトルや改ページ指示等、リスティング ファイルに関する各種の設定を行う。

リスティング 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とラベルには、ENDM マクロ以降はアクセスできない。

マシンコード(Machine Code)
コンピュータ プログラムをプロセッサが実際に読み出して解釈できる形式で表現したもの。2 進数のマシンコードで記述されたプログラムは、マシン命令のシーケンス(命令間にデータを挟む事もある)から成る。ある特定のプロセッサで使える全ての命令の集合を「命令セット」という。

マシン語(Machine Language)
ある CPU が翻訳を必要とせず実行できる命令の集合。

マクロ(Macro)
マクロ命令。一連の命令シーケンスを短い名前で表現した命令。

マクロ ディレクティブ(Macro Directive)
マクロ定義の中で実行とデータ割り当てを制御するディレクティブ。

make ファイル(Makefile)
プロジェクトの Make に関する指示をファイルにエクスポートしたもの。このファイルは、MPLAB X IDE 以外の環境で make コマンドを実行してプロジェクトをビルドする際に使う。

make プロジェクト(Make Project)
アプリケーションを再ビルドするコマンド。前回の完全なコンパイル後に変更されたソースファイルのみを再コンパイルする。

MCU
Microcontroller Unit の略。マイクロコントローラの事。「µC」と表記する事もある。

メモリモデル(Memory Model)
C コンパイラの場合、アプリケーションで利用可能なメモリを表現したもの。PIC18C コンパイラの場合、プログラムメモリを指し示すポインタのサイズに関する規定を記述したもの。

メッセージ(Message)
言語ツールの動作に問題が発生した事を知らせる文字列。メッセージが表示されても処理は停止しない。

マイクロコントローラ(Microcontroller)
CPU、RAM、プログラムメモリ、I/O ポート、タイマ等、多くの機能を統合したチップ。

マイクロコントローラ モード(Microcontroller Mode)
PIC18MCU で設定可能なプログラムメモリ構成の 1 つ。マイクロコントローラ モードでは、内部実行のみを許可する。つまり、マイクロコントローラ モードでは内蔵プログラムメモリしか使えない。

マイクロプロセッサ モード(Microprocessor Mode)
PIC18MCU で設定可能なプログラムメモリ構成の 1 つ。マイクロプロセッサ モードでは、内蔵プログラムメモリは使わない。プログラムメモリ全体を外部にマッピングする。

ニーモニック(Mnemonic)
マシンコードと 1 対 1 で対応したテキスト命令。オペコードとも呼ぶ。

モジュール(Module)
プリプロセッサ ディレクティブ実行後の前処理済みのソースファイル出力。翻訳単位とも呼ぶ。

MPLAB® X IDE
Microchip 社の統合開発環境(Integrated Development Environment)。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 X シミュレータ
Microchip 社のシミュレータ。Microchip 社の MCU、DSC、MPU デバイスに対応する。

MPLAB XC C コンパイラ
Microchip 社の C および C++コンパイラ ファミリ。これには MPLAB XC8 C コンパイラ(8 ビット PIC および AVR デバイス向け)、MPLAB XC16 C コンパイラ(16 ビット PIC デバイス向け)、MPLAB XC-DSC C コンパイラ(DSC デバイス向け)、MPLAB XC32 C/C++コンパイラ(32 ビット PIC および SAM デバイス向け)が含まれる。

MPLAB Xpress IDE
クラウド上の Microchip 社統合開発環境。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。

MPU
マイクロプロセッサ ユニット。マイクロプロセッサの略。

MRU
Most Recently Used の略。最近使ったファイルおよびウィンドウの事。メインのプルダウン メニューで選択できる。

ネイティブ データサイズ(Native Data Size)
ネイティブ トレースの場合、**[Watches]**ウィンドウで使う変数のサイズは選択したデバイスのデータメモリと同じサイズ(PIC18 の場合は同じバイトサイズ、16 ビットデバイスの場合は同じワードサイズ)である必要がある。

入れ子の深さ(Nesting Depth)
マクロに他のマクロを入れ込める階層の数。

ノード(Node)
プロジェクトの構成要素。

非拡張モード(Non-Extended Mode) (PIC18 MCU)
コンパイラの動作モードの 1 つ。拡張命令もリテラル オフセットによるインデックス アドレス指定も使わない。

非リアルタイム(Non Real Time)
ブレークポイントで停止中、シングルステップ命令の実行中、シミュレータ モードで動作中のプロセッサの状態を指す。

不揮発性ストレージ(Non-Volatile Storage)
電源を OFF にしても内容が失われないストレージ デバイス。

NOP
No Operation の略。実行してもプログラム カウンタが進むだけで何も動作を行わない命令。

オブジェクト コード/オブジェクト ファイル(Object Code/Object File)
オブジェクト コードとは、アセンブラまたはコンパイラで生成されるマシンコードの事。オブジェクト ファイルとは、マシンコードを格納したファイル。デバッグ情報を含む事もある。そのまま実行できるものと、他のオブジェクト ファイル(例: ライブラリ)とリンクしてから完全な実行プログラムを生成する再配置可能形式のものがある。

オブジェクト ファイル ディレクティブ(Object File Directive)
オブジェクト ファイル作成時にのみ使うディレクティブ。

8 進数(Octal)
0～7 の数字のみを使う、8 を底とした記数法。一番右の桁が 1 の位、次の桁が 8 の位、その次の桁が 8^2 = 64 の位を表す。

オフチップメモリ(Off-Chip Memory)
PIC18 で選択できるメモリオプション。ターゲットボードのメモリを使うか、または全てのプログラム メモリをエミュレータから供給する。

Options > **Development Mode** から**[Memory]**タブにアクセスし、ダイアログ ボックスでオフチップメモリを選択する。

オペコード(Opcode)
Operational Code の略。「ニーモニック」参照。

演算子(Operator)
定義可能な式を構成する際に使う「+」や「-」等の記号。各演算子に割り当てられた優先順位に基づいて式を評価する。

OTP (One-Time-Programmable)
One Time Programmable の略。パッケージに窓のない EPROM デバイス。EPROM を消去するには紫外線照射が必要なため、パッケージに窓のあるデバイスしか消去できない。

パスカウンタ(Pass Counter)
イベント(特定のアドレスの命令を実行する等)が発生するたびに値をデクリメントするカウンタ。パスカウンタの値がゼロになると、イベントの条件を満たす。パスカウンタはブレークロジック、トレースロジック、複合トリガダイアログの任意のシーケンシャル イベントに割り当てられる。

PC
パーソナル コンピュータまたはプログラム カウンタの略。

ホスト PC (PC Host)
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パーソナル コンピュータ。

永続データ(Persistent Data)
クリアも初期化もされないデータ。デバイスをリセットしてもアプリケーションがデータを保持できるようにするために使う。

ファントムバイト(Phantom Byte)
dsPIC アーキテクチャで、24 ビット命令ワードを 32 ビット命令ワードとみなして扱う場合に使う未実装バイト。dsPIC の HEX ファイルに見られる。

PIC MCU
Microchip 社の全ての PIC 8/16/32 ビット マイクロコントローラ ファミリの総称。

プラグイン(Plug-in)
MPLAB IDE/MPLAB X IDE では、標準コンポーネントにプラグイン モジュールを追加する事で各種のソフトウェア/ハードウェア ツールに対応する。一部のプラグインツールは、[Tools]メニューから利用できる。

ポッド(Pod)
インサーキット エミュレータまたはデバッガの筐体。丸型の場合パック(Puck)と呼ぶ事もある。あるいはプローブ(Probe)とも呼ぶが、「論理プローブ」と混同せぬよう注意が必要。

パワーオン リセット エミュレーション(Power-on-Reset Emulation)
データ RAM 領域にランダムな値を書き込んで、初回電源投入時の RAM の非初期化値をシミュレートするソフトウェア無作為化処理。

プラグマ(Pragma)
特定のコンパイラにとって意味を持つディレクティブ。一般に、実装で定義した情報をコンパイラに伝達するために使う。

優先順位(Precedence)
式の評価順を定義した規則。

量産プログラマ(Production Programmer)
デバイスを高速にプログラミングできるようにリソースを強化したプログラマ。各種電圧レベルでのプログラミングに対応し、プログラミング仕様に完全に準拠している。量産環境では応用回路が組み立てラインにとどまる時間をなるべく短くする必要があるため、デバイスへの書き込み時間の短縮が特に重要である。

プロファイル(Profile)
MPLAB X 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。

Program Memory
デバイス内で命令が保存されるメモリ空間。デバッガ、エミュレータ、シミュレータにダウンロードしたターゲット アプリケーションのファームウェアを格納するメモリ空間もプログラムメモリと呼ぶ。

プロジェクト(Project)
アプリケーションのビルドに必要なファイル(例: ソースコード、リンカスクリプト ファイル)一式と、各種ビルドツールやビルドオプションとの関連付けをまとめたもの。

プロローグ(Prologue)
コンパイラで生成したコードのうち、スタック領域の割り当て、レジスタの退避、ランタイムモデルで指定したその他のマシン固有の要件を実行するコード部分。プロローグは、関数のユーザコードの前に実行する。

プロトタイプ システム(Prototype System)
ユーザのターゲット アプリケーションまたはターゲットボードの事。

Psect
GCC のセクションに相当する OCG の用語。プログラム セクション(program section)の略語。リンカが 1 つのまとまりとして処理するコードまたはデータのブロック。

PWM 信号(PWM Signal)
パルス幅変調(Pulse Width Modulation)信号。一部の PIC MCU は周辺モジュールとして PWM を内蔵している。

修飾子(Qualifier)
パスカウンタで使ったり、複合トリガにおける次の動作前のイベントとして使ったりするアドレスまたはアドレスレンジ。

基数(Radix)
アドレスを指定する際の記数法(16 進法、10 進法)の底。

RAM
Random Access Memory の略。データメモリ。任意の順にメモリ内の情報にアクセスできる。

生データ(Raw Data)
あるセクションに関連付けられたコードまたはデータを 2 進数で表現したもの。

読み出し専用メモリ(Read Only Memory)
恒久的に保存されているデータへの高速アクセスが可能なメモリ ハードウェア。ただし、データの追加や変更は不可。

リアルタイム(Real Time)
インサーキット エミュレータまたはデバッガが Halt 状態から解放されると、プロセッサの実行はリアルタイム モードとなり、通常のチップと同じ挙動をする。リアルタイム モードでは、エミュレータのリアルタイム トレースバッファが有効になり、選択した全てのサイクルを常時キャプチャする。また、全てのブレークロジックが有効になる。インサーキット エミュレータまたはデバッガでは、有効なブレークポイントで停止するか、またはユーザが実行を停止するまでプロセッサはリアルタイムで動作する。

シミュレータでは、ホスト CPU でシミュレート可能な最大速度でマイクロ コントローラの命令を実行する事をリアルタイムと呼ぶ。

再帰呼び出し(Recursive Call)
直接または間接的に自分自身を呼び出す関数。

再帰(Recursion)
定義した関数またはマクロがそれ自身を呼び出す事。再帰マクロを作成する際は、再帰から抜けずに無限ループとなりやすいため注意が必要。

再入可能(Re-entrant)
1 つの関数を複数呼び出して同時に実行できる事。直接または間接再帰、あるいは割り込み処理中の実行によって起こる事がある。

緩和(Relaxation)
ある命令を、機能が同じでよりサイズの小さい命令に変換する事。コードサイズを抑えるために便利である。現在、MPLAB XC16 と MPLAB XC-DSC は CALL 命令を RCALL 命令へ緩和する事ができる。
この変換は、現在の命令から+/- 32k 命令ワード以内にあるシンボルを呼び出す場合に行われる。

再配置可能(Relocatable)
アドレスがメモリの固定番地に割り当てられていないオブジェクト。

再配置可能セクション(Relocatable Section)
16 ビットアセンブラ - アドレスが固定されていない(絶対アドレスでない)セクション。再配置可能セクションには、再配置と呼ばれるプロセスでリンカがアドレスを割り当てる。

再配置(Relocation)
リンカが絶対アドレスを再配置可能セクションに割り当てる事。再配置可能セクション内の全てのシンボルを新しいアドレスに更新する。

ROM
Read Only Memory の略。プログラムメモリ。内容を変更する事ができないメモリ。

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)
アプリケーションの実行中に変数の変化を**[Watches]**ウィンドウに表示される。ランタイム ウォッチの設定方法は各ツールの関連文書を参照。ランタイム ウォッチをサポートしていないツールもある。

SAM MCU/MPU
Microchip 社の全ての SAM 32 ビット マイクロコントローラおよびマイクロプロセッサ ファミリの総称。

シナリオ(Scenario)
MPLAB SIM シミュレータでスティミュラス制御を具体的に設定したもの。

セクション(Section)
OCG の psect に相当する GCC の用語。リンカが 1 つのまとまりとして処理するコードまたはデータのブロック。

セクション属性(Section Attribute)
GCC のセクションの特徴を表す情報(例: access セクション)。

シーケンス ブレークポイント(Sequenced Breakpoint)
シーケンスで発生するブレークポイント。ブレークポイントのシーケンスはボトムアップ方式で実行される。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。

SQTP (Serialized Quick Turn Programming)
デバイス プログラマでマイクロ コントローラをプログラミングする際に、各デバイスに異なるシリアル番号を書き込めるようにする機能。この番号はエントリコード、パスワード、ID 番号として使える。

シェル(Shell)
MPASM アセンブラにおいて、マクロアセンブラへの入力を行うためのプロンプト インターフェイス。MPASM アセンブラには DOS 用シェルと Windows 用シェルの 2 種類がある。

シミュレータ(Simulator)
デバイスの動作をモデル化するソフトウェア プログラム。

シングルステップ(Single Step)
コードを 1 命令ずつ実行するコマンド。1 命令を実行するたびにレジスタ ウィンドウ、ウォッチ変数、ステータスの表示が更新されるため命令実行の解析およびデバッグに役立つ。C コンパイラのソースコードもシングルステップで実行できるが、その場合は 1 命令ずつ実行するのではなく、高級言語の C で記述されたコードの 1 行から生成される全てのアセンブリレベル命令をシングルステップで実行する。

スキュー(Skew)
命令実行に対応する情報は、異なる複数のタイミングでプロセッサバスに表れる。
例えば、実行されるオペコードは直前の命令の実行時にフェッチとしてバスに表れる。ソースデータのアドレス、値、デスティネーション データのアドレスは、オペコードが実際に実行される時にバスに表れる。デスティネーション データの値は次の命令の実行時にバスに表れる。トレースバッファは、1 インスタンスでバス上に存在する情報をキャプチャする。従って、トレースバッファの 1 エントリには 3 つの命令の実行情報が含まれる。1 つの命令実行で、ある情報から次の情報までにキャプチャされるサイクル数をスキューと呼ぶ。

スキッド(Skid)
ハードウェア ブレークポイントを使ってプロセッサを停止する場合、ブレークポイント以降の命令を実行してプロセッサが停止する事がある。ブレークポイントの後に実行する命令の数をスキッドと呼ぶ。

ソースコード(Source Code)
人間が記述したコンピュータ プログラム。プログラミング言語で記述されたソースコードは、マシンコードに変換して実行するか、またはインタプリタで実行される。

ソースファイル(Source File)
ソースコードを記述した ASCII テキストファイル。

特殊機能レジスタ(SFR)
I/O プロセッサ機能、I/O ステータス、タイマ等の各種モードや周辺モジュールを制御するレジスタ専用に使うデータメモリ(RAM)領域。

SQTP
「Serialized Quick Turn Programming」参照。

ハードウェア スタック(Stack, Hardware)
PIC MCU で関数を呼び出す時に戻りアドレスを格納する場所。

ソフトウェア スタック(Stack, Software)
アプリケーションが戻りアドレス、関数パラメータ、ローカル変数を保存するのに使うメモリ。このメモリはプログラムでの命令の実行時に動的に割り当てられる。これによって、再入可能な関数の呼び出しが可能になる。

スタック、コンパイルド(Stack, Compiled)
コンパイラが管理し割り当てるメモリの領域で、この領域内で変数に静的に空間を割り当てる。ターゲット デバイス上にソフトウェア スタックのメカニズムを効率的に実装できない場合、ソフトウェア スタックがコンパイルド スタックに置き換わる。このメカニズムでは、関数は再入可能ではなくなる。

静的 RAM (SRAM) (Static RAM、SRAM)
Static Random Access Memory の略。ターゲットボード上の読み書き可能なプログラムメモリ。リフレッシュ動作は不要。

ステータスバー(Status Bar)
[MPLAB X IDE]ウィンドウの一番下にあるバーで、カーソル位置、開発モードとデバイス、アクティブなツールバー等に関する情報が表示される。

ステップイントゥ(Step Into)
Single Step と同じコマンド。Step Over とは異なり、Step Into では CALL 命令が呼び出すサブルーチン内もステップ実行する。

ステップオーバー(Step Over)
Step Over を実行すると、サブルーチン内をステップ実行せずにコードをデバッグできる。Step Over では、CALL 命令があると CALL の次の命令にブレークポイントが設定される。何らかの理由によりサブルーチンが無限ループになる等正しく戻らない場合、次のブレークポイントには到達しない。
CALL 命令の処理以外は、Step Over コマンドと Single Step コマンドは同じ。

ステップアウト(Step Out)
現在ステップ実行中のサブルーチンから抜け出すためのコマンド。このコマンドを実行すると、サブルーチンの残りのコードを全て実行し、サブルーチンの戻りアドレスで実行が停止する。

スティミュラス(Stimulus)
シミュレータへの入力。すなわち外部信号に対する応答をシミュレートするために生成されるデータ。通常、テキストファイルにアクションのリストとしてこのデータを記述する。スティミュラスの種類には非同期、同期(ピン)、クロック動作、レジスタがある。

ストップウォッチ(Stopwatch)
実行サイクルを計測するためのカウンタ。

記憶域クラス(Storage Class)
指定されたオブジェクトに対応する記憶場所の持続期間を決定する。

記憶域修飾子(Storage Qualifier)
宣言されるオブジェクトの特別な属性を示す(例: const)。

シンボル(Symbol)
プログラムを構成する各種の要素を記述する汎用のメカニズム。関数名、変数名、セクション名、ファイル名、struct/enum/union タグ名等がある。MPLAB X IDE では、主に変数名、関数名、アセンブリラベルをシンボルと呼ぶ。リンク実行後は、シンボルの値はメモリ内の値となる。

絶対シンボル(Symbol, Absolute)
メモリ内の特定アドレスに配置されたシンボル (例: `int scanMode _at(0x200);`)。

システム ウィンドウ コントロール(System Window Control)
ウィンドウと一部のダイアログの左上隅にあるコントロール。通常、このコントロールをクリックすると、[最小化]、[最大化]、[閉じる]等のメニュー項目がポップアップ表示される。

ターゲット(Target)
ユーザ ハードウェアの事。

ターゲット アプリケーション(Target Application)
ターゲットボードに読み込んだソフトウェア。

ターゲットボード(Target Board)
ターゲット アプリケーションを構成する回路とプログラミング可能デバイス。

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 ボード上の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 ファイルに保存する。

ツールバー(Toolbar)
各種機能を実行するためのアイコンを縦または横に並べたもの。

トレース(Trace)
プログラム実行を記録するエミュレータまたはシミュレータの機能。エミュレータはプログラム実行をトレースバッファに記録し、これをトレース ウィンドウにアップロードする。

トレースメモリ(Trace Memory)
エミュレータが内蔵するトレース用のメモリ。トレースバッファとも呼ばれる。

トレースマクロ(Trace Macro)
エミュレータ データからのトレース情報を提供するマクロ。これはソフトウェア トレースのため、トレースを利用するにはマクロをコードに追加し、コードを再コンパイルまたは再アセンブルし、ターゲット デバイスにこのコードをプログラミングする必要がある。

トリガ出力(Trigger Output)
任意のアドレスまたはアドレスレンジで生成でき、トレースとブレークポイントの設定から独立したエミュレータ出力信号の事。トリガ出力の設定数に制限はない。

トライグラフ(Trigraph)
「??」で始まる 3 文字のシーケンス。ISO C で定義されており、1 つの文字に置換される。

未割り当てセクション(Unassigned Section)
リンカのコマンドファイルで特定のターゲット メモリブロックに割り当てられていないセクション。リンカは、未割り当てセクションを割り当てるターゲット メモリブロックを検出する必要がある。

非初期化データ(Uninitialized Data)
初期値なしで定義されたデータ。C 言語では `int myVar;` は非初期化済みデータセクションに格納される変数を定義する。

アップロード(Upload)
エミュレータやプログラマ等のツールからホストコンピュータへ、またはターゲットボードからエミュレータへデータを転送する事。

USB
Universal Serial Bus の略。2 本のシリアル伝送線で PC と外部周辺機器の通信を行う外部周辺インターフェイス規格。Microchip 社製ハードウェア ツールが現在サポートしている USB バージョン。

USB	速度モード	最高速度 (毎秒ビット数)
USB 2.0	ハイスピード	480 Mbps
USB 3.0	スーパースピード	5 Gbps

ベクタ(Vector)
リセットまたは割り込みが発生した時にアプリケーションのジャンプ先となるメモリ番地。

Volatile
メモリ内の変数へのアクセス方法に影響を与えるコンパイラの最適化を抑制する変数修飾子。

警告(Warning)
MPLAB IDE/MPLAB X IDE の場合: デバイス、ソフトウェア ファイル、装置に物理的な損傷を与える可能性のある状況で、ユーザに注意を促すために表示されるメッセージ。

16 ビットアセンブラ/コンパイラの場合: 問題となる可能性のある状態を警告として報告するが、処理は停止されない。

ウォッチ変数(Watch Variable)
デバッグ セッション中に**[Watches]**ウィンドウで監視できる変数。

[Watches]ウィンドウ(Watches Window)
各ブレークポイントで更新されるウォッチ変数のリストを表示するウィンドウ。

ウォッチドッグ タイマ(WDT: Watchdog Timer)
PIC マイクロ コントローラに内蔵されたタイマの 1 つで、ユーザが設定した期間が経過するとプロセッサをリセットする。WDT の有効化/無効化と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 X 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。

12. サポート

以下のサポートがご利用になれます。

12.1 保証登録

www.microchip.com/mysoftware にアクセスし、お客様のツールをオンライン登録してください。myMicrochip アカウントをお持ちでないお客様は、そこでアカウントを登録する事もできます。アカウントを既にお持ちのお客様は、サインインしてから**[Register Hardware Tool]**をクリックしてください。

ツールをオンラインで登録すると、製品の最新情報の配信を受ける事ができます。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。

12.2 myMicrochip 変更通知サービス

Microchip 社の変更通知サービスは、お客様に Microchip 社製品の最新情報をお届けする配信サービスです。ご興味のある製品ファミリまたは開発ツールに関する変更、更新、リビジョン、エラッタ情報をいち早くメールにてお知らせします。

以下のリンク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。

www.microchip.com/pcn

よく寄せられる質問(FAQ)と登録方法の詳細もリンク先ページからご覧になれます。

Microchip 社の情報

商標

Microchip 社の名称とロゴ、「M」ロゴ、その他の名称、ロゴ、ブランドは米国および/またはその他の国における Microchip Technology Incorporated またはその支社および/または子会社の登録または未登録商標です。これらの Microchip 社商標に関する情報は、https://www.microchip.com/en-us/about/legal-information/microchip-trademarks でご覧になれます。

ISBN: 979-8-3371-1024-0

法律上の注意点

本書および本書に記載されている情報は、Microchip 社製品を設計、テスト、お客様のアプリケーションと統合する目的を含め、Microchip 社製品に対してのみ使用する事ができます。それ以外の方法でこの情報を使用する事はこれらの条項に違反します。デバイス アプリケーションの情報は、ユーザの便宜のためにのみ提供されるものであり、更新によって変更となる事があります。お客様のアプリケーションが仕様を満たす事を保証する責任は、お客様にあります。その他のサポートについては、弊社または代理店にお問い合わせになるか、www.microchip.com/en-us/support/design-help/ client-support-services をご覧ください。

Microchip 社は本書の情報を現状のままで提供しています。Microchip 社は明示的、暗黙的、書面、口頭、法定のいずれであるかを問わず、本書に記載されている情報に関して、非侵害性、商品性、特定目的への適合性の暗黙的保証、または状態、品質、性能に関する保証をはじめとするいかなる類の表明も保証も行いません。

いかなる場合も Microchip 社は、本情報またはその使用に関連する間接的、特殊的、懲罰的、偶発、的または必然的損失、損害、費用、経費のいかんにかかわらず、また Microchip 社がそのような損害が生じる可能性について報告を受けていた場合あるいは損害が予測可能であった場合でも、一切の責任を負いません。法律で認められる最大限の範囲を適用しようとも、本情報またはその使用に関連する一切の申し立てに対する Microchip 社の責任限度額は、使用者が当該情報に関連して Microchip 社に直接支払った額を超えません。

Microchip 社の明示的な書面による承認なしに、生命維持装置あるいは生命安全用途に Microchip 社の製品を使用する事は全て購入者のリスクとし、また購入者はこれによって発生したあらゆる損害、クレーム、訴訟、費用に関して、Microchip 社は擁護され、免責され、損害をうけない事に同意するものとします。特に明記しない場合、暗黙的あるいは明示的を問わず、Microchip 社が知的財産権を保有しているライセンスは一切譲渡されません。

Microchip 社のデバイスコード保護について

Microchip 社製品のコード保護機能について以下の点にご注意ください。

- Microchip 社製品は、該当する Microchip 社データシートに記載の仕様を満たしています。
- Microchip 社では、通常の条件ならびに仕様に従って使った場合、Microchip 社製品のセキュリティレベルは、現在市場に流通している同種製品の中でも最も高度であると考えています。
- Microchip 社はその知的財産権を重視し、積極的に保護しています。Microchip 社製品のコード保護機能の侵害は固く禁じられており、デジタル ミレニアム著作権法に違反します。
- Microchip 社を含む全ての半導体メーカーで、自社のコードのセキュリティを完全に保証できる企業はありません。コード保護機能とは、Microchip 社が製品を「解読不能」として保証するものではありません。コード保護機能は常に進歩しています。Microchip 社では、常に製品のコード保護機能の改善に取り組んでいます。