
Microchip Studio 使用者の手引き

序文

Microchip StudioはWindows® XP/Windows Vista®/Windows 7/8環境でAVR®/Arm®応用を書いてデバッグするための統合開発環境(IDE: Integrated Development Environment)です。Microchip Studioはプロジェクト管理ツール、ソースファイルエディタ、シミュレータ、アセンブラ、C/C++用前処理部、プログラミング、チップ上デバッグを提供します。

Microchip StudioはMicrochip AVRツールの範囲を完全に支援します。各々の新しい公開は新しいAVR/Armデバイスに対する支援だけでなく、ツールに対する最新の更新も含まれます。

Microchip Studioは第三者のソフトウェア供給者との相互作用を許す単位部構成を持ちます。GUIプラグインと他の単位部はシステムに対して書かれて取り入れることができます。より多くの情報についてはMicrochipにお問い合わせください。

この使用者の手引きはIDEの全ての主な機能を通してあなたを導きます。これは実践を伴う一群の映像として設計されています。各章はその部分を網羅する映像で始まります。

本書は一般の方々の便宜のため有志により作成されたもので、Microchip社とは無関係であることを御承知ください。しおりの[はじめに]での内容にご注意ください。

目次

序文	1	4.18. 追跡	153
1. 序説	4	4.19. 追跡表示部	155
1.1. 特徴	4	5. プログラミング ダイアログ	158
1.2. 新規及び注目点	4	5.1. 序説	158
1.3. インストール	10	5.2. インターフェース設定	161
1.4. お問い合わせ情報	11	5.3. ツール情報	163
2. 開始に際して	12	5.4. 基板設定/ツール設定	164
2.1. Microchip Studio、START、ソフトウェア内容	13	5.5. カード積み重ね	165
2.2. AVR®とSAMのハードウェア ツールとデバッグ	15	5.6. デバイス情報	165
2.3. データ可視器とPowerデバッグ実演	16	5.7. 発振器校正	166
2.4. インストールと更新	19	5.8. メモリ	167
2.5. Microchip陳列室とStudio拡張	21	5.9. ヒューズプログラミング	168
2.6. Atmel START統合	21	5.10. 施錠ビット	168
2.7. 新規プロジェクト作成	24	5.11. 製品識票	169
2.8. MCCプロジェクトのインポート	27	5.12. 製造ファイル	169
2.9. Arduino®スケッチから作成	30	5.13. 保護	172
2.10. 実装書き込みとキット接続	30	5.14. 自動ファームウェア更新検出	173
2.11. I/O表示部と 他の空からのプログラミング参考資料	34	6. その他のウィンドウ	173
2.12. エディタ: コードの記述と整理 (Visual Assist)	43	6.1. デバイス一括管理部	173
2.13. AVR®シミュレータ デバッグ	50	6.2. 使用者インターフェース プロファイル選択	174
2.14. デバッグ1: 中断点、段階実行、 呼び出しスタック	54	6.3. 利用可能なツール表示部	175
2.15. デバッグ2: 条件付きと活動付きの中断点	60	6.4. ツール情報ウィンドウ	177
2.16. デバッグ3: I/O表示部、メモリ表示部、監視	66	6.5. ファームウェア更新	178
3. プロジェクト管理	72	6.6. 検索と置換のウィンドウ	179
3.1. 序説	72	6.7. 雛形エクスポート ウィザード	181
3.2. GCCプロジェクト	74	6.8. キット動作設定	183
3.3. XC8プロジェクト	96	7. GNUツールチェーン	184
3.4. アセンブラプロジェクト	111	7.1. GNUコンパイラ集合(GCC)	184
3.5. プロジェクト外のインポート	115	7.2. Arm®コンパイラと ツールチェーン任意選択:GUI	184
3.6. Microchip Studioでの オブジェクト ファイル デバッグ	121	7.3. Arm® GNUツールチェーン任意選択	187
3.7. AVR® GCCプロジェクトの XC8プロジェクトへの変換	124	7.4. バイナリユーティリティ(Binutils)	189
4. デバッグ	125	7.5. AVR®コンパイラとツールチェーン任意選択:GUI	189
4.1. 序説	125	7.6. 一般的に使われる任意選択	193
4.2. デバッグ作業開始	125	7.7. 8ビット特有AVR® GCCコマンド行任意選択	195
4.3. デバッグ作業終了	125	7.8. 32ビット特有AVR® GCCコマンド行任意選択	196
4.4. 目的対象へ取り付け	126	7.9. バイナリユーティリティ(Binutils)	198
4.5. デバッグなしでの開始	126	8. XC8ツールチェーン	199
4.6. デバッグ制御	127	8.1. 序説	199
4.7. 中断点	128	8.2. XC8コンパイラとツールチェーン 任意選択:GUI任意選択	199
4.8. データ中断点	132	8.3. XC8ツールチェーン任意選択	201
4.9. 一瞬監視、監視、局所、自動のウィンドウ	141	9. Microchip Studioの拡張	205
4.10. データ情報	146	9.1. 拡張管理部UI	205
4.11. 逆アセンブリ ウィンドウ	148	9.2. Microchip拡張陳列室での登録	206
4.12. I/O表示部	149	9.3. Microchip Studioでの新規拡張インストール	208
4.13. プロセッサ表示部	150	9.4. Visual Assist	210
4.14. レジスタ表示部	150	9.5. Percepio Tracealyzer	211
4.15. メモリ表示部	150	9.6. QTouch®構成器とライブラリの概要	211
4.16. 呼び出しスタック表示部	151	9.7. スクリプト手書き拡張	213
4.17. オブジェクト ファイル形式	153	10. メニューと設定	216
		10.1. 既存のメニューとツールバーの独自化	216
		10.2. 設定のリセット	217

目次

10.3. 任意選択ダイアログ枠	217
10.4. コード断片管理部	245
10.5. 外部ツール	247
10.6. 予め定義されたキーボードショートカット	249
11. コマンド行ユーティリティ(CLI)	257
12. 良くある質問	258
12.1. 過去のAVR [®] ソフトウェアと 第三者製品との互換性	259
12.2. Microchip Studioインターフェース	260
12.3. 性能の問題	261
12.4. ドライバとUSBの問題	261
13. 文書改訂履歴	263
Microchipウェブサイト	264
製品変更通知サービス	264
お客様支援	264
Microchipデバイスコード保護機能	264
法的通知	264
商標	265
品質管理システム	265
世界的な販売とサービス	266

1. 序説

1.1. 特徴

Microchip Studioはプロジェクトの開発とデバッグのための大きな機能群を提供します。最も注目に値する特徴が下で一覧にされます。

- 強力なVisual Assist拡張を特徴とするC/C++とアセンブリ言語用の豪華なコード エディタ
- 高度なデバッグ機能を持つ周期が正確なシミュレータ
- どのAVR基盤でも試作のために単位部構成応用の作成を許して構造部を提供する高度なソフトウェア枠組み
- デバッグ ツールを用いる実際のデバイス上でのデバッグ
- お客様プラグインの厳格な統合を許すための豪華なSDK
- 多くのMicrosoft® Visual Studio®プラグインと互換

1.2. 新規及び注目点

入手可能な新しい機能

1.2.1. AVR®とSAMデバイス用Microchip Studio

AVR®とSAMデバイス用Microchip Studio 7.0.2542

- AVRとSAMデバイス用Microchip StudioにAtmel Studioの改名
- Microchip StudioインストーラはAVR GCCツールチェーン、Arm GCCツールチェーン、それとAVRデバイス支援付きMPLAB® XC8コンパイラを扱います。MPLAB XC8コンパイラの全ての最適化任意選択を解錠するにはPRO許諾権を得てやってみてください。
- デバッグ構成設定用既定最適化は-Ogで、以前は-O1でした。
- 大きなプロジェクト/ファイルで改善されたスクロール性能
- AVRマクロ アセンブラ 2.2.8版
- 3.42に戻す以前版を含む高度なソフトウェア枠組み(Advanced Software Framework) 3.49.1
- 更新されたキット認識

Atmel Studio 7.0.2397

- 接続問題を修正する更新したnEDBGファームウェア(Ver.1.14.464)

Atmel Studio 7.0.2389

Atmel Studio 7.0.2389は以下を含みます。

- 高度なソフトウェア枠組み(ASF) 3.47.0
- AVR 8ビットGCCツールチェーン 3.6.2
- 上り回線を持つArm GCCツールチェーン 6.3.1版: GCC(Arm/embedded-6-branch改訂249437)、GNU Arm組み込みツール チェーン: 6-2017-q2-update
- 2019年9月現在、インストーラに含まれる最新デバイスシステムパックの内包、最新のデバイス支援を得るために更新を調べるのにデバイスパック管理部を使ってください。また、旧来支援についてはより古いデバイスシステムパックをダウンロードしてください。
- 以下の問題に対する修正を含みます。
 - AVR/SV-8221 : EDBGでATtiny817に見られるSRAMアクセスでの問題
 - AVR/SV-8212 : SAM D21に対する目的対象電圧読み込み失敗
 - AVR/SV-8187 : atpack管理部でCMSIS 5.4.0図表を支援
 - AVR/SV-8170 : SAM D51, E5x - フラッシュでのECCで不正なELF解析
 - AVR/SV-8105 : SAM E54に対する書き込み前の使用者ページ消去失敗
 - AVR/SV-8073 : **atprogram**を使うSAM E54使用者ページ読み書き
 - AVR/SV-8130 : Studioがインストールされたものの代わりに古いJLinkArm.dllを取り上げる
 - AVR/SV-8166 : リンカ スクリプトでの不正なメモリ定義のため、tinyAVR® 2デバイスをコンパイルしない
 - AVR/SV-8176 : UPDIで壊されるEEPROMと使用者識票の消去
 - AVR/SV-8158 : StudioでAVR64DA128用ASMプロジェクトが構築不能
 - AVR/SV-8123 : デバイスID不一致時に電圧読み込みが動かない
 - AVR/SV-8152 : stashからbitbucketへのlib-elf-dwarf構築ジョブを更新
 - AVR/SV-8132 : J-Linkでのチップ消去発行
 - AVR/SV-8131 : SAMD21J17DはAtmel-ICEで書き込み/デバッグが不能
 - AVR/SV-8171 : SAM L11のBOOTOPT書き込み問題
 - AVR/SV-8159 : ELFファイル内からSAM L11のBOCOR書き込み不能
 - AVR/SV-8149 : TrustZone SG命令に対する中断点(ブレークポイント)支援
 - AVR/SV-8182 : SAM L11に対するセキュア ブート支援での問題

Atmel Studio 7.0.1931

Atmel Studio 7.0.1931は以下を含みます。

- 高度なソフトウェア枠組み(ASF) 3.40.0
- 新しいMicrochip展示室拡張
- Atmel START拡張は必要とされるデバイスバック依存性で改善された反響を提供
- TrustZoneでのArm® Cortex®-M23の支援
- 新しいnEDBGデバッグ基盤を持つキットの支援
- デバイス支援
 - ATSAMHA1E[14|15|16]AB
 - ATSAML10[D|E][14|15|16]A
 - ATSAML11[D|E][14|15|16]A
 - ATtiny202/204, ATtiny402/404/406, ATtiny804/806/807, ATtiny1604/1606/1607
- AVR 8ビットGCCツールチェーン 3.6.1
- 上り回線を持つArm GCCツールチェーン 6.3.1版: GCC(Arm/embedded-6-branch改訂249437)、GNU Arm組み込みツールチェーン: 6-2017-q2-update
- Atmel Studio 7.0.1931は7.0.1645に存在した以下の問題に対する修正を含みます。
 - AVR/SV-8001: ツール ファイムウェア更新の不安定性
 - AVR/SV-8063: ELF製品ファイル書き込みがATtiny817系に対するヒューズを不支援
 - AVR/SV-8075: ATSAM4Lでのデバッグ開始が或る場合で不安定
 - AVR/SV-7895: プロジェクト間のリンクを持つ解決策が不正なファイルをコンパイル
 - AVR/SV-7745: 副フォルダでキックしたファイルが構築失敗を発生
 - AVR/SV-7939: AVRデバイスに対して関数中断点(ブレークポイント)が失敗
 - AVR/SV-8005: 或る場合でM0+デバイスのヒューズとメモリの書き込みが失敗

Atmel Studio 7.0.1645

Atmel Studio 7.0.1645は以下を含みます。

- 高度なソフトウェア枠組み3.35.1.898
- デバイス支援
 - ATmega4808, ATmega4809
 - ATtiny1614, ATtiny3214, ATtiny3216, ATtiny3217
 - ATSAMC[20|21][J|N][15|17|18]A
 - ATSAMD20[E|G|J][14|15|16]B
 - ATSAMD51[G|J|N|P][18|19|20]A
 - ATSAME[51|53|54][J|N][18|19|20]
 - ATSAME70[N|Q][19|20|21]B
 - ATSAMS70[J|N|Q][19|20|21]B
- AVR 8ビットGCCツールチェーン 3.6.1版
- 上り回線を持つArm GCCツールチェーン 6.3.1版: GCC(Arm/embedded-6-branch改訂249437)、GNU Arm組み込みツールチェーン: 6-2017-q2-update
- Atmel Studio 7.0.1645は7.0.1417で提示された以下の問題に対する修正を含みます。
 - AVR/SV-4914: 新しいavr-gccの_int24と_uint24の型の支援を追加
 - AVR/SV-7106: UNIX®行末でのHEX解析失敗
 - AVR/SV-7202: Arduinoライブラリの群化はより良い表現を持つことができます。
 - AVR/SV-7742: Arduino zeroに対してインポートしたArduinoスケッチのコンパイルが異常を示す。
 - AVR/SV-7798: ELFからのATtiny817ヒューズのプログラミングの問題を修正
 - AVR/SV-7834: CMSIS DFPのダウンロードに対して一括管理部が失敗
 - AVR/SV-7845: _ReallyTerminateAfterLaunchFinishedでの破壊
 - AVR/SV-7854: **atprogram**によって処理されないNaN値
 - AVR/SV-7876: **KitsDatabase.xml**のためにhttp先頭部にチェックサム領域を追加
 - AVR/SV-7877: 外部SRAMを持つデバイスが利用可能なSRAM計算に対して失敗
 - AVR/SV-7883: AS 7.0.1417のインストール中にKB2978092に対する不正な警告メッセージ
 - AVR/SV-7889: AS 7.0.1417で8ビットAVRに対して歪んだデバッグ情報
 - AVR/SV-7892: SAML22のRWWフラッシュ メモリ書き込み失敗
 - AVR/SV-7903: Atmel Studioが自動的にGPNVM8,7ビットを設定、それによってTCMを許可
 - AVR/SV-7911: ATmega4809でのデバイスID読み込みの問題

- AVRSV-7927 : デバイスプログラミングの保護ビット ウィンドウはMCUに応じて常に利用可能でないかもしれません。
- AVRSV-7961 : BOD(BODCFG.LVL)に対するFUSE構成設定警告がAtmel Studioで不正
- AVRSV-7973 : SAM4Lでプログラミング作業外のチップ消去が失敗

注: Atmel Studio 7.0.1645で動くため、QTouch構成器(Composer)拡張は5.9.122またはそれ以降版に更新されなければなりません。

Atmel Studio 7.0.1417

Atmel Studio 7.0.1417は7.0.1416で提示された以下の問題に対する修正を含みます。

- AVRSV-7827 : 32ビットWindows®でのインストールに対して新しいWinUSBドライバが失敗

Atmel Studio 7.0.1416

Atmel Studio 7.0.1416では以下の変更が行われました。

- AVR Dragon、AVRISP mkII、JTAGICEmkII、JTAGICE3、AVR ONE!、STK600、QT600用WinUSBに対して変更されたドライバ
- インストーラ改良
- 古いデバイス系列一括をインストールするための支援を改良
- 上り回線を持つAVR 8ビットGCCツールチェーン 3.6.0版
 - GCC 5.4.0
 - Binutils 2.26.20160125
 - avr-libc 2.0.0
 - gdb 7.8
- 上り回線を持つArm GCCツールチェーン 6.2.1版
- Atmel Studio 7.0.1645は7.0.1417で提示された以下の問題に対する修正を含みます。
 - gcc(Arm/embedded-6-branch改訂243739)、GNU Arm組み込みツールチェーン: 6-2016-q4-major
 - Binutils 2.27
 - gdb 7.12
- 高度なソフトウェア枠組み3.34.1

Atmel Studio 7.0.1416は7.0.1188で提示された以下の問題に対する修正を含みます。

- AVRSV-7486 : 高クロックでのCortex®-M0+ SAMデバイスのデバッグが失敗するかもしれない。
- AVRSV-7492 : SAMD10での何回かの再開/休止後の不正なPC値
- AVRSV-7693 : 監視ウィンドウからソースへ行くとAtmel Studio破壊
- AVRSV-7741 : ISP/SPIプログラミングで\$100または\$1000の大きさでのフラッシュメモリまたはEEPROMの書き込みが失敗

Atmel Studio 7.0.1188

Atmel Studio 7.0.1188では以下の変更が行われました。

- 新しいAVR8X基本構造に対する支援を追加
- インストーラ改良
- 改善されたArduinoインポート
- プログラミングダイアログでヒューズが一覧にされる方法を変更
- 上り回線を持つAVR 8ビットGCCツールチェーン 3.5.4版
 - GCC 4.9.2
 - Binutils 2.26
 - avr-libc 2.0.0
 - gdb 7.8

Atmel Studio 7.0.1188は7.0.1006で提示された以下の問題に対する修正を含みます。

- AVRSV-7149 : EEPROM書き込み時に\$FFのバイトが不正に飛ばされる。
- AVRSV-7393 : COFFオブジェクトファイルデバッグ時にAtmel Studio後処理部が破壊
- AVRSV-7564 : Atmel Studioインストールが立往生
- AVRSV-7580 : Atmel StudioがSAM Cortex® M7デバイスでDCACHEプロパティを扱わない。
- AVRSV-7582 : ファイル保存時の空白削除が予期した効果を示さない。
- AVRSV-7594 : デバッグ停止時の或る場合にAtmel Studio破壊
- AVRSV-7602 : 現在の関数の範囲を見つけれない。
- AVRSV-7607 : SAML2xとSAMC2xのデバイスに対する無効なMTB緩衝部開始アドレス

Atmel Studio 7.0.1006

以下の変更はAtmel Studio 7.0.1006で行われます。

- Atmel Studioの範囲内でAtmel STARTプロジェクトを作成して構成設定することを使用者に許す新しいAtmel START拡張

- デバッグ作業で複数単位部を読み込み設定する能力(実験用)
- 上り回線付きAVR 8ビットGCCツールチェーン 3.5.3版
 - GCC 4.9.2
 - Binutils 2.26
 - avr-libc 2.0.0
 - gdb 7.8
- 上り回線付きArm GCCツールチェーン 5.3.1版
 - GCC (Arm/embedded-5-branch改訂234589)
 - Binutils 2.26
 - gdb 7.10

Atmel Studio 7.0.1006は7.0.943で提示された以下の問題に対する修正を含みます。

- AVRSV-6878 : Atmel StudioでのいくつかのSAMデバイスで一度だけ書けるWDTレジスタを書く。
- AVRSV-7470 : SAM Cortex[®]-M7デバイスが時々開始に失敗する。
- AVRSV-7471 : 外部と内部のRAMを持つデバイスが利用可能として全てのRAMを一覧にする。
- AVRSV-7473 : Atmel Studioが始動中に停止する。
- AVRSV-7474 : Atmel Studioに接続したキットがQTouch Startページで列挙(接続認証)されない。
- AVRSV-7477 : 全ファイル表示>Show all filesが解決策(Solution)エクスプローラから動かない。
- AVRSV-7482 : SAM4Lでの中断点追加時の例外
- AVRSV-7486 : 高速クロックでのCortex[®]-M0+ SAMデバイスでデバッグが失敗するかもしれない。

Atmel Studio 7.0.943

Atmel Studio 7.0.943は以下の問題に対する修正を含みます。

- AVRSV-7459 : 大文字のファイル名を持つファイルを含むプロジェクトは構築で失敗し得ます。大文字のファイル名を持つファイルの保存はファイル名を小文字に変換します。

Atmel Studio 7.0.934

以下の変更はAtmel Studio 7.0.934で行われます。

- 上り回線付きAVR 8ビットGCCツールチェーン 3.5.3版
 - GCC 4.9.2
 - Binutils 2.26
 - avr-libc 2.0.0
 - gdb 7.8
- 上り回線付きAVR 32ビットGCCツールチェーン 3.4.3版
 - GCC 4.4.7
 - Binutils 2.23.1
 - Newlib 1.16.0
- 上り回線付きArm GCCツールチェーン 4.9.3版
 - GCC (Arm/embedded-4_9-branch改訂224288)
 - Binutils 2.24
 - gdb 7.8.0.20150304-cvs

Atmel Studio 7.0.934は7.0.790で提示された以下の問題を解決します。

- AVRSV-7376 : 遅いAtmel-ICEプログラミング
- AVRSV-7379 : 自動読み込みOFF時のヒューズまたは施錠ビット書き込み時の処理されない例外
- AVRSV-7396 : いくつかのマシンが'Exception in MemoryPressureReliever'に関する異常を示す。
- AVRSV-7400 : 標準動作時、デバッグ(Debug)メニューでDisable debugWire and Closeが見えない。
- AVRSV-7408 : 標準動作でAtmel Studio使用时、Set Startup Projectメニューが見つからない。

Atmel Studio 7.0.790

Atmel Studio 7.0.790で以下の機能が追加されます。

- ドラッグ&ドロッププログラミングを許す、組み込みデバッグ(EDBG)での大容量記憶動作を支援
- 使用者インターフェースプロファイルの導入。使用者はツールバーとメニュー項目のいくつかを削除されたインターフェースを選ぶことができます。
- 以前にインポートしたスケッチにライブラリのインポートを支援。Arduino ZeroとZero Proの支援を追加
- 既定で並行構築ON

Atmel Studio 7.0.790は7.0.634で提示された以下の問題を解決します。

- AVRSV-7084 : 更新中に使用者設定を主張

- AVRSV-7014 : いくつかのATmegaとATtinyのデバイスがシミュレータでのデバッグ開始に失敗
- AVRSV-7230 : 解決策(Solution)エクスプローラで”Show all files”に一貫性がない
- AVRSV-7062 : Xplained Miniキットのファームウェア更新不検出
- AVRSV-7164 : .binファイルへのフラッシュ メモリ読み込みが不正な.binファイルを作成
- AVRSV-7106 : Unix®または混合されたファイルで終わるHEXファイルが読み込み設定に失敗します。
- AVRSV-7126 : Armに対するデータ中断点はRAMに制限されないかもしれません。

Atmel Studio 7.0.634

本公開はSAM B11デバイスシステムに対するデバイス支援を追加します。

Atmel Studio 7.0.634は7.0.594で提示された以下の問題を解決します。

- AVRSV-6873 : Windows 10でのJungoドライバの問題
- AVRSV-6676 : Intelグラフィックドライバでの問題のためデバッグ開始が失敗

Atmel Studio 7.0.594

Atmel Studio 7.0.594は7.0.582で提示された以下の問題を解決します。

- AVRSV-7008 : Atmel Studio 7.0.582で6.2プロジェクトを開くと他の全ての構成設定に対してデバッグ構成設定を主張
- AVRSV-6983 : Studio拡張のアンインストールがいくつかの場合で動かない。
- AVRSV-7018 : プロジェクト作成がいくつかの文化特有のユーザー名で失敗
- AVRSV-7019 : ヘルプ表示部が32ビットマシンで動かない。
- 認証されたツール/デバッグを得るまたは見ることでの問題。対策は’Atmel Studio.0.594-readme.pdf’の2.4項をご覧ください。

Atmel Studio.0.582

- Visual Studio隔離したシェル2015に更新
- Atmel STARTと統合
 - このツールは便利で最適化された規則でああなたの組み込み応用を逃えるためにソフトウェア部品、ドライバ、中間ソフトウェア、例プロジェクトを選んで構成設定ことを手助けします。
- 新しいデバイス支援システム、CMSISバック適合
- データの処理と可視化に使われる、データ可視器(Data Visualizer)
- ヘルプ システム更新、改善された文脈感受性ヘルプ
- 高度なソフトウェア枠組み(Advanced Software Framework)3.27.3。ASFはソフトウェア階層と例の広大なソフトウェア ライブラリです。
- コードを高速に読み書き、整理、指示で手伝う、Atmel Studioに対するVisual Assist拡張の主更新
- Atmel StudioにArduinoスケッチ プロジェクトをインポート
- **atprogram**とプログラミング ダイアログで**Flip**互換ブートローダに対する支援。接続したデバイスがツールとして現れます。
- 上り回線を持つAVR 8ビットGCCツールチェーン 3.5.0版 (**注1**)
 - GCC 4.9.2
 - Binutils 2.25
 - avr-libc 1.8.0svn
 - gdb 7.8
- 上り回線を持つAVR 32ビットGCCツールチェーン 3.4.3版 (**注1**)
 - GCC 4.4.7
 - Binutils 2.23.1
 - Newlib 1.16.0
- 上り回線付きArm GCCツールチェーン 4.9.3版 (**注1**)
 - GCC 4.9 (改訂221220)
 - Binutils 2.24
 - gdb 7.8.0.20150304-cvs

1.2.2. Atmel Studio 6.2.サービスパック2

- 高度なソフトウェア枠組み(ASF) 3.21.0
- ATSAML21システムに対する支援追加
- ATM Cortex-M7コアに基づくATSAMV7システムに対する支援追加

1.2.3. Atmel Studio 6.2.サービスパック1

- 高度なソフトウェア枠組み(ASF) 3.19.0

注1: より多くの情報についてはツールチェーンの一部としてインストールされる**readme**をご覧ください。

- ・ 上り回線付きAVR 8ビットGCCツールチェーン 3.4.5版
 - GCC 4.8.1
 - Binutils 2.41
 - avr-libc 1.8.0svn
 - gdb 7.8
- ・ 上り回線付きAVR 32ビットGCCツールチェーン 3.4.2版
 - GCC 4.4.7
 - Binutils 2.23.1
- ・ 上り回線付きArm GCCツールチェーン 4.8.4版
 - GCC 4.8.4
 - Binutils 2.23.1
 - gdb 7.8
- ・ Arm(MTB)と32ビットAVR UC3(Nano追跡)用追跡緩衝部を支援
- ・ 目的対象への取り付けを支援

1.2.4. Atmel Studio 6.2

- ・ Atmelソフトウェア枠組み(ASF) 3.17.0
- ・ AVR 8ビットGCCツールチェーン 3.4.4(上り回線付きGCC 4.8.1)
- ・ AVR 32ビットGCCツールチェーン 3.4.2(上り回線付きGCC 4.4.7)
- ・ Arm GCCツールチェーン 4.8.3
- ・ Atmel-ICEを支援
- ・ Xplained Miniを支援
- ・ データ中断点を支援
- ・ tinyAVR[®]とmegaAVR[®]に対するOSCCAL校正読み込み
- ・ プログラミング ダイアログを用いるAVR 8ビット用のELF製品ファイル作成
- ・ Live監視
- ・ 以下を含むSAM3とSAM4系統のデバイスに対する非侵入型追跡を支援
 - 割り込みの追跡と監視
 - データ追跡
 - FreeRTOS™認知
 - 統計に基づくコード鑑定
- ・ Cortex[®] MO+に対するポーリング データ追跡を支援
- ・ SAM用既定デバッガは今やGDB。GDBはいくつかの筋書きでより良く最適化されたコードのデバッグを処理します。
- ・ インストールした全てのASF版に対してGCC基板プロジェクト(Microchip基板/使用者基板)作成を支援
- ・ 基板プロジェクト追加と削除(Add or Remove Board Project)雛形に新しいASF基板(ASF Board)ウィザード
- ・ 既定によって1つのASF版だけを読み込むことにより、新規例プロジェクト(New Example project)ダイアログの読み込み時間改善
- ・ IDR事象が今や出力ウィンドウ内の独立した枠で表示されるようになります。
- ・ LSSファイル構文を強調表示

1.2.5. Atmel Studio 6.1更新2

- ・ JTAGICE3でSAM D20デバイスを支援
- ・ 高度なソフトウェア枠組み(ASF:Advanced Software Framework) 3.11.0

1.2.6. Atmel Studio 6.1更新1.1

- ・ 6.1更新1で導入されたXMEGAデバイスに対するブート領域のプログラミングを修正
- ・ SAM4LSP32ベアボーンプロジェクト構成設定を修正

1.2.7. Atmel Studio 6.1更新1

- ・ 高度なソフトウェア枠組み(ASF:Advanced Software Framework) 3.9.1
- ・ 拡張開発キット(XDK:Extension Development Kit)。Microchip展示室拡張への組み込み応用プロジェクトの一括化を支援
- ・ SAM D20とSAM4Nのデバイスを支援
- ・ 実験的なnewlib-nanoとmultilibsを持つArm GCCツールチェーン 4.7.3

1.2.8. Atmel Studio 6.1

- ・組み込みデバッグが基盤を支援
- ・Xplained Proキットを支援
- ・高度なソフトウェア枠組み(ASF:Advanced Software Framework) 3.8.0
- ・AVR 8ビットGCCツールチェーン 3.4.2(上り回線付きGCC 4.7.2)
- ・AVR 32ビットGCCツールチェーン 3.4.2(上り回線付きGCC 4.4.7)
- ・Arm GCCツールチェーン 4.7.3
- ・CMSIS 3.20
- ・Visual Assist更新
- ・ファームウェア更新用コマンド行ユーティリティ
- ・シミュレータ用起因。シミュレーション実行と同時にレジスタ値を書くための起因ファイルを作成

1.2.9. Atmel Studio 6.0

- ・Atmel SAM-ICE™でのAtmel Armに基づくMCUを支援
- ・高度なソフトウェア枠組み(ASF:Advanced Software Framework) 3.1.3
- ・Armツールチェーン 3.3.1
- ・拡張としてQTouch構成器(Composer)を支援
- ・Visual Assist更新
- ・新しい拡張陳列室

1.2.10. AVR® Studio 5.1

- ・高度なソフトウェア枠組み(ASF:Advanced Software Framework)の新版
- ・Studio 5の更新なしに拡張管理部を通して新しいASF版の入手可能性とインストール
- ・プロジェクトを更新する能力を持つASFの並立版を支援
- ・C++プロジェクトに対する構文強調とより良いデバッグ支援
- ・AVR 32 Studio C++プロジェクトのインポートを支援
- ・AVRツールチェーンの新版
- ・全てのMicrochip AVRツールとデバイスに対する支援を持つ新しいコマンド行ユーティリティ(atprogram)
- ・ELFプログラミング支援を含むプログラミング ダイアログの強化
- ・様々な強化とバグ修正でのVisual Assistの新版

1.3. インストール

インストール注意書き

支援オペレーティング システム

- ・Windows 7 SP1またはそれ以降
- ・Windows Server 2008 R2 SP1またはそれ以降
- ・Windows 8/8.1
- ・Windows Server 2012とWindows Server 2012 R2
- ・Windows 10

支援基本構造

- ・32ビット (x86)
- ・64ビット (x64) – MPLAB XC8コンパイラ使用が64ビットを必要とすることに注意してください。

ハードウェア要件

- ・1.6GHzまたはより速いプロセッサを持つコンピュータ
- ・RAM
 - x86に対して1Gバイト
 - x64に対して2Gバイト
 - 仮想マシンで走行する場合、追加の512MバイトRAM
- ・6Gバイトの利用可能なハード ディスク空間

ダウンロードとインストール

- ・最新のMicrochip Studioインストーラーをダウンロードしてください。

- Microchip StudioはAtmel Studio 6.2以前版とAVR Studioの旧版と並走することができます。以前の版のアンインストールは不要です。
- 「システム要件」項でハードウェアとソフトウェアの必要条件を確認してください。
- ローカル管理者権限を持つことを確実にしてください。
- 開始する前に全ての作業を保存してください。インストールは必要とされる場合に再始動を指示するかもしれません。
- 全てのUSB/シリアルハードウェア装置を切断してください。
- インストーラ(実行可能ファイル)をダブルクリックしてインストールウィザードに従ってください。
- 一旦終了すると、インストーラは**Start Microchip Studio after completion**(完了後にMicrochip Studio開始)の任意選択を表示します。**Open**を選んだ場合、インストーラが管理者または昇格された権限のどちらかとして開始されたため、その後にMicrochip Studioが管理者権限で開始することに注意してください。

1.4. お問い合わせ情報

この版のMicrochip Studioで経験したどの問題も報告してください。更なる開発とMicrochip Studioの公開の改良を手助けし得る良い考えや要求を受け取ることも望みます。

あなたが出会うかもしれないどんな問題についても**Microchip支援**を調べてください。この頁から支援入り口を通してMicrochip支援に問い合わせることが可能です。

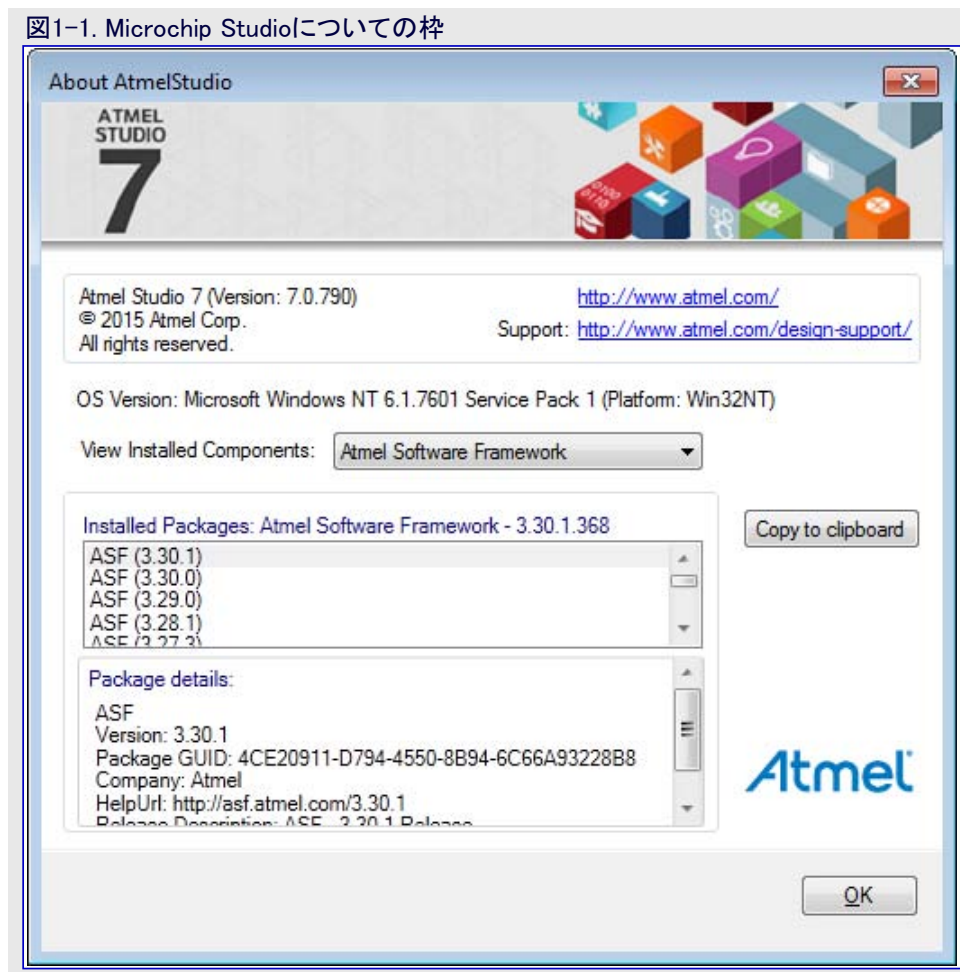
最終更新についてはMicrochipウェブサイトのMicrochip Studio頁を訪ねてください。

バグ報告

版ダイアログ(下図をご覧ください)からの情報を複製してMicrochipへのEメールにそれを含めてください。また、問題の詳細な記述を提供することを確実にしてください。

1. 問題を再作成する方法を記述してください。
2. 問題を起こすどの試験プログラムも添付してください。
3. 複製した版情報が使ったデバッグ基盤とデバイスを含むことを調べてください。

版ダイアログはファイルメニューの**Help**(ヘルプ)⇒**About Microchip Studio**(Microchip Studioについて)によって開かれます。デバッグ基盤とデバイスはデバッグ動作だった場合にだけ表示されます。内容を複製するには**Copy to clipboard**(クリップボードに複製)を押してください。



2. 開始に際して

Microchip Studioの開始に際して - 再生一覧。Atmel StudioはMicrochip Studioに改名されたことに注意してください。

<p>映像 説明</p>	<p>映像 説明</p>	<p>映像 実演コード</p>
<p>映像 実践</p>	<p>映像 実践</p>	<p>映像 実践</p>
<p>映像 実践</p>	<p>映像 実践</p>	<p>映像 実践</p>
<p>映像 実践</p>	<p>映像 実践</p>	<p>映像 実践</p>
<p>映像 実践</p>	<p>映像 実践</p>	<p>映像 実践</p>

事前要件

練習の多くはエディタとシミュータを用いることによって完了することができます。けれども、全てを網羅するために以下が推奨されます。

ハードウェア要件:

- ATtiny817 Xplained Pro
- 標準A-マイクロB USBケーブル

ソフトウェア要件:

- AVRとSAMデバイス用Microchip Studio
- avr-gccツールチェーン
- tinyAVR®デバイス用最新部品パック

使うMicrochip Studioプラグイン

- Atmel START 1.0.113.0またはそれ以降
- データ可視器(Data Visualizer)拡張2.14.709またはそれ以降

アイコン鍵識別子

以下のアイコンはこの資料内で各種仕事区分を識別するのと複雑さを減らすのに使われます。



情報: 特定の話題について文脈上の情報を伝えます。



助言: 有用な助言と技法を強調します。



すべきこと: 完了されるべき目標を強調します。



結果: 課題段階の予期される結果を強調します。



警告: 重要な情報を示します。




実行: 必要な時に目的対象から実行されるべき活動を強調します。

2.1. Microchip Studio、START、ソフトウェア内容

本項はAVR®とSAMのツール体系内の個々の概要とそれらが互いにどう関連するかを与えます。

[開始に際しての話題](#)



AVR® & SAM Tools: Intro & Overview

In this video:

Context in Microchip Tools Ecosystem

- IDE, Compiler, MCU & SW configurator tools, Firmware Libraries

START, Software Content and IDEs

- How these pieces fit together.
- START-based development
 - START user manual
 - Getting Started projects in START

Atmel Studio 7

- Bare-metal- vs. START-based development
- Build from scratch (bare-metal):
 - Getting Started Atmel Studio 7
 - Getting Started with AVR Tools

8-Bit PIC® MCU	16-Bit PIC® MCU and dPIC®	32-Bit PIC® MCU	AVR® MCU	SAM MCU
MPLAB® X IDE <small>MPLAB Xpress IDE (Cloud-Based)</small>			Atmel Studio	
MPLAB XC C Compilers			AVR GCC C Compilers	ARM GCC C Compilers
MPLAB Code Configurator			Atmel START	
Microchip Libraries for Applications (MLA)		MPLAB Harmony	Advanced Software Framework	
MPLAB XC PRO C Compiler Licenses			IAR Workbench	IAR Workbench Keil MDK

映像: AVRとSAMのツール体系概要

2.1.1. Atmel START / MCC

Atmel START/MCCは様々なソフトウェア枠組みに対するウェブに基づくソフトウェア構成設定ツールで、これはMCU開発開始を手助けします。新規プロジェクトまたは例プロジェクトのどちらかからの開始でも、Atmel START/MCCは便利で最適化した規則で、(ASF4とAVR Codeから)あなたの組み込み応用を仕立てるためのドライバやミドルウェア(中間ソフトウェア)のようなソフトウェア部品を選んで構成設定することを許します。一旦最適化されたソフトウェア構成設定が行われると、生成されたコードプロジェクトをダウンロードしてMicrochip Studio、IAR embedded Workbench®、Keil® µVision®を含み、あなたの選ぶIDEでそれを開くか、または単にmakefileを生成することができます。

Atmel START/MCCは以下を許します。

- ソフトウェアとハードウェアの両要件に基づくMCU選択を助けます。
- あなたの基板用の例を探して開発します。
- ドライバ、ミドルウェア、例プロジェクトを構成設定します。
- 有効なPINMUX配置の構成設定を助けます。
- システムクロック設定を構成設定します。

図2-1. START、ソフトウェア内容、IDE間の関係



2.1.2. ソフトウェア内容 (ドライバとミドルウェア)

高度なソフトウェア枠組み(ASF)

ASF(Advanced Software Framework)はお客様の設計時間を減らすために専門家によっては開発された検証済みのドライバとコード単位の豊富な組を提供します。これはドライバと高価値のミドルウェアを通してハードウェアに対する抽象化を提供することによってマイクロコントローラの使用を簡単化します。ASFは評価、試作、製造段階に使われるように設計された無料の開放ソースコードライブラリです。

SAM製品部門を支援するASF4はASFの第4主要世代です。これはメモリ量、コード性能を改善するだけでなく、Atmel START/MCCウェブユーザーインターフェースとも良く調和するように、枠組み全体の完全な再設計と再実装となります。ASF4はAtmel START/MCCと共に使われなければならない、これはASF2と3のASFウィザードを置き換えます。

Microchip.com: [ASF製品頁](#)

AVR® Code

AVR製品部門を支援するAVR Codeは8ビットと16ビットのPIC® MCUを支援する創設サービスと等価なAVR 8ビットMCU用の簡単なファームウェア枠組みです。AVR Codeはコード量とコード速度だけでなく、コードの簡潔性と信頼性に対しても最適化されています。AVR CodeはAtmel START/MCCによって構成設定されます。

2.1.3. 統合開発環境 (IDE)

IDE(Integrated Development Environment)はAtmel START/MCCで構成設定されてエクスポートされたドライバとミドルウェアのようなソフトウェア部品に基づく応用の開発(または例応用の更なる開発)に使われます。Atmel START/MCCはMicrochip Studio、IAR embedded Workbench®、Keil® µVision®を含むIDEの範囲を支援します。


Microchip Studioは全てのAVRとSAMのマイクロコントローラ応用を開発してデバッグするための統合開発基盤(IDP: Integrated Development Platform)です。Microchip Studio IDPはC/C++またはアセンブリ言語のコードで書かれた応用を書いて構築してデバッグするための継ぎ目がなく使い易い環境を与えます。それはAVRとSAMのデバイスを支えるデバッグ、書き込み器、開発キットに対しても継ぎ目なく繋がります。Atmel STARTとMicrochip Studio間の開発体験は最適化されています。Microchip StudioでのAtmel STARTに基づくプロジェクトの反復的な開発は再構成設定と機能併合を通して支援されます。

このMicrochip Studioに対する練習開始はIDEの主な機能の全てを通してあなたを導きます。これは実践を伴う映像系列として設計されています。各項はその項を網羅する映像で始まります。

2.2. AVR®とSAMのハードウェア ツールとデバッグ

本項はAVR®とSAM MCU用のハードウェア ツール体系を記述します。

[開始に際しての話題](#)

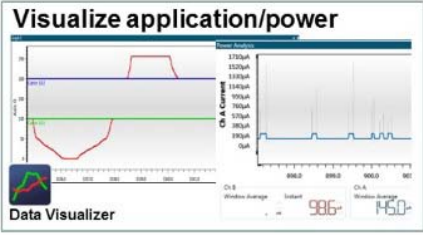


AVR® & SAM HW Tools & Debuggers

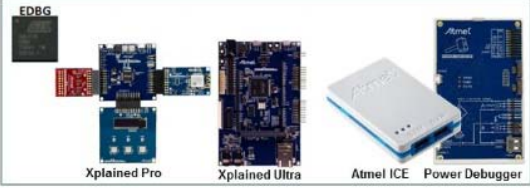
In this video:

Debugging Platform & user interface

- **Explained Development kit platform**
- **In circuit debuggers**
 - Atmel ICE / Power Debugger
- **Data Visualizer**
 - User Interface for debugging platform
 - Visualizes data to give insight to application
 - Analyze and correlate power consumption to code



Visualize application/power
 Data Visualizer
 CPU Current: 3120µA, 1320µA, 1180µA, 1040µA, 760µA, 570µA, 380µA, 190µA, 0µA
 CPU Voltage: 986mV, 1450mV



映像: AVR & SAMハードウェア ツール & デバッグ

データ可視器 (Data Visualizer)

データ可視器はデータを処理して可視化するためのプログラムです。データ可視器は組み込みデバッグ データ中継器インターフェース(DGI:Data Gateway Interface)とCOMポートのような様々な供給元からデータを受け取る能力があります。支援される探針や基板と共に使われると、端末や図表を用いて応用の走行時の追跡したり、コード実行と電力消費の相関を通して応用の電力諸費を分析します。走行時のコードの動きの完全な制御を持つことは決して容易ではありません。

独立型とMicrochip Studio用プラグインの両版が下のリンクのウェブサイトで入手可能です。

ウェブサイト: [データ可視器](#)

Atmel-ICE

Atmel-ICEはUPDI、JTAG、PDI、デバッグWIRE、aWire、TPI、SPI目的対象インターフェースを用いるAVRマイクロ コントローラとJTAGまたはSWD目的対象インターフェースを用いるArm® Cortex®-Mに基づくSAMマイクロ コントローラのプログラミングとデバッグ用の強力な開発ツールです。

Atmel-ICEはArm Cortex-Mに基づくSAMとチップ上デバッグ能力を持つAVRマイクロ コントローラをプログラミングしてデバッグするための強力な開発ツールです。

ウェブサイト: [Atmel-ICE](#)

Powerデバッグ

PowerデバッグはUPDI、JTAG、PDI、デバッグWIRE、aWire、TPI、SPI目的対象インターフェースを用いるAVRマイクロ コントローラとJTAGまたはSWD目的対象インターフェースを用いるArm Cortex-Mに基づくSAMマイクロ コントローラのプログラミングとデバッグ用の強力な開発ツールです。

加えて、Powerデバッグは設計の電力消費を測定して最適化するための2つの独立した電流感知チャンネルを持ちます。

PowerデバッグはCDC仮想COMポートだけでなく、SPI、UASRT、TWI、GPIO供給元からホスト コンピュータへ応用データを流すためのデータ中継器インターフェースも含まれます。


PowerデバッグはMicrochip Studioまたはそれ以降、または一般的なCMSIS-DAP部に接続する能力がある他の前処理ソフトウェアと共に動くCMSIS-DAP互換デバッグです。Powerデバッグは実時間分析のために電力測定と応用デバッグのデータをデータ可視器に流します。

ウェブサイト: [Powerデバッグ](#)

2.3. データ可視器とPowerデバッグ実演

本項はPowerデバッグを含みデータ可視器を用いる実演を示します。

[開始に際しての話題](#)

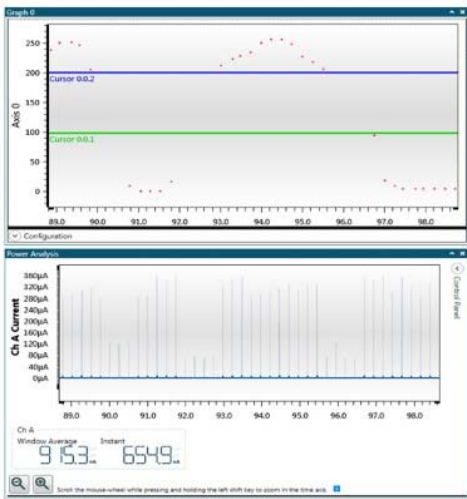


Studio 7: Data Visualizer & Power Debugging

In this video:
Studio 7: Data Visualizer & Power Debugging
Context
Low-power demo: RTC periodic timer, starts ADC conversion, via event system. ADC result sent on USART.

Features covered:

- **mEDBG: ATtiny817 Xplained Mini**
 - Data input: serial port
 - Visualization: terminal, graph
- **EDBG: ATtiny817 Xplained Pro**
 - Data Input: Serial + DGI (USART, SPI, I²C, GPIO)
 - Visualization: Graph (serial + DGI GPIO)
- **Power Debugger Analog module: ATtiny817 Xplained Pro**
 - Power measurement& DGI GPIO graphs
- **User-guide**
 - Tips for F1 access



映像: [データ可視器とPowerデバッグ実演](#)

```
/*
 * Power_Demo_ADC_SleepWalking.c
 * デバイス/基板: ATtiny817 Xplained Pro
 * 作成: 2017年8月6日 PM 3:15:21
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>

#define F_CPU (20E6/2)

void sys_init(void)
{
    _PROTECTED_WRITE(CLKCTRL.MCLKCTRLB, CLKCTRL_PEN_bm | CLKCTRL_PDIV_2X_gc);
}

void rtc_pit_init(void)
{
    RTC.CLKSEL = RTC_CLKSEL_INT1K_gc;
    RTC.PITCTRLA = RTC_PITEN_bm | RTC_PERIOD_CYC256_gc;
}

// picoPower 4: 事象システム 対 IRQ。IRQを不使用との比較
void evsys_init(void)
{
    EVSYS.ASYNCCH3 = EVSYS_ASYNCCH3_PIT_DIV128_gc;
    EVSYS.ASYNCUSER1 = EVSYS_ASYNCUSER1_ASYNCCH3_gc;
}

// picoPower 3: 自身の採取、例えば窓動作を評価 (起き上がり時間をかなり短縮)

void adc_init(void)
{
    ADC0.CTRLC = ADC_PRESC_DIV8_gc | ADC_REFSEL_VDDREF_gc;
    ADC0.CTRLA = ADC_ENABLE_bm | ADC_RESSEL_8BIT_gc;
    ADC0.MUXPOS = ADC_MUXPOS_AIN6_gc;

    ADC0.CTRLA |= ADC_RUNSTBY_bm; // picoPower 1: 故に休止で走行可
    ADC0.CTRLE = ADC_WINCM_OUTSIDE_gc; // picoPower 3: 故に自身の採取を評価可
    ADC0.INTCTRL = ADC_WCMP_bm;
    ADC0.WINHT = 200;
    ADC0.WINLT = 100;

    ADC0.EVCTRL = ADC_STARTEI_bm; // picoPower 4: 故に事象は変換を起動可
}

uint8_t adc_get_result(void)
{
    return ADC0.RESL;
}

// picoPower 5: 素早く送出、その後休止に戻る: compare 9600,115200,1250000のボーレートと比較
// 1バイトだけ送出に注意
#define BAUD_RATE 57600
void usart_init()
{
    USART0.CTRLB = USART_TXEN_bm;
    USART0.BAUD = (F_CPU * 64.0) / (BAUD_RATE * 16.0);
}

void usart_put_c(uint8_t c)
```

```
{
    VPORTB.DIR |= PIN2_bm | PIN6_bm;           // picoPower 2b: 下の送信禁止をご覧ください。
    USART0.STATUS = USART_TXCIF_bm;

    VPORTB.OUT |= PIN6_bm;
    USART0.TXDATAL = c;
    while(!(USART0.STATUS & USART_TXCIF_bm));
    VPORTB.OUT &= ~PIN6_bm;
    VPORTB.DIR &= ~PIN2_bm | PIN6_bm;         // picoPower 2b: 送信間でTxレイン禁止
}

// picoPower 2: 未使用GPIO禁止 (比較: なし、PORT_ISC_INPUT_DISABLE_gc、PORT_PULLUPEN_bp)

void io_init(void)
{
    for (uint8_t pin=0; pin < 8; pin++)
    {
        (&PORTA.PINCTRL)[pin] = PORT_ISC_INPUT_DISABLE_gc;
        (&PORTB.PINCTRL)[pin] = PORT_ISC_INPUT_DISABLE_gc;
        (&PORTC.PINCTRL)[pin] = PORT_ISC_INPUT_DISABLE_gc;
    }
}

int main(void)
{
    sys_init();
    rtc_pit_init();
    evsys_init();
    adc_init();
    io_init();
    usart_init();

    VPORTB.DIR |= PIN6_bm;
    VPORTB.OUT &= ~PIN6_bm;
    sei();

    // picoPower 1: 休止へ行く。休止なし、アイドル、スタンバイと比較
    set_sleep_mode(SLEEP_MODE_STANDBY);

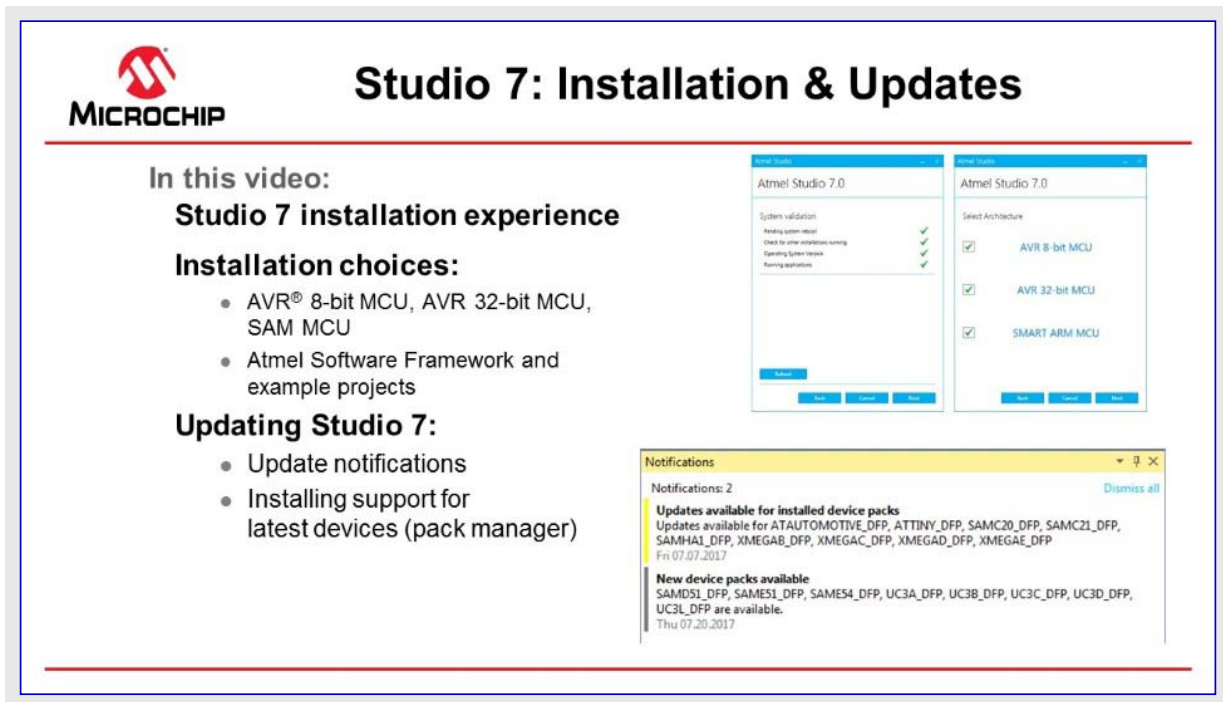
    while (1)
    {
        sleep_mode();
    }
}

ISR(ADCO_WCOMP_vect)           // picoPower 3: 関連採取の場合にだけ呼ばれます。
{
    ADC0.INTFLAGS = ADC_WCMP_bm;
    usart_put_c(adc_get_result());
}
```


2.4. インストールと更新

本項はAVRとSAMデバイス用Microchip Studioインストール、Microchip Studioやプラグインに対する更新インストールだけでなく、新デバイスに対する支援追加の手順を記述します。

[開始に際しての話題](#)



映像: [インストールと更新](#)

2.4.1. インストール

支援オペレーティング システム

- Windows 7 SP1またはそれ以降
- Windows Server 2008 R2 SP1またはそれ以降
- Windows 8/8.1
- Windows Server 2012とWindows Server 2012 R2
- Windows 10

支援基本構造

- 32ビット(x86)
- 64ビット(x64)

ハードウェア要件

- 1.6GHzまたはより速いプロセッサを持つコンピュータ
- RAM
 - x86に対して1Gバイト
 - x64に対して2Gバイト
 - 仮想マシンで走行する場合、追加の512MバイトRAM
- 6Gバイトの利用可能なハード ディスク空間

ダウンロードとインストール

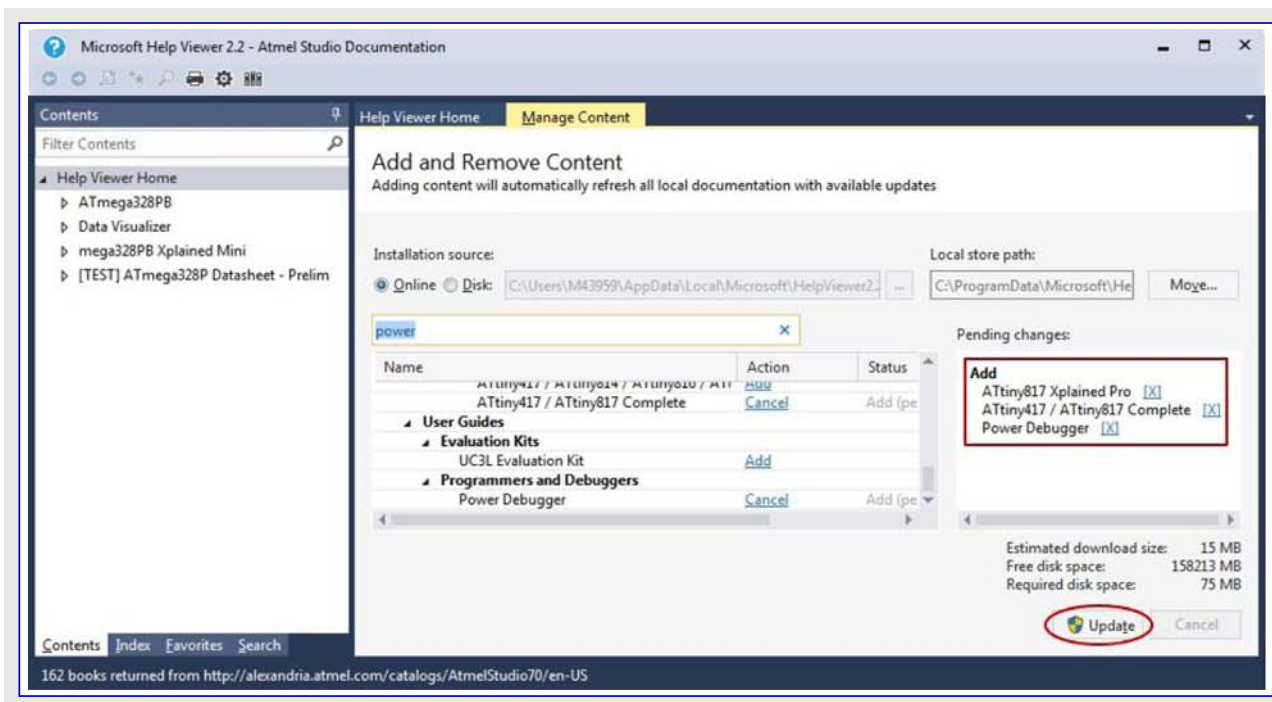
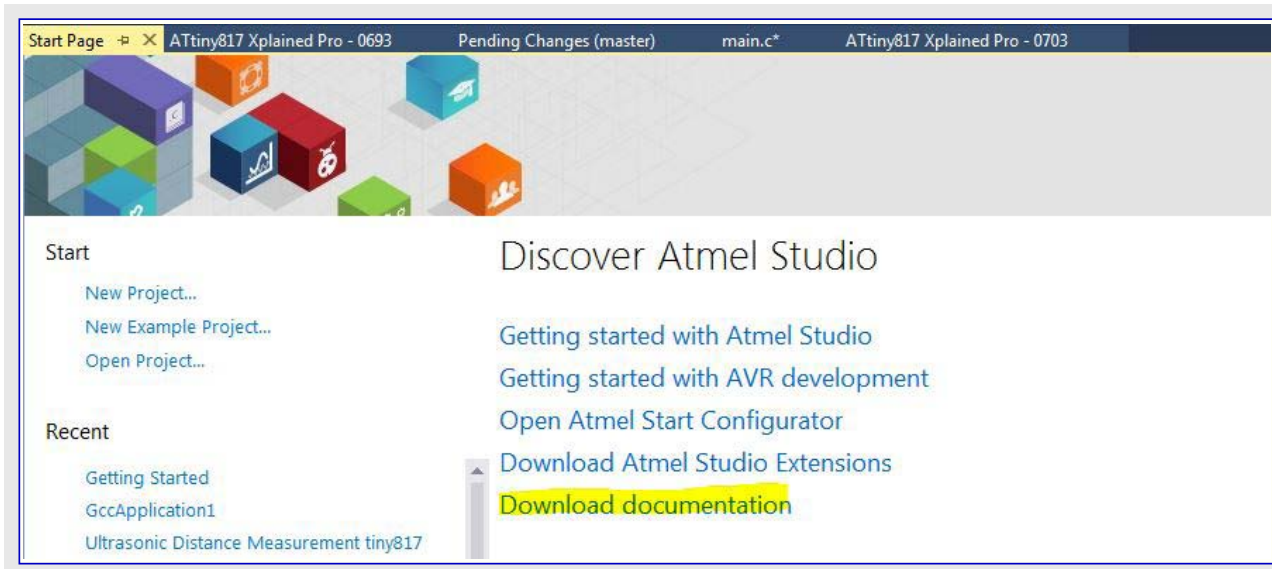
- 最新のMicrochip Studioインストーラーをダウンロードしてください。: [Microchip Studio](#)
 - ウェブ インストーラーは(<10Mバイトの)小さなファイルで、必要とされる時に指定される部分をダウンロードします。
 - オフライン インストーラーは組み込まれた全ての部分を持ちます。
- Microchip StudioはAtmel Studio 6.2以前とAVR Studioの旧版と並走することができます。以前の版のアンインストールは不要です。Microchip StudioはAtmel Studio 7と共に並走することはできません。
- 「システム要件」項でハードウェアとソフトウェアの必要条件を確認してください。
- ローカル管理者権限を持つことを確実にしてください。
- 開始する前に全ての作業を保存してください。インストールは必要とされる場合に再始動を指示するかもしれません。
- 全てのUSB/シリアル ハードウェア装置を切断してください。

- ・ インストーラ(実行可能ファイル)をダブル クリックしてインストール ウィザードに従ってください。
- ・ 一旦終了すると、インストーラは**Start Microchip Studio after completion**(完了後にMicrochip Studio開始)の任意選択を表示します。**Open**を選んだ場合、インストーラが管理者または昇格された権限のどちらかとして開始されたため、その後にMicrochip Studioが管理者権限で開始することに注意してください。
- ・ Atmel StudioやMicrochip Studioの早期版からMicrochip Studioを更新して違うユーザーアカウントで応用を走行する場合、局所応用ユーザーキャッシュを解消する必要があるかもしれません。この理由はインストーラがユーザー用キャッシュだけ解消し、それでインストールされるからです。これはWindowsのファイル エクスプローラから%localappdata%\Atmel\AtmelStudio\7.0のフォルダを削除することで行われます。
- ・ Microchip Studioに於いてタイトル バーの迅速起動領域傍で更新通知(旗のシンボル)を見るかもしれません。ここでは更新された構成部分やデバイスの支援を選んでインストールすることができます。

2.4.2. オフライン資料のダウンロード

オフラインで作業したい場合、Microchip Studio用のオフライン資料を使うのがお勧めです。これを行うにはMicrochip Studioの**Start Page**(開始頁)から**Download documentation**(資料ダウンロード)をクリックしてください。手助け表示部がポップアップすると、最初に**Online**(オンライン) 釦をクリックし、(利用可能な資料が表示されるまで待つ) **data sheets**(データシート)、**user manuals**(ユーザー手引書)、**application notes**(応用記述)のような興味のある資料を探してください。


下の例では**Power Debugger user manual**(Powerデバッガユーザー手引書)、**ATtiny817 Xplained Pro user manual**(ATtiny817 Xplained Proユーザー手引書)だけでなく、**ATtiny817 Complete data sheet**(ATtiny817完全データシート)もダウンロードするために選んでいます。その後の**Update**(更新)のクリックがダウンロードを開始します。



2.5. Microchip陳列室とStudio拡張

本項はMicrochip陳列室(gallery)を通してMicrochip Studioがどう拡張されて更新されるかを記述します。最も有用で人気のある拡張のいくつかが記述されます。

開始に際しての話題



Studio 7: Gallery & Extensions

In this video:

How to add extensions

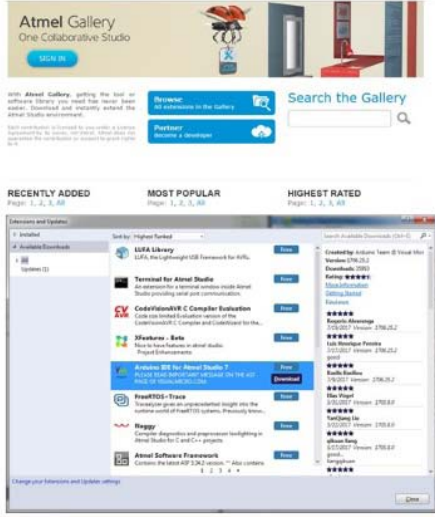
- Tools -> Gallery Profile

Extensions:

- Part of Studio 7:** Visual Assist, Atmel START, Data Visualizer, Toolchain
- Popular:** Arduino® IDE for Studio 7, LUFA Library, ASF (Naggy)
- Used in series:** Doxygen integrator, Git Source Control Provider,

Extension options/settings

- Tools → Options



映像: 陳列室、Studio拡張、更新


この短い映像はMicrochip Studioに拡張を追加する手順を説明します。これは既定によって含まれてこれらが使われる拡張を網羅します。人気のある拡張だけでなく、拡張の任意選択と設定を変更する方法も網羅されます。

ウェブサイト: [Microchip陳列室\(Gallery\)](#)

2.6. Atmel START統合

Atmel STARTとMicrochip Studio間での開発経験は最適化されています。本項はre-configure(再構成設定)とmerge(結合)の機能を通して、Microchip StudioでSTARTに基づくプロジェクトの反復的な開発手順を実演します。

開始に際しての話題



Studio 7: Atmel START Integration

In this video:

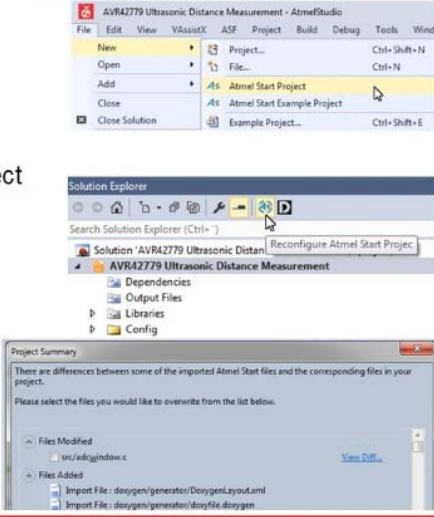
START-based dev. in Studio 7

Creating:

- New Atmel START project
- New Atmel START example project
 - Open:** Ultrasonic distance measurement example

Iterative development

- Re-configure Atmel START project
- Handling Diff/Merge
- AVR® code project documentation



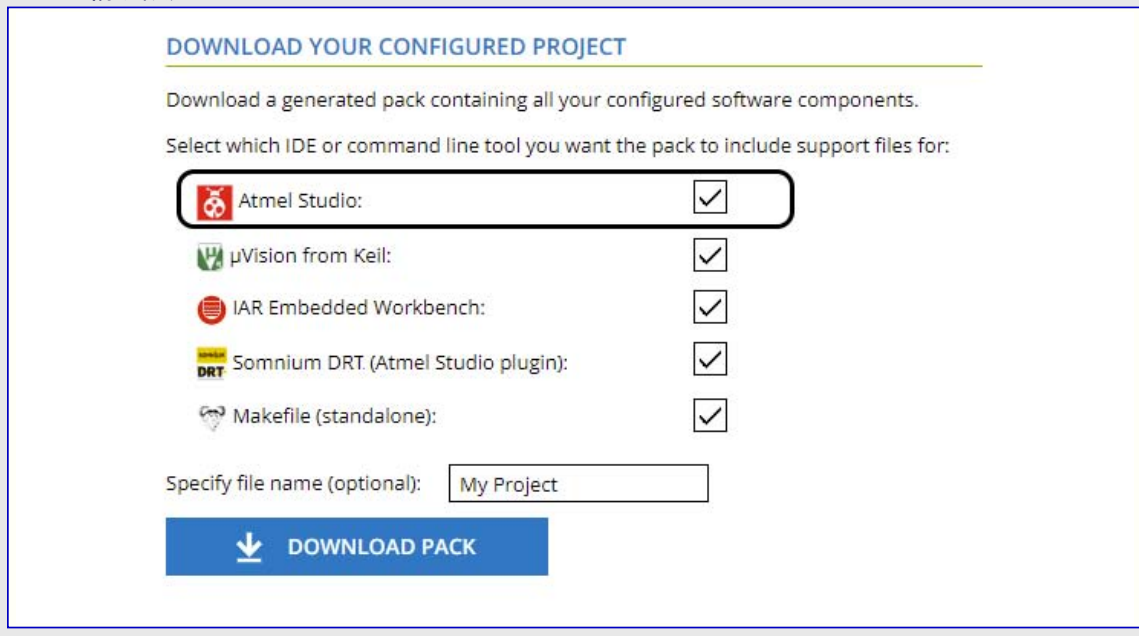
映像: Atmel START統合



すべきこと: Atmel STARTからプロジェクトをエクスポートします。

1. Atmel STARTウェブサイトで新しいプロジェクト(例または基板)を作成してください。
2. **Export Software Component**(ソフトウェア構成部品エクスポート)鈕をクリックしてください。Microchip Studioチェック枠がチェックされていることを確実にしてください。
3. **DOWNLOAD PACK**(一括ダウンロード)上でクリックしてください。atmelstart.atzip 一括ファイルがダウンロードされます。

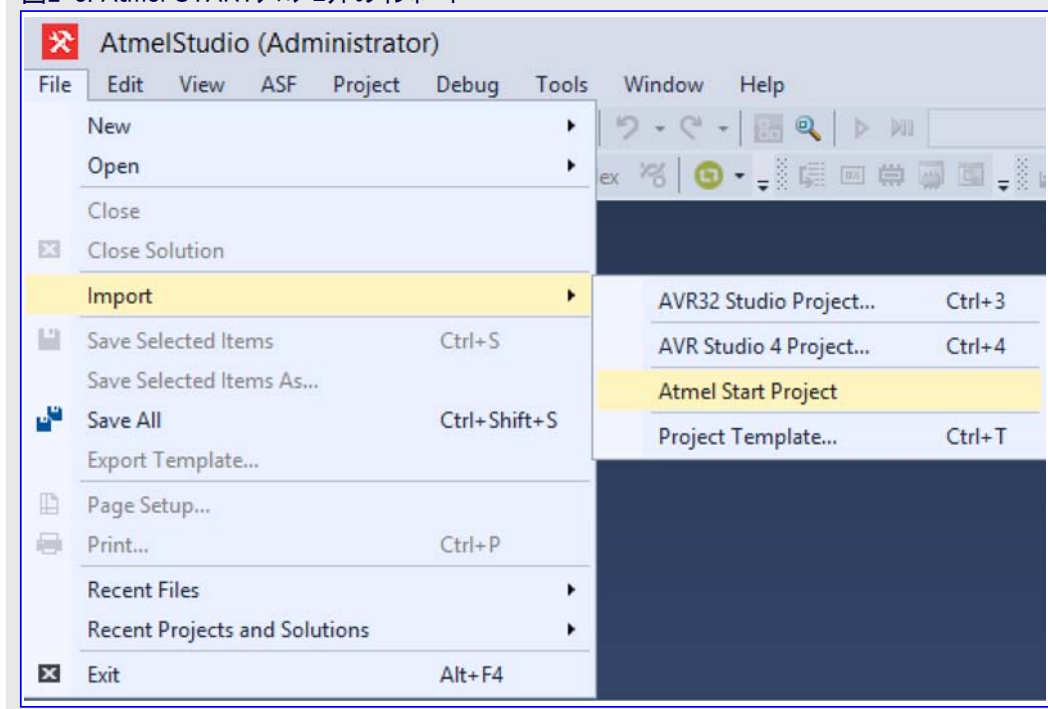
図2-2. 構成設定したプロジェクトのダウンロード



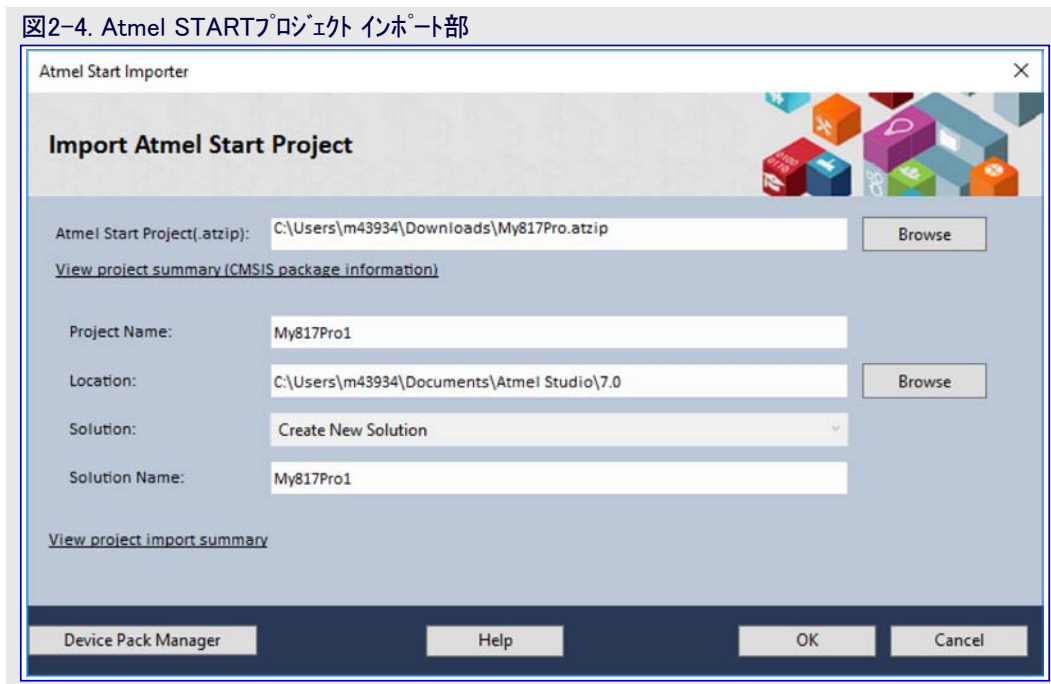
すべきこと: Atmel STARTの出力をMicrochip Studioにインポートします。

4. Microchip Studioを起動してください。
5. **File**(ファイル)⇒**Import**(インポート)⇒**Atmel START Project**(Atmel STARTプロジェクト)を選んでください。

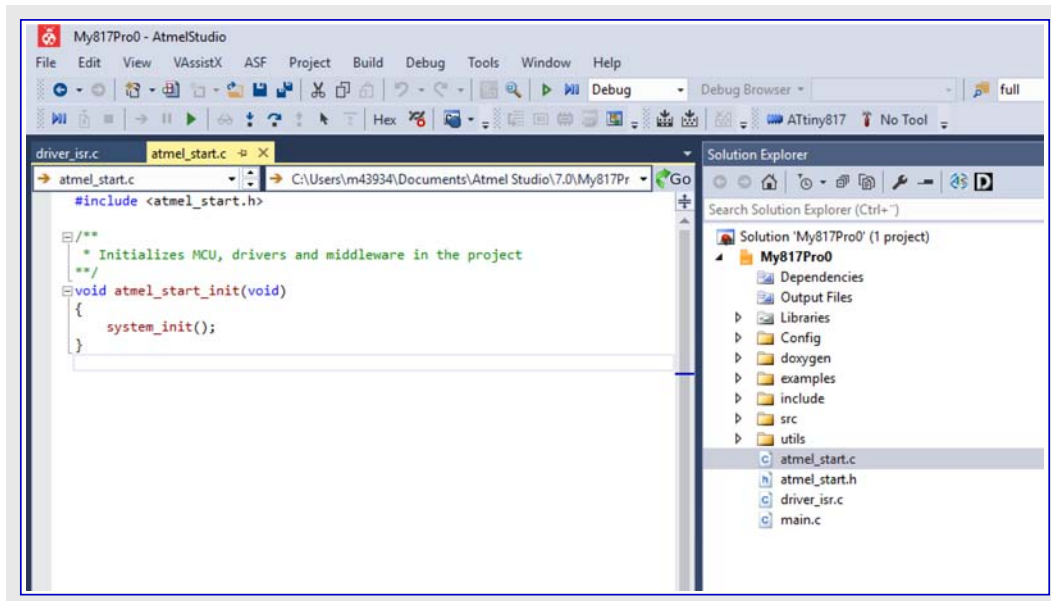
図2-3. Atmel STARTプロジェクトのインポート



6. ダウンロードした `atmelstart.atzip` ファイルを探して選んでください。
7. Atmel STARTインポートダイアログ枠が開きます。Project Name(プロジェクト名)、Location(場所)、Solution Name(解決策名)としてプロジェクトの詳細を入力してください。




8. 新しいMicrochip Studioプロジェクトが作成され、ファイルがインポートされます。

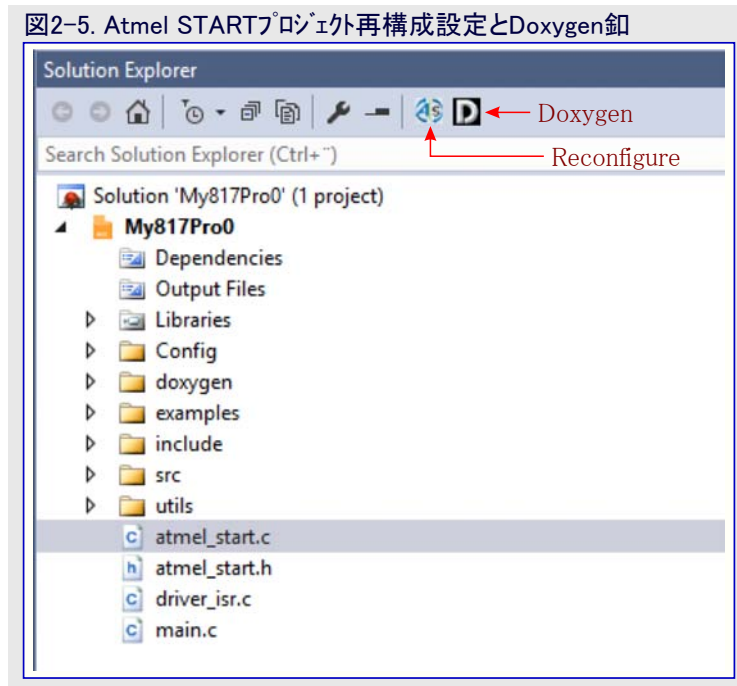


すべきこと: Atmel STARTの出力をMicrochip Studioにインポートします。

9. いくつかのプロジェクトはDoxygen用に形式化された資料を含みます。
注: Doxygenは<http://doxygen.org>からダウンロードされてインストールされなければなりません。Doxygen実行形式物の場所を示すようにMicrochip Studioを構成設定することをお勧めします。
10. 資料を生成するにはDoxygenアイコンをクリックしてください。Doxygenが走行して生成された資料が新しいウィンドウで開きます。

 **すべきこと:** Atmel STARTを用いてプロジェクトを再構成設定します。

11. **Reconfigure**(再構成設定)鈕をクリックするか、またはSolution Explorer内のプロジェクト節点上を右クリックし、そのメニューから**Reconfigure Atmel START Project**(Atmel STARTプロジェクトを再構成設定)を選んでください。
12. Microchip Studio内のウィンドウでAtmel STARTが開きます。




13. プロジェクトに対して必要な変更を行ってください。Atmel STARTウィンドウの下部で**GENERATE PROJECT**(プロジェクト生成)鈕をクリックしてください。

2.7. 新規プロジェクト作成

本項は新しいMicrochip Studioプロジェクトを作成する手順を概説します。

[開始に際しての話題](#)



Studio 7: Creating a New Project

In this video:

**Create new project:
Selecting the right project type**

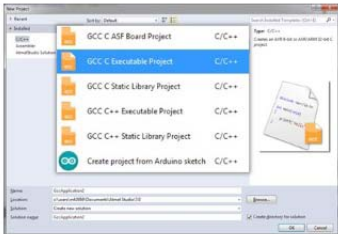
- GCC C/C++ Executable Project
- GCC C/C++ Static Library Project
- Create project from Arduino® Sketch


ASF3 Projects

- GCC ASF Board Project
- ASF Example Project


ASF4/AVR® Code Projects:

- Atmel START project
- Atmel START example project






The Atmel ATtiny817 Xplained Pro evaluation kit is a hardware platform to evaluate the Atmel ATtiny817 microcontroller. Supported by the Atmel Studio integrated development platform, the kit provides easy access to the features of the Atmel ATtiny817 and explains how to integrate the device in a customer design.

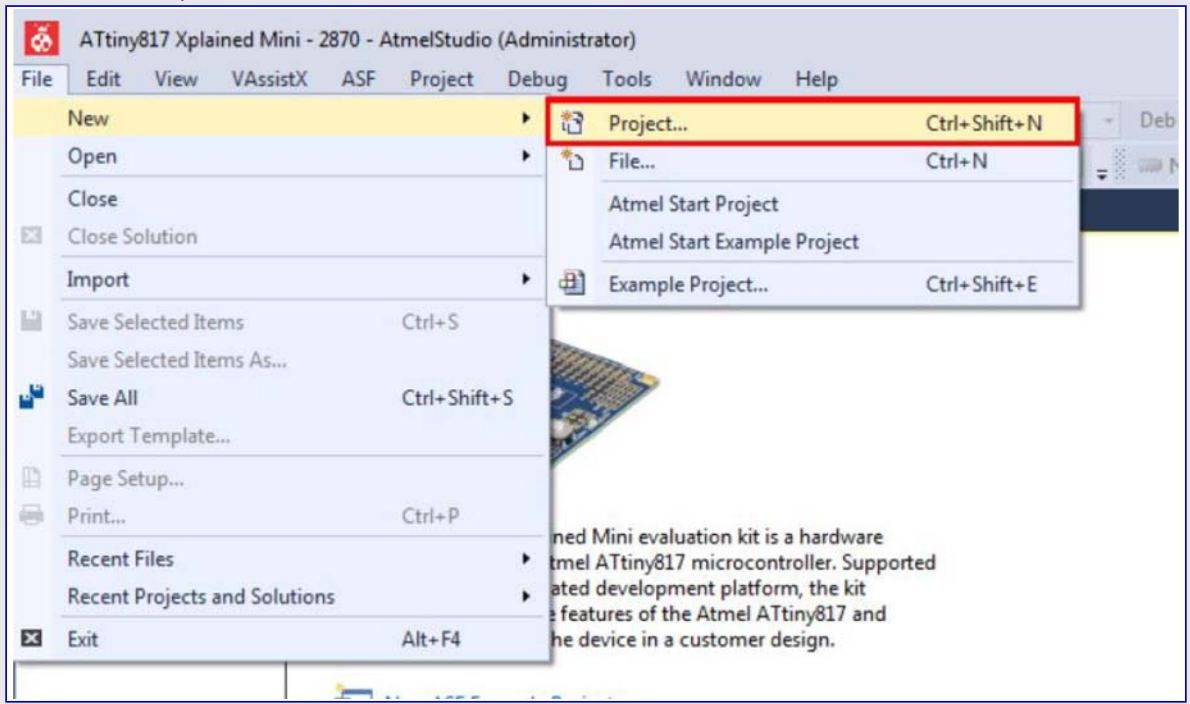
 Atmel START example projects using this board...
New Atmel START project using this board...

映像: 新規プロジェクト作成

 **すべきこと:** ATtiny817デバイス用の新しい空GCC C実行可能プロジェクトを作成します。

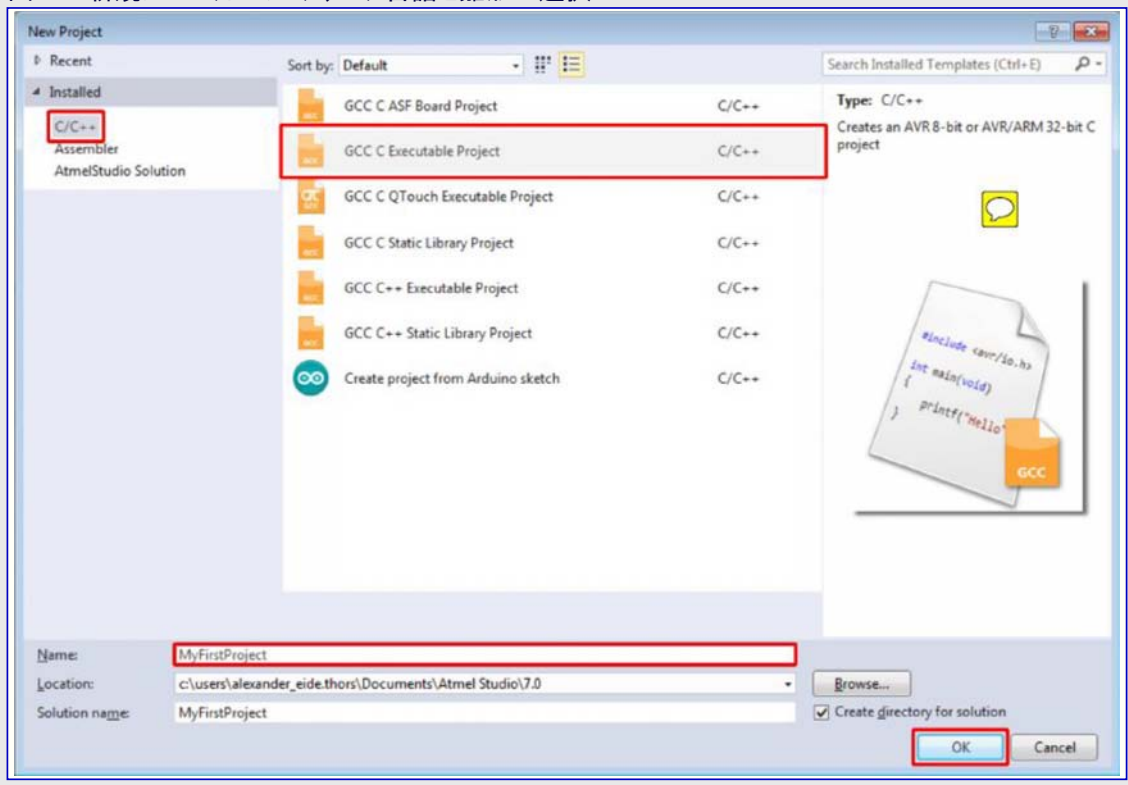
1. Microchip Studioを開いてください。
2. Microchip Studioに於いて、[図2-6](#)で描かれたように**File**(ファイル)⇒**New**(新規)⇒**Project**(プロジェクト)に行ってください。

図2-6. Microchip Studioでの新規プロジェクト作成



3. プロジェクト生成ウィザードが現れます。このダイアログは使われるプログラミング言語とプロジェクト雛形を指定するための任意選択を提供します。このプロジェクトはCを使い、故に左上隅でC/C++が選ばれることを確実にしてください。骨子の実行可能プロジェクトを生成するために雛形一覧からGCC C Executable Project任意選択を選んでください。Nameにプロジェクト名を与えてOKをクリックしてください。図2-7をご覧ください。

図2-7. 新規プロジェクトのプログラミング言語と雛形の選択



助言: 全てのMicrochip Studioプロジェクトは解決策(solution)に属し、既定によってMicrochip Studioは新しく作成された解決策とプロジェクトの両方に対して同じ名称を使います。解決策名領域は手動で解決策名を指定するのに使うことができます。

助言: create directry for solution(解決策用にディレクトリを作成)チェック枠は既定でチェックされます。この枠がチェックされると、Microchip StudioはLocation(場所)領域によって指定される場所で指定された解決策名を持つ新しいフォルダを作成します。

プロジェクト形式について

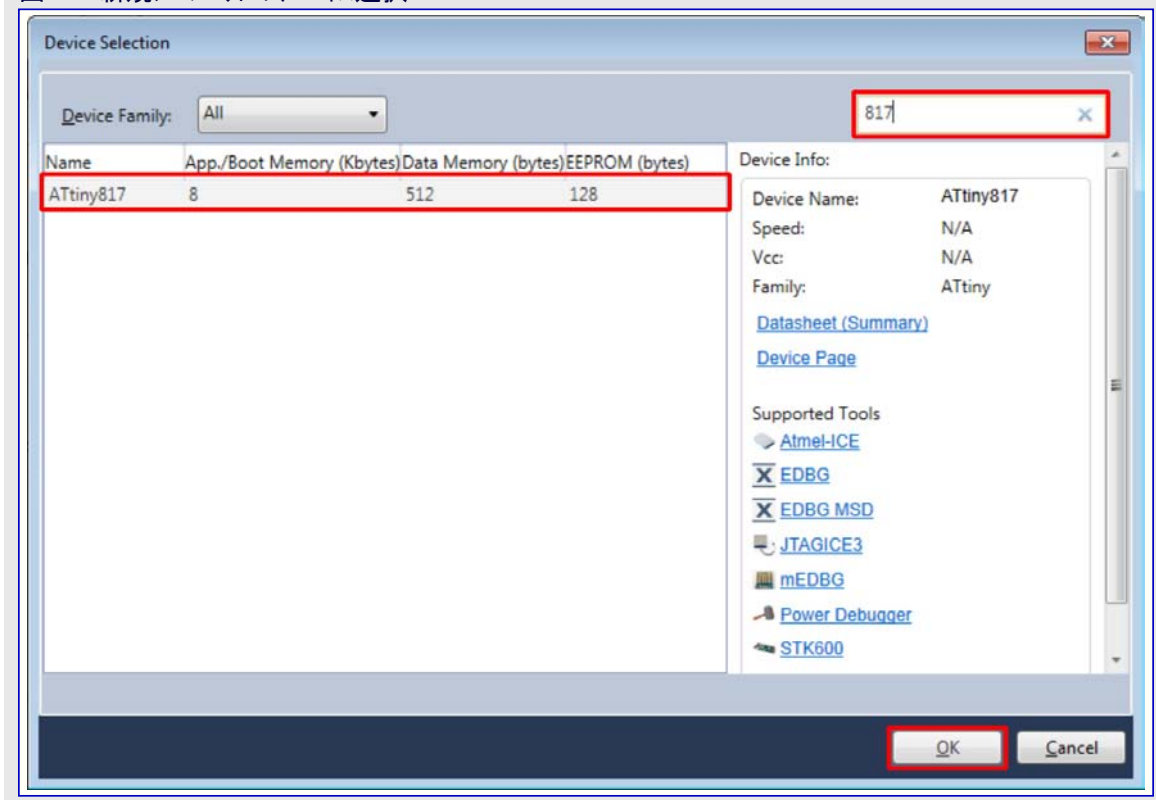
表2-1. プロジェクト形式

区分	プロジェクト雛形	説明
C	AVR XC8応用プロジェクト	MPLAB XC8コンパイラを使うように構成設定したAVR 8ビットプロジェクトを作成するにはこの雛形を選んでください。
C	AVR XC8ライブラリ プロジェクト	AVR 8ビットMPLAB XC8静的静的ライブラリ(LIB)プロジェクトを作成するにはこの雛形を選んでください。この事前コンパイルしたライブラリ(.a)は他のプロジェクト(閉じられたソース)へのリンク、または同じ機能を要する応用からの参照(コード再利用)に使えます。
C/C++	GCC C ASF基板プロジェクト	AVR 8ビットかAVR/Arm 32ビットのASF3基板プロジェクトを作成するにはこの雛形を選んでください。ASF3によって支援される各種基板から選んでください。
C/C++	GCC C実行可能プロジェクト	AVR 8ビットかAVR/Arm 32ビットのGCCプロジェクトを作成するにはこの雛形を選んでください。
C/C++	GCC C 静的ライブラリ プロジェクト	AVR 8ビットかAVR/Arm 32ビットのGCC静的ライブラリ(LIB)プロジェクトを作成するにはこの雛形を選んでください。この事前コンパイルしたライブラリ(.a)は他のプロジェクト(閉じられたソース)へのリンク、または同じ機能を要する応用からの参照(コード再利用)に使えます。
C/C++	GCC C++実行可能プロジェクト	AVR 8ビットかAVR/Arm 32ビットのC++プロジェクトを作成するにはこの雛形を選んでください。
C/C++	GCC C++ 静的ライブラリ プロジェクト	AVR 8ビットかAVR/Arm 32ビットのC++静的ライブラリ(LIB)プロジェクトを作成するにはこの雛形を選んでください。この事前コンパイルしたライブラリ(.a)は他のプロジェクト(閉じられたソース)へのリンク、または同じ機能を要する応用からの参照(コード再利用)に使えます。
アセンブラ	アセンブラプロジェクト	AVR 8ビットアセンブラプロジェクトを作成するにはこの雛形を選んでください。

注意: この表は既定プロジェクト形式だけを一覧にします。拡張によって他のプロジェクト形式が追加されるかもしれません。

4. 次に、プロジェクトがどのデバイスに対して開発されるのかを指定することが必要です。デバイスの一覧はDevice Selection(デバイス選択)ダイアログで提示され、これは図2-8.で描かれるように全体を通してスクロールすることができます。Device Family(デバイス系統)引き落としメニューまたは検索箱を用いることによって検索を狭めることが可能です。このプロジェクトはATtiny817 AVRデバイス用に開発されており、故に右上隅の検索箱に'817'を入力してください。デバイス一覧でATtiny817項目を選んでOKをクリックすることによってデバイス選択を承認してください。

図2-8. 新規プロジェクトのデバイス選択



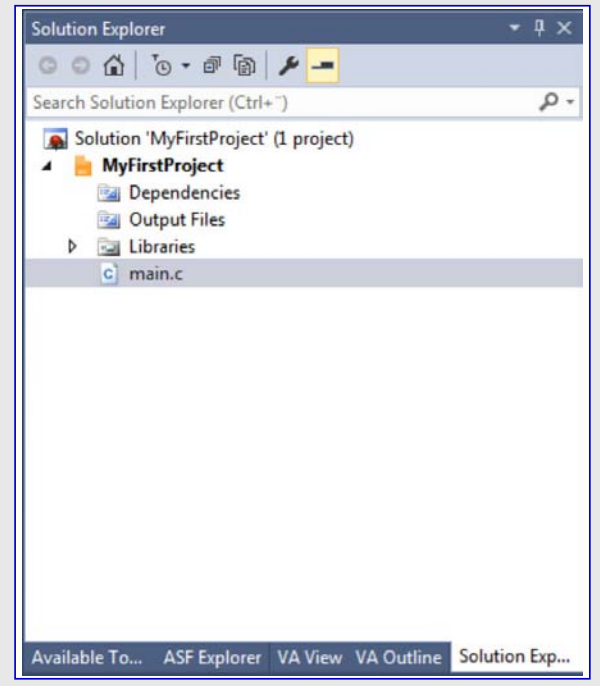


助言: 'tiny' に対する検索は支援される全てのATtinyデバイスの一覧が提供されます。'mega' に対する検索は支援される全てのATmegaデバイスの一覧が提供されます。Tools(ツール)⇒Device Pack Manager(デバイス一括管理部)は追加デバイス用の支援をインストールするのに使うことができます。



結果: 新規GCC C実行可能プロジェクトは今やATtiny817デバイス用に作成されました。Solution Explorer(解決策エクスプローラ)は図2-9. で描かれるように新しく作成された解決策の内容を一覧にします。未だ開いていないなら、View(表示部)⇒Solution Explorer(解決策エクスプローラ)を通して、またはCtrl+Alt+Lを押すことによってアクセスすることができます。

図2-9. 解決策エクスプローラ



2.8. MCCプロジェクトのインポート

本項はオンラインのMCCツールで作成したプロジェクトをインポートする方法を記述します。1.0.248またはそれ以降版の構成設定インポート拡張がMicrochip Studioにインストールされるのを保証してください。ツールメニュー下で拡張からこれを行って更新してください。

次のここからMCCツールをアクセスしてください。

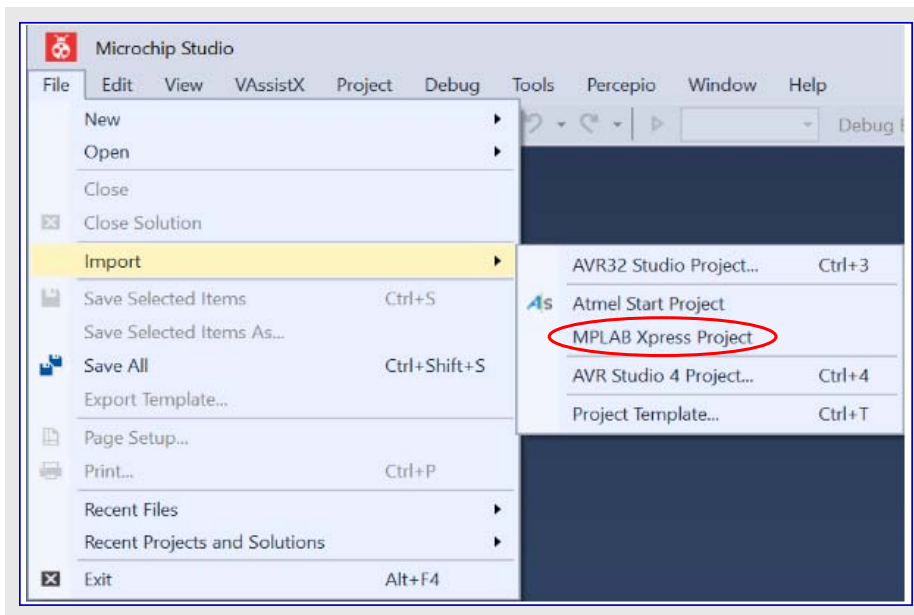
www.microchip.com/en-us/development-tools-tools-and-software/mplab-cloud-tools-ecosystem

Learn More About the Applications that Power Our Cloud Tools

MPLAB Discover	MPLAB Code Configurator	MPLAB Xpress IDE
<p>Search and Discover Content by Microchip</p> <ul style="list-style-type: none"> Universal search to discover content that is tested by Microchip Intuitive navigation based on client-focused categorization Targeted search terms to speed up discovery of specific content Code examples can be launched or accessed instantly in MPLAB Xpress IDE or Git 	<p>Race Through Code Generation For Your Project</p> <ul style="list-style-type: none"> Easy-to-use graphical configuration including point-and-click options and selections Highly optimized peripheral libraries simplify device setup Tight integration with GitHub to get modular downloads and updates Quick access through MPLAB Xpress IDE 	<p>Write Code, Share, Collaborate and Access Content</p> <ul style="list-style-type: none"> Free, fully cloud-based design environment Easy starting point for importing projects into MPLAB X IDE Projects can be shared and collaborated on using GitHub interface controls Can be used with MPLAB XC Pro compiler licenses
Access Tool	Access Tool	Access Tool

望む構成設定を行い、その後にファイルを生成してMicrochip Studioにインポートすることができるzipファイルとして保存されるプロジェクトをエクスポートしてください。

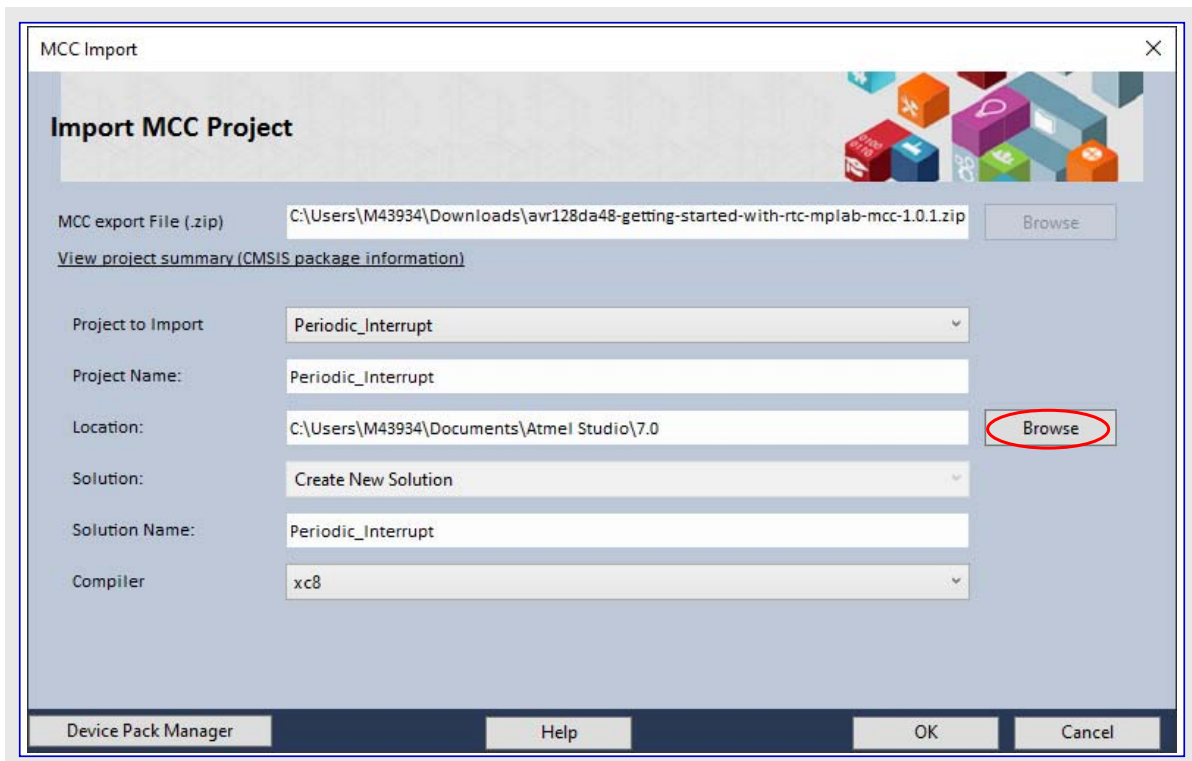
Microchip Studioを開始してImport(インポート)⇒MPLAB Xpress Project(MPLAB Xpressプロジェクト)を選んでください。



Import(インポート)ダイアログ:

MCCオンラインからエクスポートしたプロジェクトをインポートするにはこのダイアログを使ってください。Export(エクスポート)を選ぶ前にファイルを作成するためにMCCでGenerate(生成)を押すことを忘れないでください。

エクスポートしたzip形式のファイルの場所を見つけるのにBrowse(検索)を使ってください。



・ **Project to import**(インポートするプロジェクト)はzipファイルからのプロジェクト名です。1つを超えるプロジェクトが存在する場合、引き落としメニューがインポートするものを選ばせます。この領域が灰色なら、殆どの場合で1つファイルだけがあります。

・ **Project Name**(プロジェクト名)はMCCからのプロジェクト名によって設定されますが、変更することができます。

・ **Location**(位置)はMicrochip Studioが既定ですが、変更することができます。

・ **Solution Name**(解決策名)はインポートしたプロジェクトから自動で満たされますが、変更することができます。

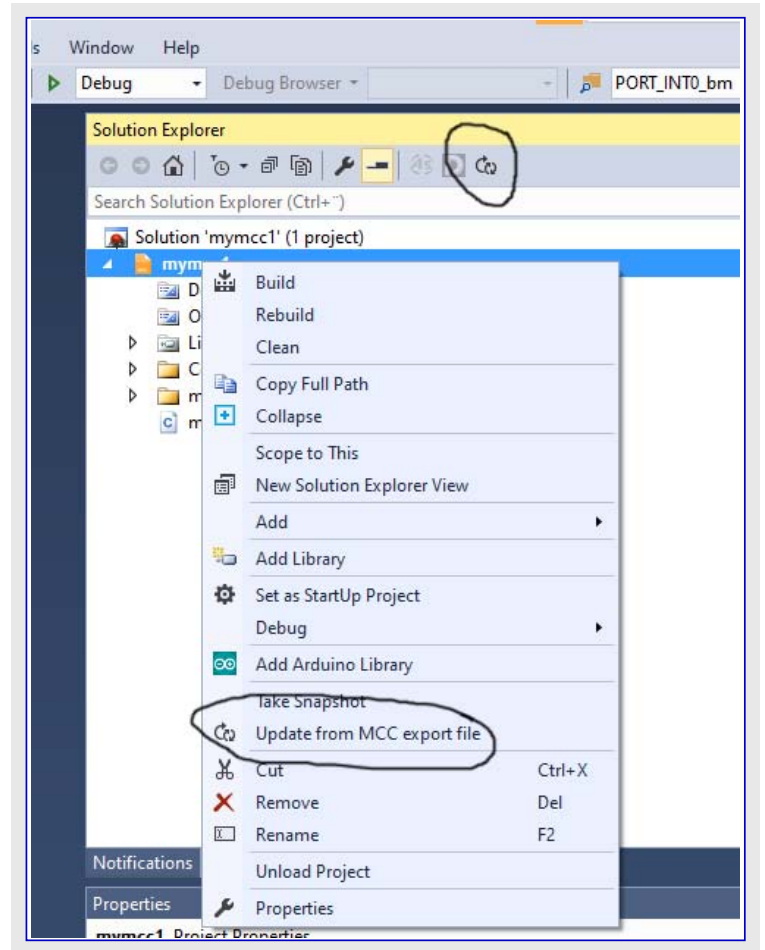
・ **Compiler**(コンパイラ)はMCCプロジェクトによって定義され、変更することができません。

プロジェクトをインポートするにはOKを押してください。

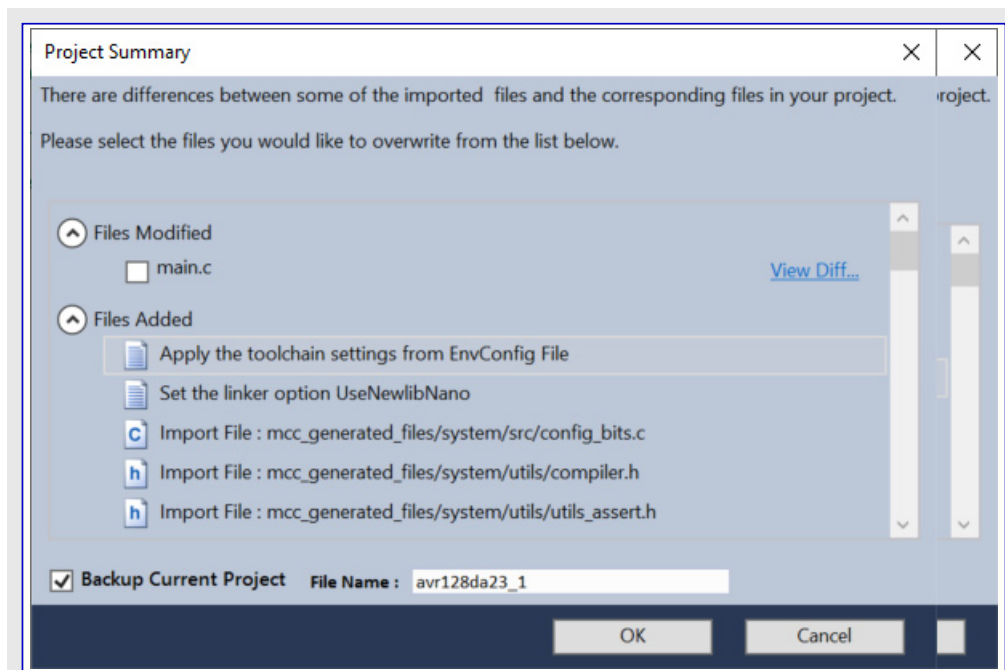
MCCでプロジェクトの再構成設定

ブラウザでMCCでの再構成設定を行ってください。プロジェクトを生成してエクスポートしてください。

ツールバーのアイコンまたは右クリックメニューのどちらかからUpdate from MCC export File(MCCエクスポートファイルから更新)を選んでください。



更新したファイルを選択してください。要約が表示されます。



プロジェクトで変更したファイルが一覧にされて選択されない限り上書きされません。変更を表示するにはView Difference(違い表示)を使ってください。


既定では".project-backup"フォルダに置かれたzipファイルが現在のプロジェクトのバックアップです。

このファイルはそれが変更を元に戻すのに必要な場合の安全策として働きます。けれども、これはgitのような版管理システムを使うことによってより良く達成されます。

2.9. Arduino®スケッチから作成

本項はArduino®スケッチから新しいMicrochip Studioプロジェクトを作成する手順を概説します。

開始に際しての話題

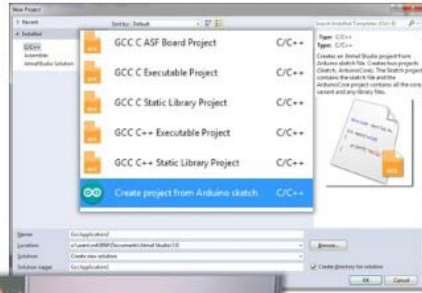



Studio 7: Create from Sketch

In this video:

Create project from Arduino® sketch file

- Sketch project, with sketch file
- Arduino core project, core & library files





映像: Arduinoスケッチから作成




すべきこと: Arduinoスケッチから新しいプロジェクトを作成します。

2.10. 実装書き込みとキット接続

この映像はキット接続を調べるためにデバイスプログラミングダイアログ枠の概要を与えます。ATtiny817 Xplained Proキットは専用の書き込み器/デバッガに対する必要性を無くす基板上の組み込みデバッガ(EDBG)を持ちます。本項はプロジェクトとEDBGを連携する手順も通って行きます。

開始に際しての話題



Studio 7: In System Programming

In this video:

Kit Autodetection

- Xplained Pro MCU & Extension Boards
- Key links

Device Programming Dialog

- Device signature & target voltage
- Tool/device information
 - Device silicon version
- Memories
 - Flash, EEPROM
- Fuses (Config bits equivalent)
- Project output files

AVR, SAM and PIC Differences

- (in terms of) Memory programming

MCU board


ATtiny817 Xplained Pro

Extension

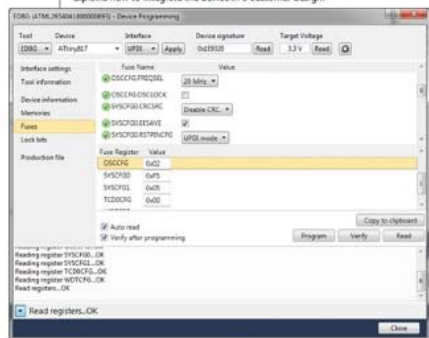
BTLC1000 Xplained Pro

I/O1 Xplained Pro

ATtiny817 Xplained Pro



The Atmel ATtiny817 Xplained Pro evaluation kit is a hardware platform to evaluate the Atmel ATtiny817 microcontroller. Supported by the Atmel Studio integrated development platform, the kit provides easy access to the features of the Atmel ATtiny817 and explains how to integrate the device in a customer design.



映像: キット接続と実装書き込み



すべきこと: プロジェクトを持つATtiny817 Xplained Proキット上のEDBGを関連付けします。

1. 提供されたマイクロUSBケーブルを使って**ATtiny817 Xplained Pro**基板をコンピュータに接続してください。下図のようにMicrochip Studioでキット頁が提示されるべきです。

図2-10. ATtiny817 Xplained Pro開始頁



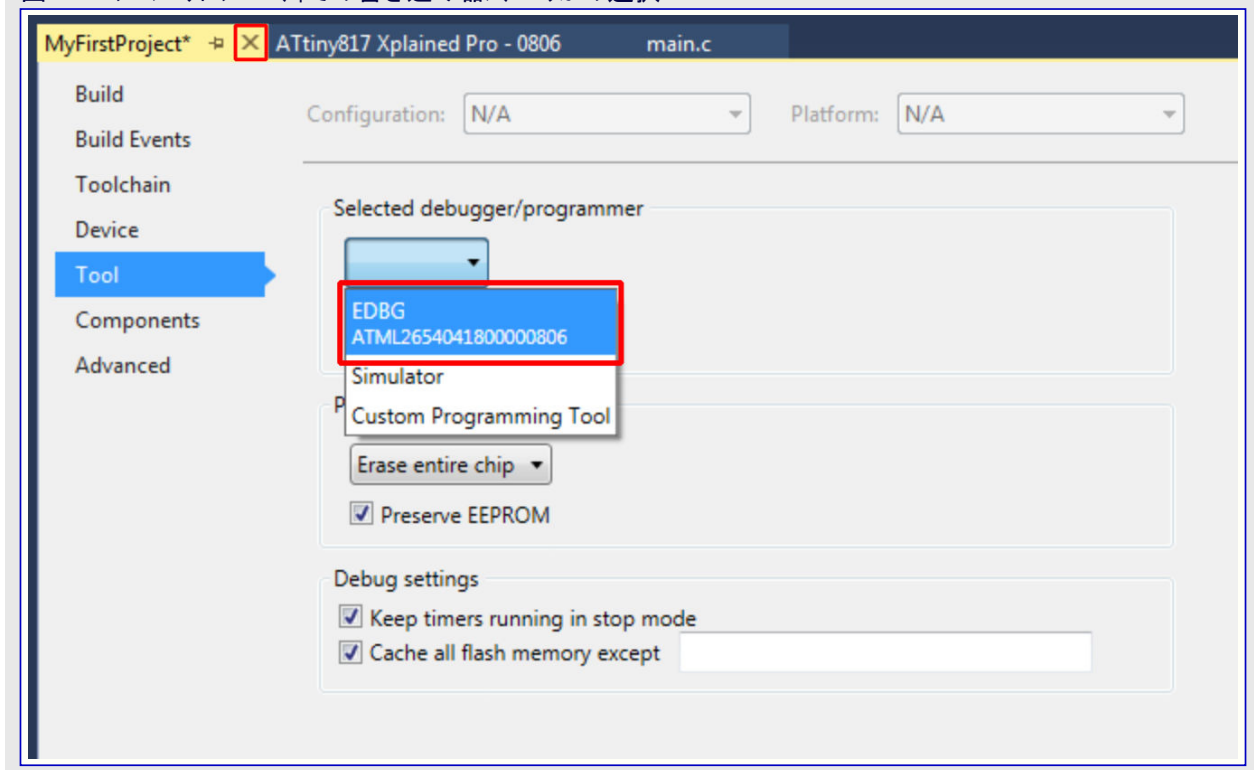
- a. 基板用の資料とデバイス用データシートへのリンクがあります。
 - b. 基板用のAtmel STARTプロジェクトを作成することが可能です。Atmel STARTリンクのプロジェクトリンクをクリックすると、この特定基板用の任意選択を得るAtmel STARTに連れてきます。
2. Tools(ツール)⇒Device Programming(デバイス プログラミング)によってProgramming(プログラミング)ダイアログを開きます。
 - a. EDBGツールを選んでDevice=ATtiny817を確実にし、その後にデバイス識票と目的対象電圧を読んでも構いません。
 - b. Interface settings(インターフェース設定)：インターフェース クロック周波数を見て変更しても構いません。
 - c. Tool infomation(ツール情報)：EDBGツールについての情報を表示
 - d. Device information(デバイス情報)：デバイスについての情報を表示。デバイスのシリコン改訂を見ることもできることに留意してください。これは顧客支援の場合に有用かもしれません。

- e. **Memories**(メモリ) : ファイルからフラッシュ メモリ、EEPROM、使用者識票を独立して書くことができます。
 - f. **Fuses**(ヒューズ) : ヒューズ、例えば、発振器周波数(16または20MHz)、低電圧検出などの読み込みと設定
 - g. **Lock bits**(施錠ビット) : メモリ施錠
 - h. **Production file**(製品ファイル) : フラッシュ メモリ、EEPROM、使用者識票を書くために製品ファイルを使ってデバイスを書き込み
 - i. AVRは**HEX**ファイルでフラッシュ メモリ、**EFP**ファイルでEEPROMを持ち、一方でPICは**HEX**ファイルで全てとヒューズさえも持ちます。
 - j. 例えば、SAML21JデバイスはEEPROMを持ちません(フラッシュ メモリで模倣できます)。デバイスを施錠する保護ビット任意選択も持ちます。
3. **File**(ファイル)⇒**New project**(新規プロジェクト)を選ぶことによる**Create a new project**(新規プロジェクト作成)で実体に対してC実行可能プロジェクトを選び、デバイス名で選別することによってデバイスを選んでください。違うプロジェクト形式は別の開始に際しての映像で検討されます。
 4. プロジェクトが選ばれたなら、下図で示されるように、ツール ダイアログを開くために上部メニュー バーに配置された**Tool**(ツール)鈕をクリックしてください。



5. **Project Properties**(プロジェクトプロパティ)の**Tool**(ツール)が開きます。引き落としメニューで下図で示されるように**EDBG**ツールを選んでください。インターフェースは自動的に統一プログラム/デバッグ インターフェース(UPDI:Unified Programming Debugging Interface)に初期化されべきです。

図2-12. プロジェクトプロパティでの書き込み器/デバッガの選択



助言: ツールの通番は引き落としメニューでのその名称と連携します。この通番は各ツールの裏側に印刷され、複数接続時の区別を許します。



助言: 次の書き込み/デバッグ作業に別のツールが使われるべきなら、これらの段階が常に繰り返され得ます。




警告 ATtiny817 Xplained Proで、EDBGは定常的に目的対象MCUに接続されますが、独自ハードウェア解決策に対してはデバッグ作業を起動し得るのに先立って目的対象デバイスが給電されて正しく接続されるのを保証することが必要です。

 **結果:** 書き込み/デバッグ作業が開始される時にMicrochip Studioによって使われるツールが今や指定されました。

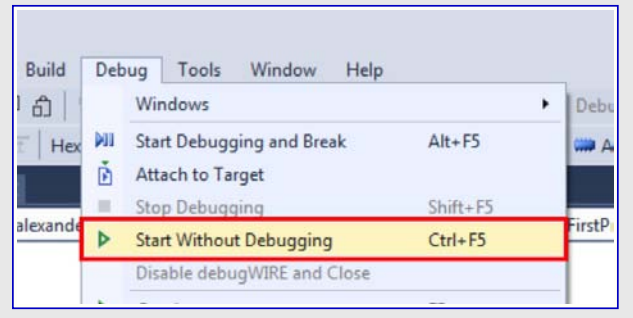
2.10.1. 設定確認

本項は空のプロジェクトをコンパイルしてそれをATtiny817に書くことによってツールとプロジェクトの構成設定構成を確認するための手引きです。


 **すべきこと:** 前項で行ったツールとプロジェクトの構成設定構成を確認します。

1. 右図で示されるように、**Debug**(デバッグ)メニューに配置された**Start Without Debugging**(デバッグなしで開始)鈕をクリックしてください。これはプロジェクトをコンパイルし、構成設定されたツールを使って指定された目的対象MCUにそれを書きます。

図2-13. デバッグなしで開始

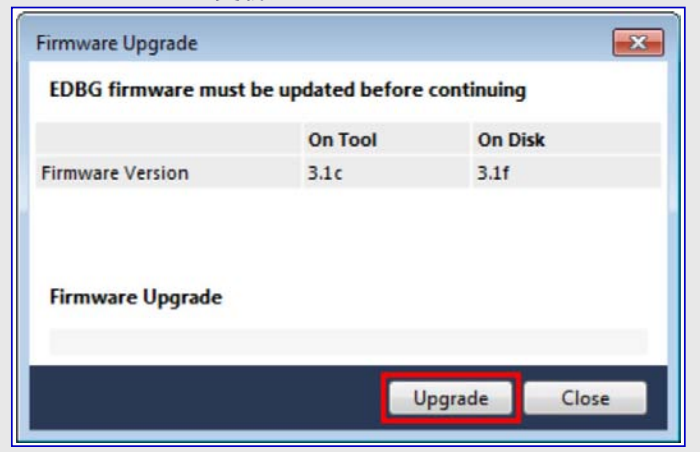



2. Microchip Studioがプロジェクトを構築する(**Start Without Debugging**(デバッグなしで開始)を押す時に自動的に行われる)と、**Solution Explorer**(解決策エクスプローラ)でいくつかの生成された出力ファイルが現れます。以下の出力ファイルが生成されます。
 - a. **EEP**ファイル : デバイスに書かれるEEPROM内容
 - b. **ELF**ファイル : プログラム、EEPROM、ヒューズを含み、デバイスに書かれる全てを含みます。
 - c. **HEX**ファイル : デバイスに書かれるフラッシュメモリ内容
 - d. **LSS**ファイル : 逆アセンブルしたELFファイル
 - e. **MAP**ファイル : リンカ情報、リンカが何を行ったか、物を置く場所についての決定
 - f. **SREC**ファイル : HEXと同じですが、Motorola形式です。

 **情報:** 選んだツールに対して利用可能なファームウェアがあった場合、**図2-14**で描かれるように、**Firmware Upgrade**(ファームウェア更新)ダイアログが現れます。ファームウェア更新を開始するには**Upgrade**(更新)鈕をクリックしてください。

接続したツールの状態と実際のファームウェア更新に依存して、更新は最初の試みで失敗するかもしれません。これは普通で、**Upgrade**(更新)を再び押す前に、キットを切断して再接続することによって解決することができます。更新完了後、ダイアログは'**EDBG Firmware Successfully Upgraded**(EDBGファームウェア更新成功)'を示すべきです。ダイアログを閉じて(**Close**)、再び**Start Without Debugging**(デバッグなしで開始)鈕をクリックすることによってキットの書き込みで新たな試みを行ってください。

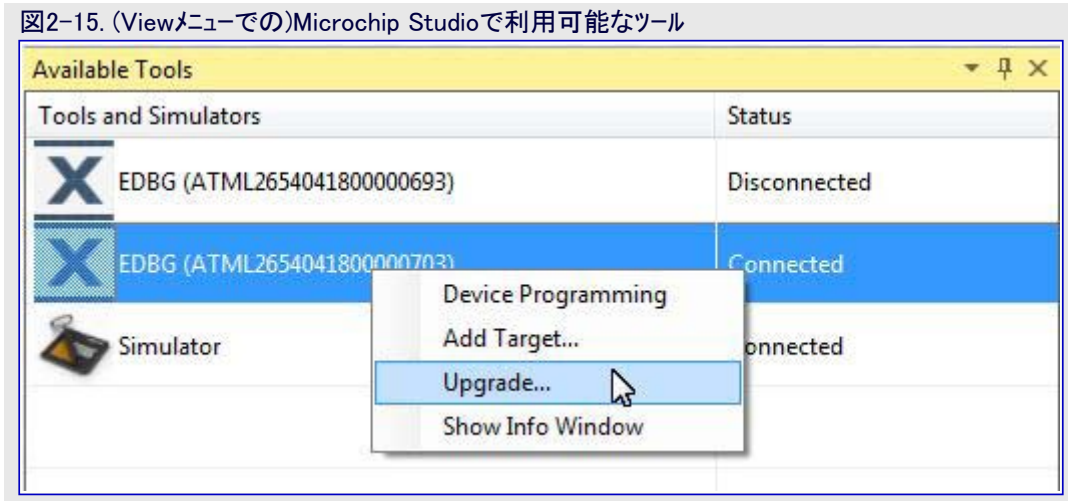
図2-14. ファームウェア更新ダイアログ



 **結果:** 空のプロジェクトをコンパイルしてATtiny817に書くことにより、以下が確認されました。

- プロジェクトは正しいMCUに対して構成設定されています。
- 正しいツールが接続されています。
- ツールのファームウェアは最新です。

View(表示部)⇒Available Tools(利用可能なツール)下で、利用可能なツールや最近使ったツールの一覧を見ることができます。ここでツールに対するファームウェア更新をMicrochip Studioに特別に尋ねることができます。



2.11. I/O表示部と他の空からのプログラミング参考資料

本項はソフトウェア構成設定ツールや枠組みと無関係に、即ち空からMicrochip Studioで一般的にコード書く方法を記述します。これは(下でリンクされる)映像と実践資料の両方として網羅されます。主な焦点は各々の関連プログラミング参考資料で、各々がどうアクセスされるか、各々が何に使われるかです。プロジェクトの脈絡はLEDをONにしてその後に遅延と共に点滅することです。ATtiny817 Xplained Proが使われ、この原理はMicrochip Studioで支援される殆どのデバイスに適用するにも関わらず、この原理はMicrochip Studioでどのキットと共に使うのにも充分一般的です。

[開始に際しての話題](#)

Studio 7: I/O View & Bare-Metal Prog. Refs.

In this video:

Context:

- Turn on LED, then blink with delay.

Programming References:
(How to easily access & what to use each for)

- Device datasheet
- Datasheet (from IO view)
- IO view (debugging)
- Kit user-guide & schematics
- Device header files
- Editor (Visual Assist)
- AVR® LibC
- Atmel START

映像: I/O表示部と空からのプログラミング参考資料

下の一覧は一般的に使われるプログラミング参考資料の概要です。特定の重点物はI/O表示部に置かれ、これは編集やデバッグの時にデータシートのレジスタ説明を誘導することだけでなく、デバッグ時に現在の構成設定を理解することの方法を提供します。デバッグ時のI/O表示部のこの2つ目の使用は新しいレジスタ構成設定の試験にも使われます。

この話題は「2.16. デバッグ3: I/O表示部、メモリ表示部と監視」だけでなく「2.12. エディタ: コードの記述と整理 (Visual Assist)」の両方に対して密接に関連します。

- デバイスのデータシート
- (I/O表示部からの)データシート
- キットの使用者の手引きと回路図
- I/O表示部(デバッグ)
- エディタ (Visual Assist)

- デバイスのヘッダ ファイル
- AVR Libc (AVR特有)
- Atmel START: ATtiny817プロジェクト

その過程で以下のコードが書かれます。このコードが簡単とはいえ、前のプログラミング参考資料の一覧を使い、決定過程が記述されま

```
#include <avr/io.h>
#define F_CPU 3333333
#include <util/delay.h>

int main(void)
{
    PORTB.DIR = PIN4_bm;

    while (1)
    {
        _delay_ms(500);
        PORTB.OUTTGL = PIN4_bm;
    }
}
```

警告 main.cの先頭に#include <avr/io.h>行を保つことに注意してください。このヘッダ ファイルは選んだデバイス用の正しいレジスタ割り当てをインクルードし、この行なしではコンパイラが上のコードで参照されたどのマクロも認知しません。

デバイスのデータシート (PDF)

I/O表示部がレジスタレベルでデータシートに誘導するための容易なアクセスを許すとは言え、PDF版は未だ役割を持ちます。デバイスのデータシートはPDF形式で、少なくとも**構成図**と**機能的な説明**を通して周辺機能の理解を得るために使われがちです。例えば、ATtiny817のPORT周辺機能を理解するために我々はデータシートの**PORT構成図**と**機能的な説明**⇒**動作**⇒**基本機能**項を調べました。これら2つの項は共に説明を図に繋げることでPORT周辺機能の基本的な理解を与えます。

図2-16. PDFデータシートからのPORT構成図

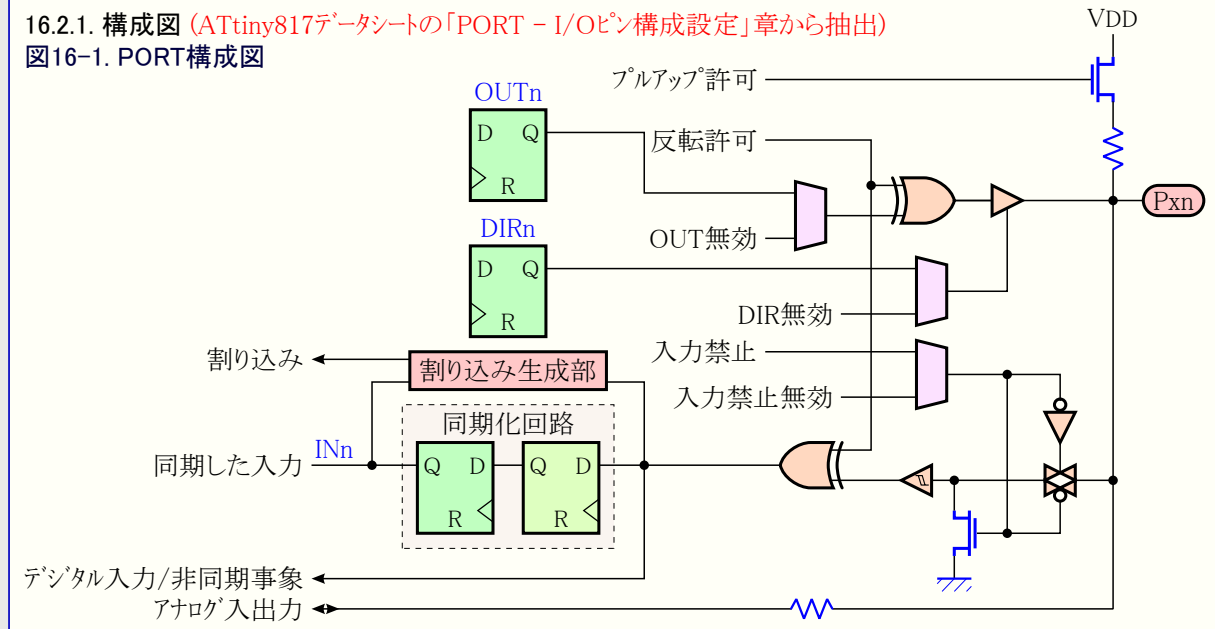


図2-17. ATtiny817のPDFデータシートからの基本機能

16.3. 機能的な説明

16.3.2. 動作

16.3.2.1. 基本機能

各入出力(P_{xn})ピンはPORT_x内のレジスタによって制御することができます。各ピン群(x)はそれ自身のPORTレジスタ一式を持ち、(n)ピン用のレジスタ式の基準アドレスはバイトアドレスでPORT+\$10です。そのレジスタ式内の指標はnです。

出力専用としてピン番号nを使うには、**データ方向(PORT.DIR)レジスタ**のビットnに'1'を書いてください。これは**データ方向設定(PORT.DIRSET)レジスタ**のビットnに'1'を書くことによっても行うことができ、これはその群内の他のピンの構成設定の妨害を避けます。**出力値(PORT.OUT)レジスタ**のビットnは望む出力値が書かれなければなりません。

同様に、**出力値設定(PORT.OUTSET)レジスタ**のビットへの'1'書き込みはPORT.OUTレジスタの対応するビットを'1'に設定します。**出力値解除(PORT.OUTCLR)レジスタ**のビットへの'1'書き込みはPORT.OUTレジスタの対応するビットを'0'に解除します。**出力値切り替え(PORT.OUTTGL)**または**入力値(PORT.IN)**のレジスタのビットへの'1'書き込みはPORT.OUTレジスタ内のそのビットを論理反転します。

ピンを入力として使うには出力駆動部を禁止するためにPORT.DIRレジスタのビットnが'0'を書かれなければなりません。これは**データ方向解除(PORT.DIRCLR)レジスタ**のビットnに'1'を書くことによっても行うことができ、これはその群内の他のピンの構成設定の妨害を避けます。入力値は**ピン制御(PORT.PINnCTRL)レジスタ**の**入力/感知構成設定(ISC)ビット**が入力禁止(INPUT_DISABLE)に設定されない限り、PORT.INレジスタのビットnから読むことができます。

注: 我々は周辺機能構成図だけでなく、**PORT DIR**と**OUT**のレジスタの説明に対してもデバイスのデータシートを使用しました。

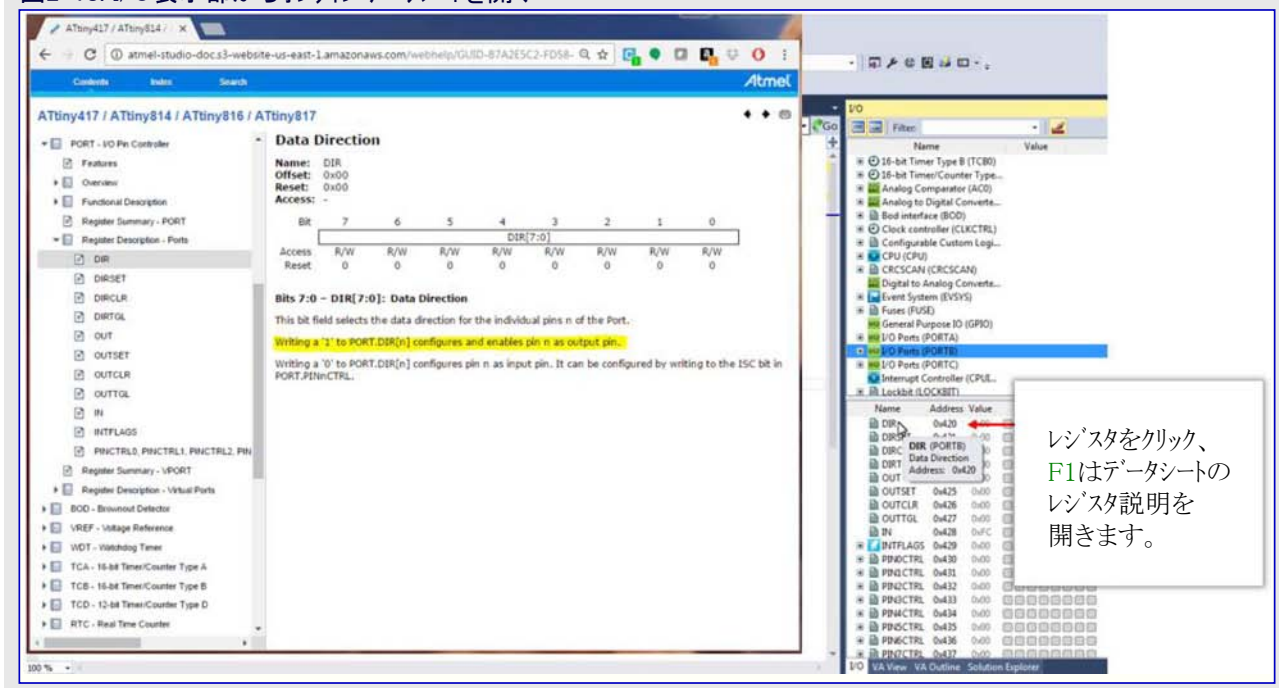
I/O表示部データシート

Microchip Studioは関連レジスタ記述でF1を押すことによってデータシートのレジスタ説明の容易なアクセスを許します。データシートのHTML版が(既定によって)オンラインで開きます。データシートは関連レジスタ記述の脈絡で開きます。

注: この方法でそれを理解するために我々は**Data sheet from I/O View**(I/O表示部からのデータシート)を使います。

1. PORT.DIR_nへの'1'書き込みはピンnを出力として構成設定して許可します。
2. OUT_nが'1'を書かれた場合、ピンnはLowを駆動します。

図2-18. I/O表示部からオンラインデータシートを開く



I/O表示部(デバッグ)

この機能はStart Debugging and Break(デバッグ開始と中断)を用いてデバッグ作業を開始することによって直接試験することができます。故に今や次の画像で示されるように機能的な試験を始めることができます。

I/O表示部は「2.16. デバッグ3: I/O表示部、メモリ表示部と監視」でもっと詳細に網羅されます。

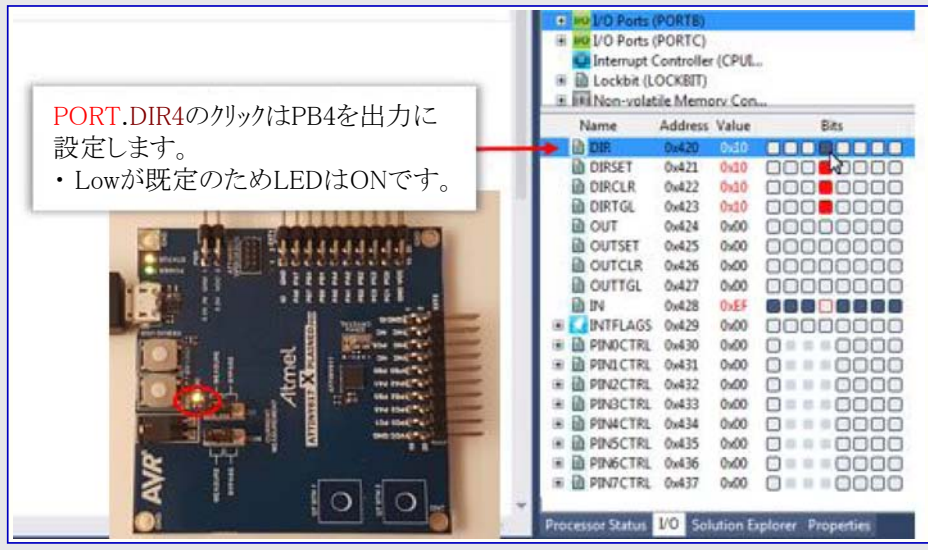
注: デバッグ時のI/O表示部は以下に使われます。

1. PORT.DIR₄に'1'を書き、LEDをONにするためにピンを出力、既定によってLowとして設定するのを確認。
2. PORT.OUT₄に'1'を書き、LEDがOFFになるのを確認。

表2-2. Microchip Studio 釦機能 (プログラミングとデバッグ作業開始)

釦	機能	キーボード ショートカット
	デバッグを開始して中断	Alt + F5
	目的対象に取り付け	
	デバッグを開始	F5
	全て中断	Ctrl + Alt + Break
	デバッグなしで開始	Ctrl + F5

図2-19. デバッグ時にI/O表示部レジスタ操作を通してキットのLEDをON/OFF切り替え



Microchip Studio 資料ダウンロード

データシートはMicrochip Studioのヘルプ システムを使うことによってもダウンロードすることができます。この場合、同様の機能がオフラインで動きます。これは「2.4.2. オフライン資料のダウンロード」で記述されます。

Microchip Studio エディタ (Visual Assist)

Visual Assistを装備したMicrochip Studioのエディタはコードを書いて整理するだけでなく、大きなプロジェクトの容易な誘導も手助けする協力的な機能を持ちます。示唆機能は図2-20.で示され、一方でコード誘導の概要は図2-21.で示されます。次の「2.12. エディタ: コードの記述と整理 (Visual Assist)」項ではエディタ機能がもっと詳細に網羅されます。

図2-20. コード書きに対するMicrochip Studioのエディタでの示唆機能

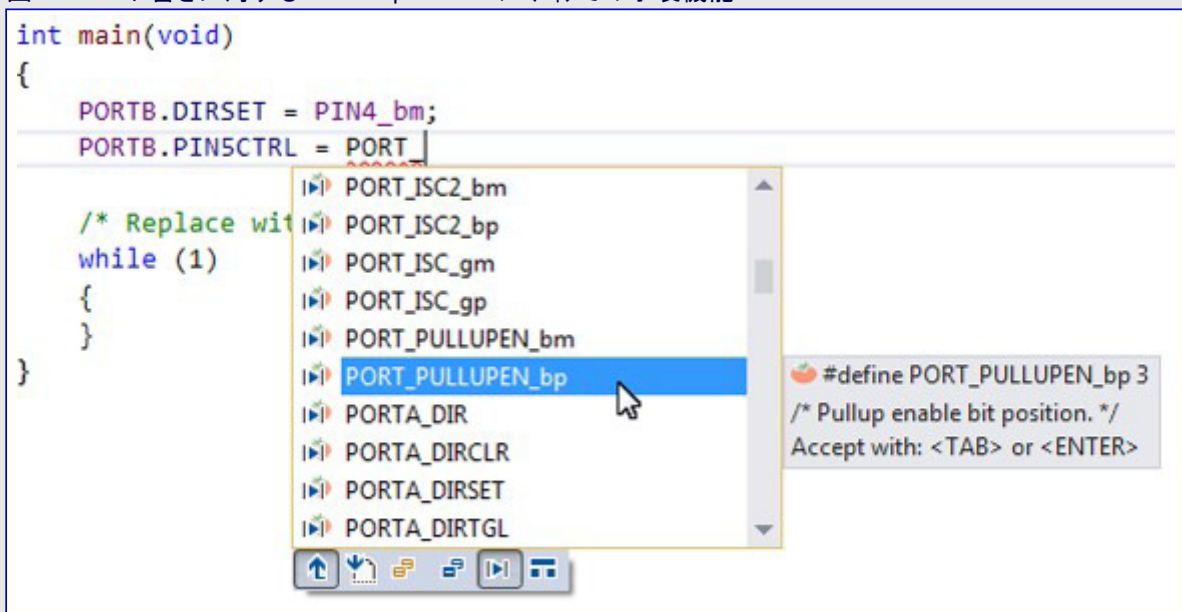


図2-21. Microchip Studio エディタの誘導概要

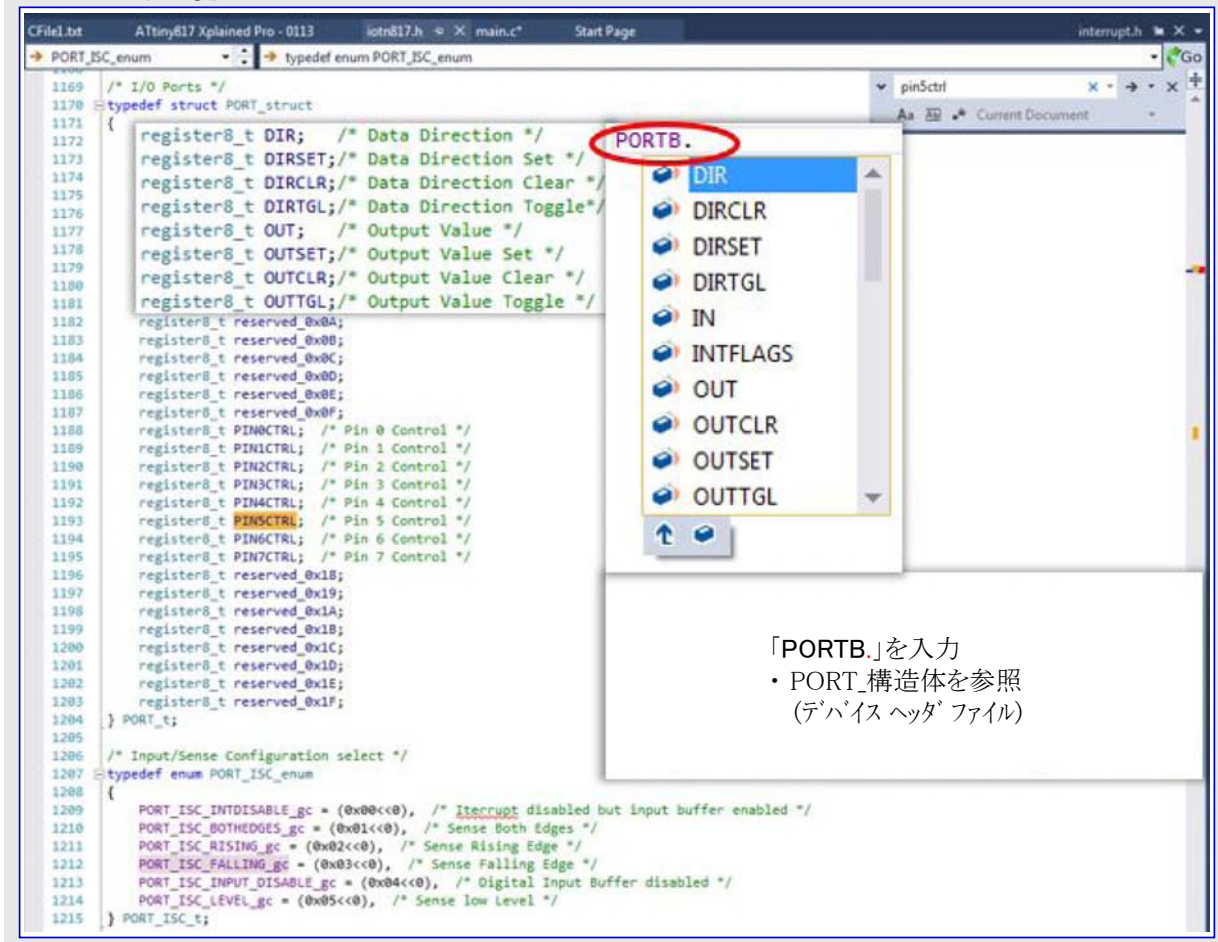


この項に関連する映像では具体的にエディタが以下のために使われます。

デバイスヘッダファイル

エディタの**移動定義**機能を通して、即ち、どれかのレジスタをクリックしてその後に移動(Go)鈕をクリックするか、Alt+Gを入力することにより、MCUデバイスヘッダファイルにアクセスすることが容易です。PORTB.を書くことが「[図2-22. 示唆一覧とMCUデバイスヘッダファイル](#)」で示される、PORT構造体から潜在的なレジスタの示唆一覧を与えます。AVRヘッダファイルがどう構成されるかについてのより多くの情報に関しては[AVR1000応用記述](#)をご覧ください。

図2-22. 示唆一覧とMCUデバイスヘッダファイル



キット回路図と使用者の手引き

キット回路図と使用者の手引きはキットのMCUピン接続を理解するのに有用です。完全な回路図とガーバーのようなキット設計ファイルはwww.microchip.comのキットの製品頁で入手できます。

図2-23. 特定開発基板用回路図の探し方

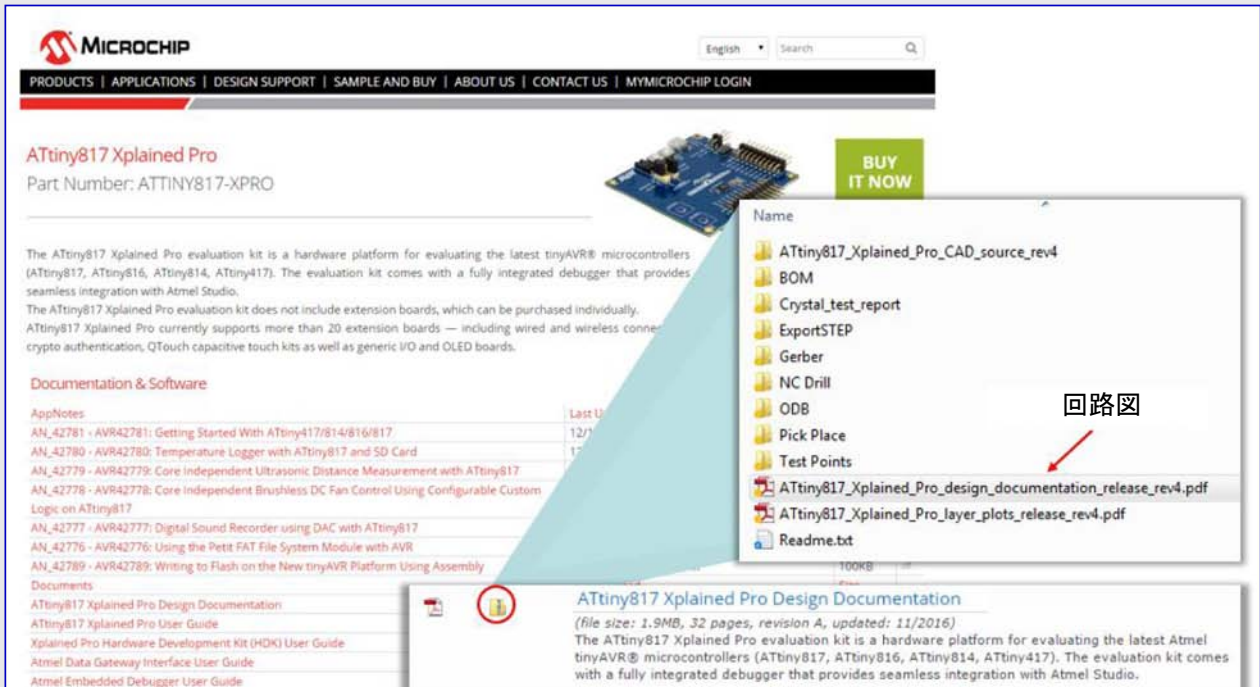
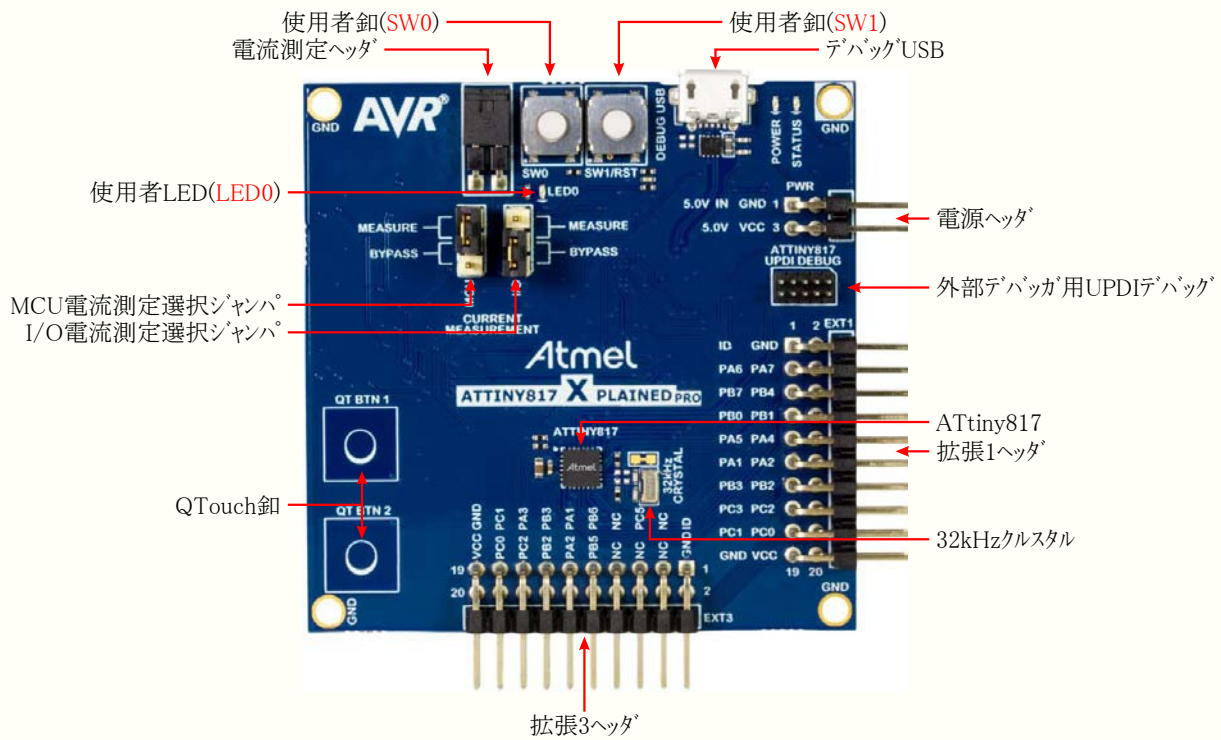


図1-1. ATtiny817 Xplained Pro評価キット概要



LEDと釦はATtiny817 Xplained Pro使用者の手引きから右表のようにピンへ接続されます。

表2-3. ATtiny817 Xplained Pro GPIO接続

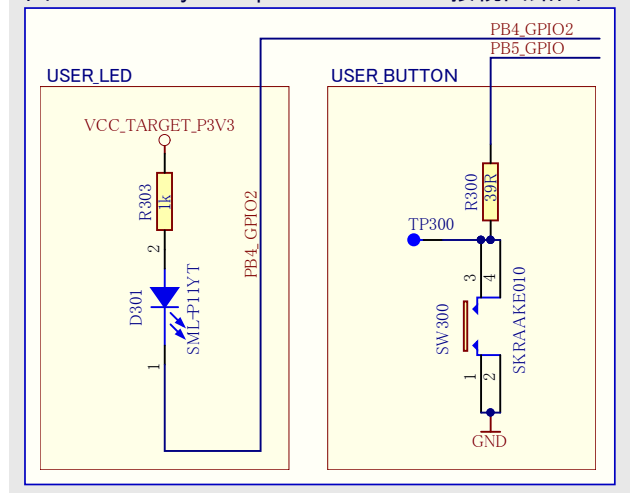
シルク文字	ATtiny817 GPIOピン
LED0	PB4
SW0	PB5

ATtiny817 Xplained Pro設計資料回路図は右図のようにLEDと鉤に対する接続を示します。

この回路図からは以下が断定されます。

- LEDはPB4をLowに駆動することによってONにすることができます。
- SW0は直接GNDへと電流制限抵抗を通してPB5に接続されます。
- SW0は外部プルアップ抵抗を持ちません。
- SW0はATtiny817の内部プルアップが許可された場合に押下時に'0'、解放時に'1'として読みます。

図2-24. ATtiny817 Xplained Pro GPIO接続回路図



AVR® Libc

この点までに網羅された全ての言及がAVRに関しては単にSAMに関連するだけですが、これは名前が示唆するようにAVRに対して特有です。AVR LibcはAVRマイクロコントローラでGCCと共に使うための高品質Cライブラリを提供するのを目標とする無料のソフトウェア事業です。avr-binutils、avr-gcc、avr-libcは共にAVRマイクロコントローラ用の無料ソフトウェアツールチェーンの心臓部を形成します。更に、それらは実装書き込みソフトウェア(avrdude)、シミュレーション(simulavr)、デバッグ(avr-gdb, AVaRICE)の事業も伴います。

図2-25.で示されるように、ライブラリ参考基準(Library Reference)は通常、AVR Libcへの迅速な遡り取り部です。関連ライブラリに対して頁を迅速に検索することができます。プロジェクトに追加されるべき関連ヘッダファイルが単位部(Module)名で示されます。例えば'interrupts(割り込み)'を検索すると、関連インクルードは#include <avr/interrupt.h>でしょう。単位部内でクリックすると、図2-26.で示されるように、利用可能な関数と関連割り込み呼び戻しの一覧を見つけることができます。

図2-25. AVR® Libcライブラリ参照基準

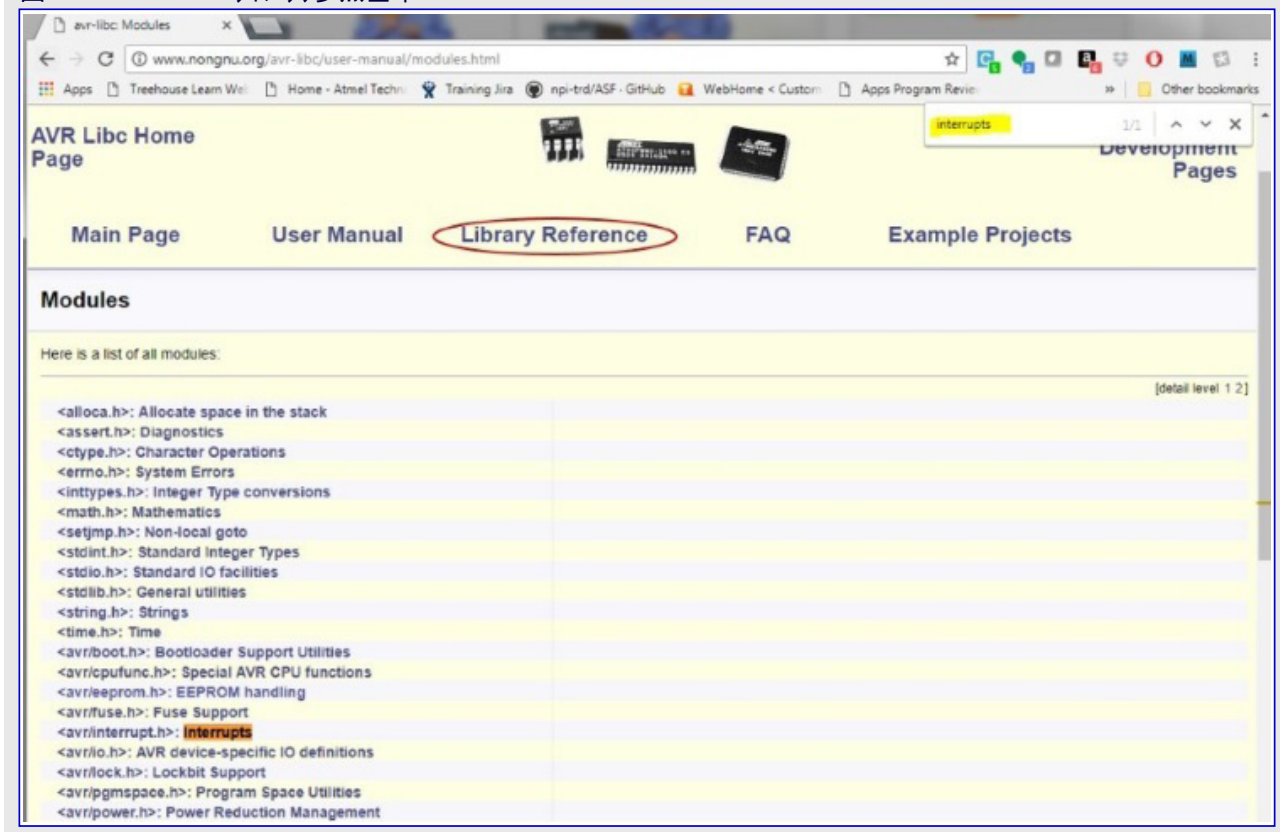


図2-26. AVR® Libcでの割り込みの使い方

<avr/interrupt.h>: Interrupts

Global manipulation of the interrupt flag

The global interrupt flag is maintained in the I bit of the status register (SREG).

Handling interrupts frequently requires attention regarding atomic access to objects that could be altered by code running within an interrupt context, see <util/atomic.h>.

Frequently, interrupts are being disabled for periods of time in order to perform certain operations without being disturbed, see **Problems with reordering code** for things to be taken into account with respect to compiler optimizations.

#define sei() Enables interrupts by setting the global interrupt mask.

#define cli()

Macros for writing interrupt handler functions

#define ISR(vector, attributes)
 #define SIGNAL(vector)
 #define EMPTY_INTERRUPT(vector)
 #define ISR_ALIAS(vector, target_vector)
 #define reti()
 #define BADISR_vect

ヘッダ ファイル追加
 関連IRQヘッダ

```
#include <avr/interrupt.h>
ISR(ADC_vect)
{
  // user code here
}
```

Atmel START

Atmel STARTは様々なソフトウェア枠組みに対してMCU開発の開始を助けるウェブに基づくソフトウェア構成ツールです。新しいプロジェクトまたは例プロジェクトのどちらかから始めると、Atmel STARTは使用に便利で最適化した規則で組み込み応用を眺めるためにドライバやミドルウェアのようなソフトウェア構成部品を(ASF4とAVR Codeから)選んで構成することを許します。一旦最適化されたソフトウェア構成が行われると、生成されたコードプロジェクトをダウンロードしてMicrochip Studio、MPLAB X、IAR Embedded Workbench、Keil µVisionを含み、選んだIDEにそれを開くか、または単にmake-fileを生成することができます。

Atmel STARTがMCUとソフトウェアを構成するための道具とは言え、それは空からの開発、即ち、この項で記述されるプログラミング参照基準の一覧を使って0からコードを書くことにさえ未だ有用で有り得ます。PINMUX表示部を用いて、使うキット用の新しいプロジェクトを作成することは有用な代替で有り得ます。加えて、CLOCKS表示部はデバイスの既定クロックを調べるのに有用で有り得ます。更に、構成コードを見ることで、有用な部分をあなたのプロジェクトに貼り付けることができます。例えば、図2-29.で示されるように、AVR Libc遅延関数はクロック周波数で定義されることが必要とされます。ATtiny817に対してこの既定値は#define F_CPU 3333333です。

図2-27. 関連基板用新規プロジェクト作成にAtmel STARTを使用

Atmel START

CREATE NEW PROJECT

Select device or board before creating a new project. You can filter devices and boards by what software you need and also with hardware requirements such as memory sizes.

FILTERS

HARDWARE

SEARCH FOR SOFTWARE

Find software...

MIDDLEWARE

- + Bootloader
- + Crypto

DRIVERS

- AC 0
- ADC 0
- CRC 0
- DAC 0

RESULTS

817 Show all Show only boards Show only devices

Name	Architecture	Package	Pins	Flash	SRAM
ATtiny817-MNRES	AVR	VQFN24	24	8 KB	512 B
ATtiny817-MNR	AVR	VQFN24	24	8 KB	512 B
ATtiny817-MFR	AVR	VQFN24	24	8 KB	512 B
Tiny817 QTouch Moisture Demo					
ATtiny817 Xplained Pro					
ATtiny817 Xplained Mini					

6 of 817 boards and devices

CREATE NEW PROJECT

図2-28. キット回路図に対する代替としてAtmel STARTで基板分類表示

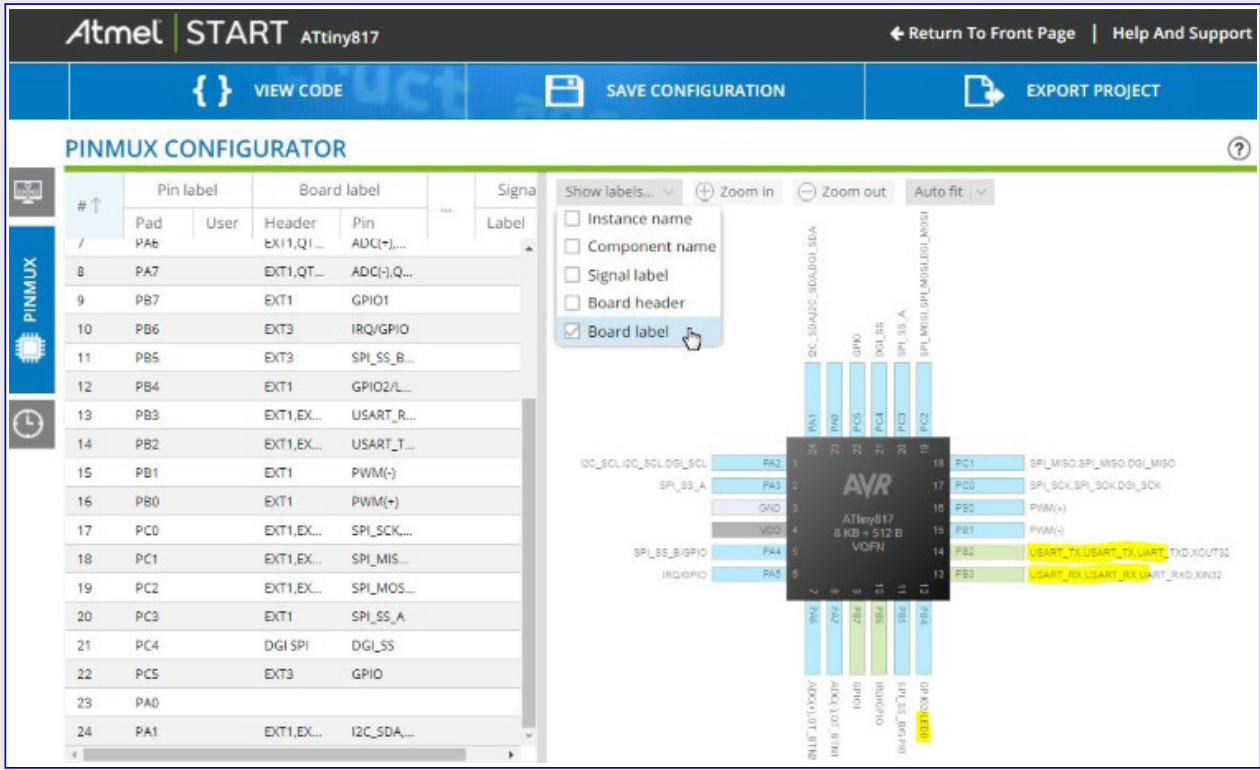
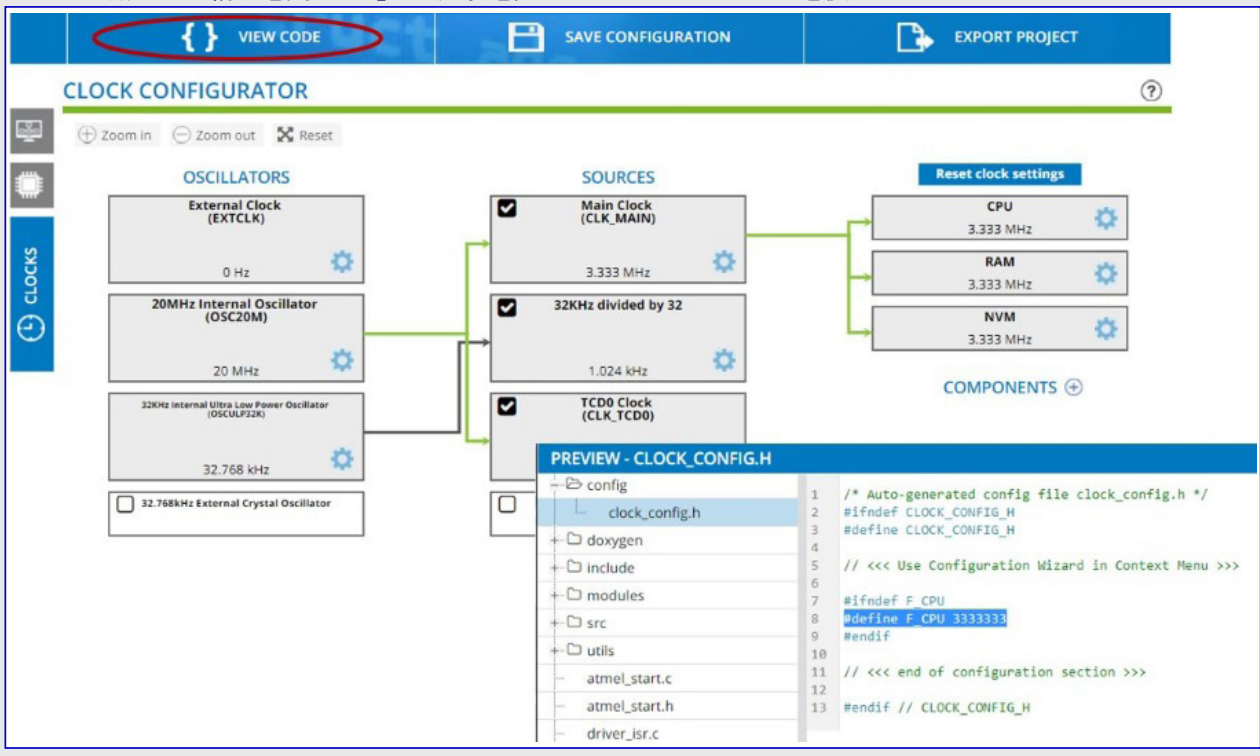



図2-29. 既定クロック構成を調べてF_CPU定義を見つけるのにVIEW CODEを使用



2.12. エディタ: コードの記述と整理 (Visual Assist)

Microchip StudioのエディタはCとC++のコードを書いて読んで整理して誘導するための生産性道具であるVisual Assistと呼ばれる拡張によって強化されます。

開始に際しての話題



Studio 7: Editor (Visual Assist)

In this video:

Studio 7 Editor...

Context:

- Turn on LED, when switch pressed
- Polled, then with pin change IRQ

Writing Code

- Suggestion lists, enhanced list boxes
- Visual assist code snippets

Refactoring Code

- Extract method
- Introduce variable
- Contextual rename

Header File Navigation

- Finding enumerators to configure bit groups

```
int main(void)
{
    PORTB.DIR = PIN4_bm;
    PORTB.PIN5CTRL = pp

```

- PORT_PULLUPEN_bm
- PORT_PULLUPEN_bp
- PORTA_PIN5CTRL
- PORTB_PIN5CTRL
- PORTC_PIN5CTRL

#define PORT_PULLUPEN_bm 0x08
/* Pullup enable bit mask. */
Accept with: <TAB> or <ENTER>

VAssistX	ASF	Project	Build	Debug	Tools	Window
Open File in Solution...						Shift+Alt+O
Open Corresponding File (h/cpp, aspx/cs)						Alt+O
List Methods in Current File						Alt+M
Find Symbol...						Shift+Alt+S
Find References						Shift+Alt+F
Find References in File						
Clone Find References Results						
Find Previous by Context						
Find Next by Context						
Goto Implementation						Alt+G
Goto Related						Shift+Alt+G

映像: Microchip Studio エディタ (Visual Assist)

- 「2.11. I/O表示部と他の空からのプログラミング参考資料」から基本的な機能で開始すると、main.cは以下のコードを持ちます。

```
#include <avr/io.h>

int main(void)
{
    PORTB.DIR = PIN4_bm;

    while (1)
    {
    }
}
```

ATtiny817 Xplained Pro設計資料回路図は右図のようにLEDと鉤に対する接続を示します。

この回路図からは以下が断定されます。

- LEDはPB4をLowに駆動することによってONにすることができます。
- SW0は直接GNDへと電流制限抵抗を通してPB5に接続されます。
- SW0は外部プルアップ抵抗を持ちません。
- SW0はATtiny817の内部プルアップが許可された場合、押下時に'0'、解放時に'1'として読みます。

図2-30. ATtiny817 Xplained Pro GPIO接続回路図

© 2021 Microchip Technology Inc.とその子会社

使用者の手引き

DS50002718D - 43頁

2. 示唆一覧と強化された一覧枠を用いてPORTB5のプルアップを許可してください。示唆一覧は頭字語を支援し、故に'pp'と入力すると、PORT_PULLUPENが先頭示唆になることに注意してください。

```
int main(void)
{
    PORTB.DIR = PIN4_bm;
    PORTB.PIN5CTRL = pp

```

3. けれども、Enterを打つ前に、最初に'POR'を入力してその後にCtrl+Spaceを打ってください。これは可能な全ての任意選択を持つ強化された一覧枠を提示します。

今や、下図で示されるように、入力によって示唆を選別することが可能です。

```
int main(void)
{
    PORTB.DIR = PIN4_bm;
    PORTB.PIN5CTRL = por

```

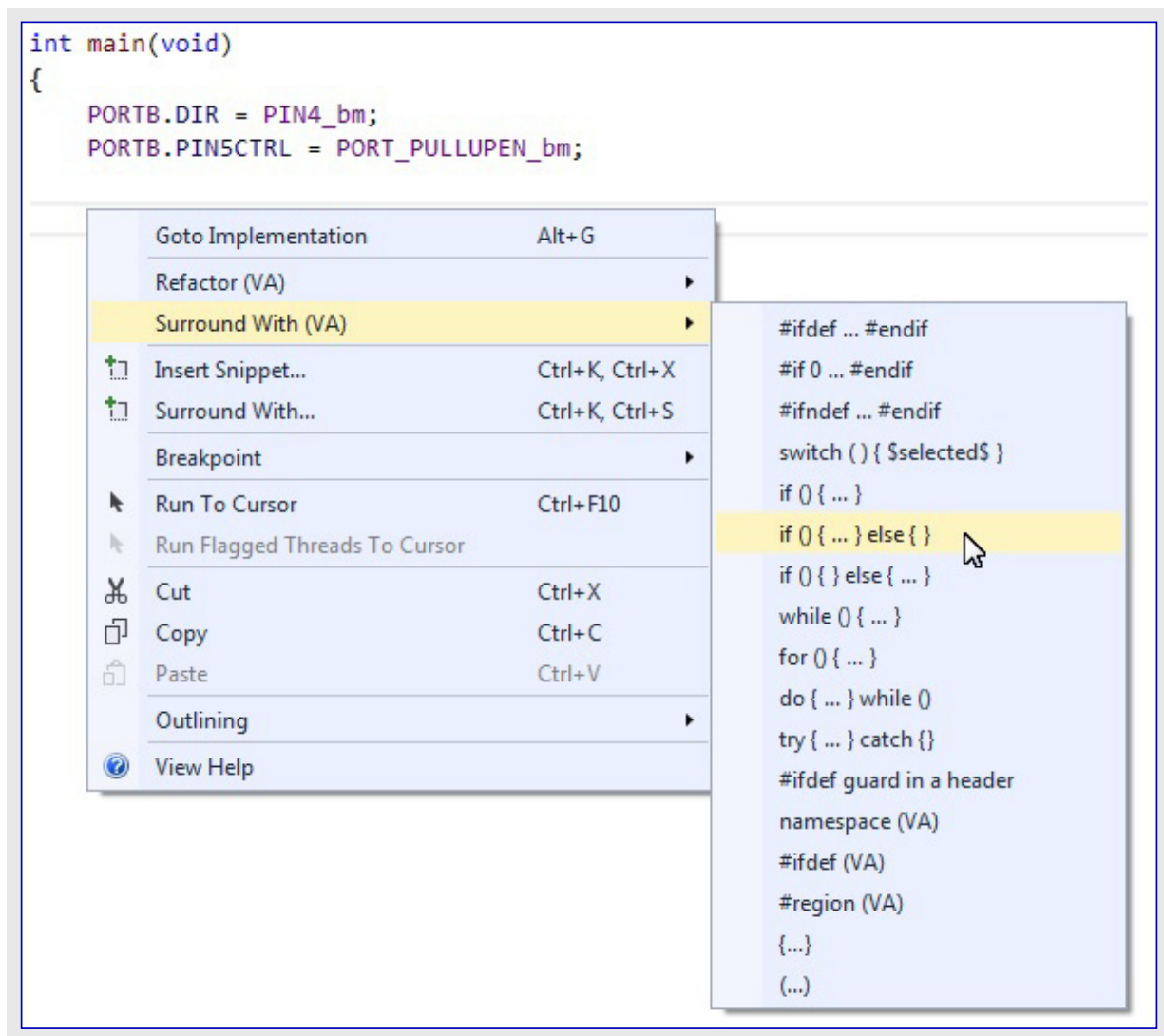
Enumまたはtypedef
Enum/typedefのパラメータ
#Define
typedef構造体 Extern(全域変数)

4. if(...){...} Visual assistコード断片を用いてSW0が押されたかを確認してください。単なる'if'入力が任意選択を提示します。または右クリックして断片の完全な一覧を与えるSurround With (VA)を選ぶことができます。これは編集可能な一覧で、故に使用者自身の断片を追加することができます。

```
int main(void)
{
    PORTB.DIR = PIN4_bm;
    PORTB.PIN5CTRL = PORT_PULLUPEN_bm;

    if

```



5. `if(){...}else{...}`条件としてスイッチが押されたかを確認し、押された場合にLEDをONにそうでない場合にOFFにしてください。main.cは今や次のように見えるべきです。

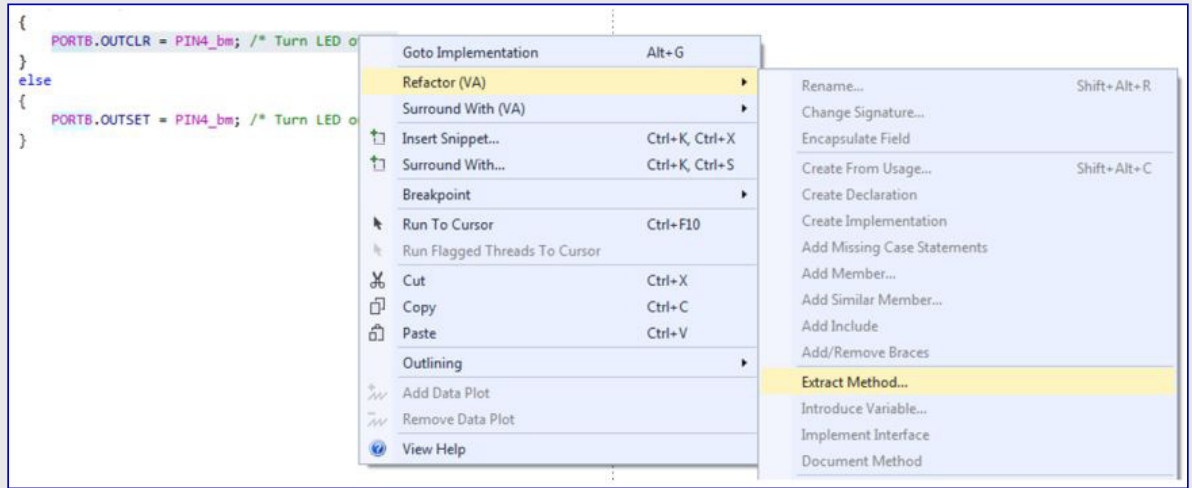
```
#include<avr/io.h>

int main(void)
{
    PORTB.DIRSET    = PIN4_bm;           /* LEDピンを出力として構成設定 */
    PORTB.PIN5CTRL = PORT_PULLUPEN_bm; /* SW0ピンに対してプルアップを許可 */

    while(1)
    {
        if (!(PORTB.IN & PIN5_bm))      /* スイッチ状態調査 */
        {
            PORTB.OUTCLR = PIN4_bm;     /* LEDをONに切り替え */
        }
        else
        {
            PORTB.OUTSET = PIN4_bm;     /* LEDをOFFに切り替え */
        }
    }
}
```

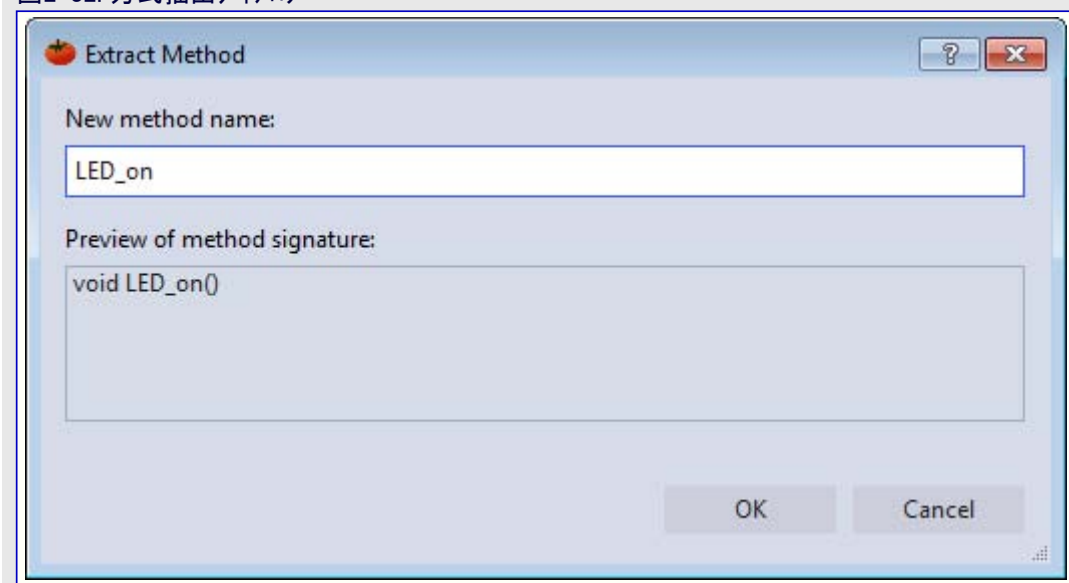
6. SW0押下時にLED0が点灯することを確認してください。ATtiny817 Xplained ProキットでSW0押下時にLED0が点灯することを確認するため、Start Without Debugging(デバッグなしで開始) (Ctrl+Alt+F5)をクリックすることによってコードを走らせてください。今や基本的な機能が整ったので、もっと読み易くするためにコードを整理しましょう。
7. Refactor(整理)⇒Extract Method(方式(メソッド)抽出)を使ってLED_in()とLED_off()の関数を作成してください。LEDをONにするコードの行はSW0が押された時に実行されます。コードのこの行を(選択して)強調表示し、右クリックして下図で示されるようにそこへ行ってください。

図2-31. 方式抽出



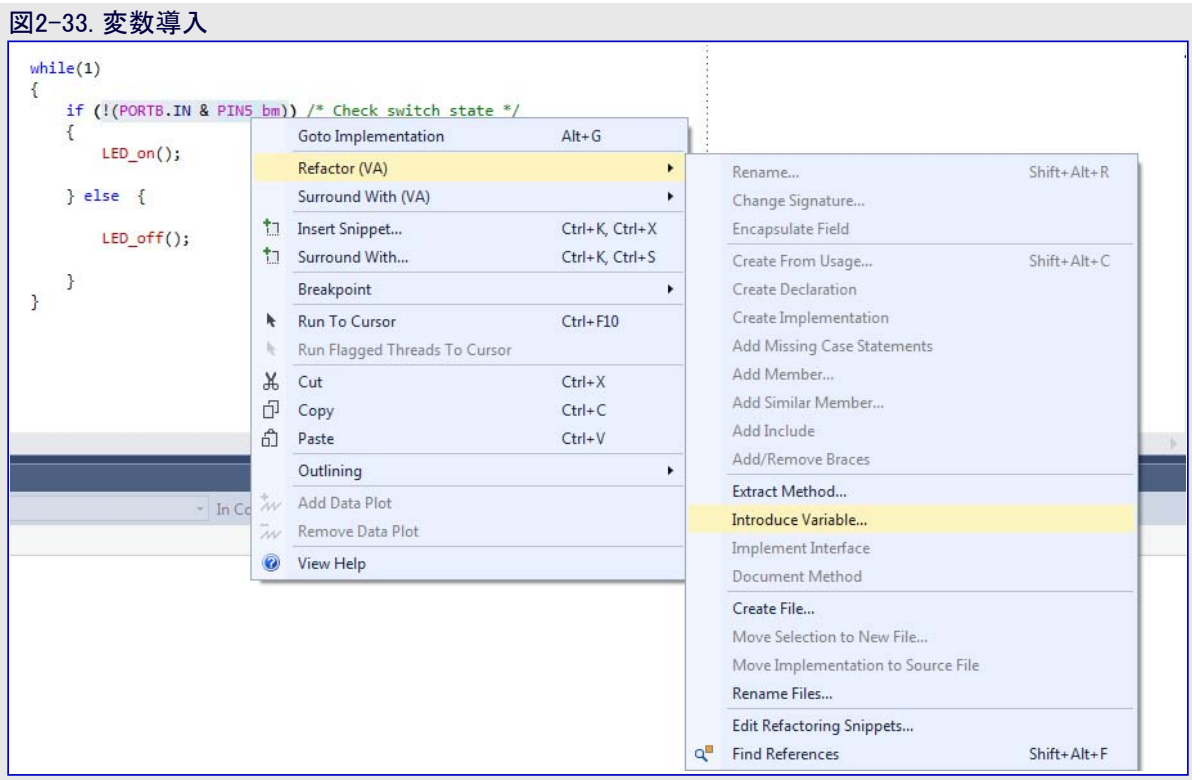
Extract Method(方式抽出)ダイアログが現れます。下図で示されるように、関数を'LED_on'と名付けてください。

図2-32. 方式抽出ダイアログ

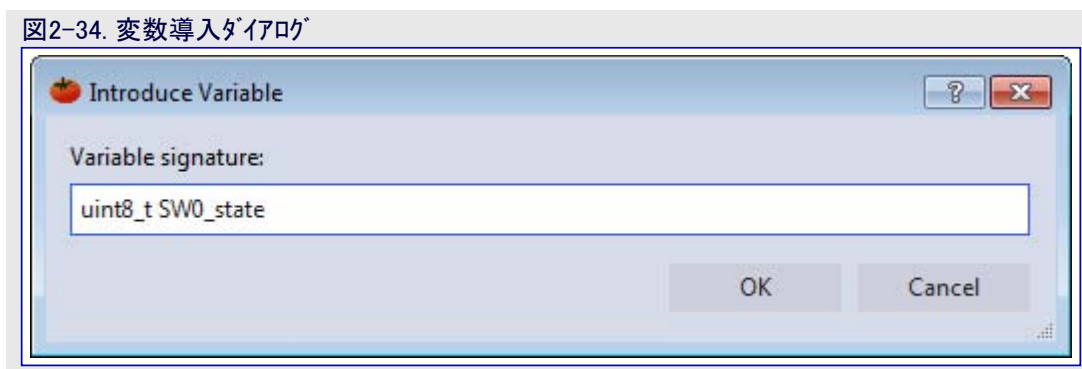


OKをクリックし、コードが変更されるべきです。使われるべきコードの行の場所での関数呼び出しと共に、LED_on()と呼ばれる新しい関数がファイルの先頭に現れるでしょう。LED_off()を実装するのに同じ方法を使ってください。

8. Refactor(整理)⇒Introduce Variable(変数導入)を使ってSW0の状態用変数を作成してください。次に、SW0の状態用の変数を作成する必要があります。main()のwhile (1)繰り返しでif()内側の条件を(選択して)強調表示にしてください。右クリックして下図で示されるように、そこへ行ってください。



- 図2-34. で描かれるように、Introduce Variable(変数導入)ダイアログが現れます。変数を 'uint8_t SW0_state' と名付けてください。



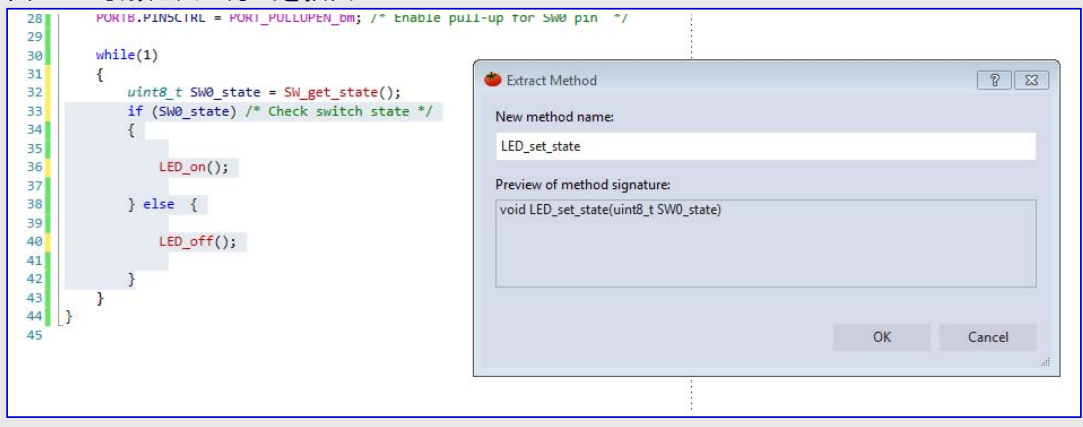
- 助言:** プール値で扱うのに追加のヘッダをインクルードするのを避けるため、uint8_tに対する戻り値を自動的に生成したboolに変更してください。

OKをクリックし、コードが変更されるべきです。下のコードの塊で示されるように、if()文内側の条件は今やその上の行の変数に対して割り当てられた変数を参照すべきです。

```
while (1)
{
    uint8_t SW0_state = !(PORTB.IN & PIN5_bm);
    if (SW0_state)
    {
        LED_on();
    }
    else
    {
        LED_off();
    }
}
```


9. Refactor(整理)⇒Extract Method(方式(メソッド)抽出)を使ってSW_get_state関数を作成してください。SW0_state文の右側を選択してSW_get_stateに対する方式を抽出してください。
10. void LED_set_state(uint8_t state)関数を実装してください。方式を抽出してください。図2-35.で示されるように、Microchip Studioは引数SW0_stateを検出します。

図2-35. 引数と共に方式を抽出



OKをクリックし、コードが変更されるべきです。今や、LED状態を設定するための独立した方式(メソッド)があります。

11. void LED_set_state(uint8_t state)関数でRefactor(整理)⇒Rename(改名)を使ってSW0_stateを改名してください。より大きな応用ではこの関数がSW0の情態と無関係な状況でLEDの状態を設定するのに使われるかもしれません。Microchip Studioは脈絡での改名の能力があり、故にこの機能は引数を容易に改名して混乱を避けるのに使うことができます。図2-36.で示されるように、LED_set_state()関数内で、SW0_state変数を右クリックしてRefactor(整理)⇒Rename(改名)に行ってください。

図2-36. 脈絡での改名

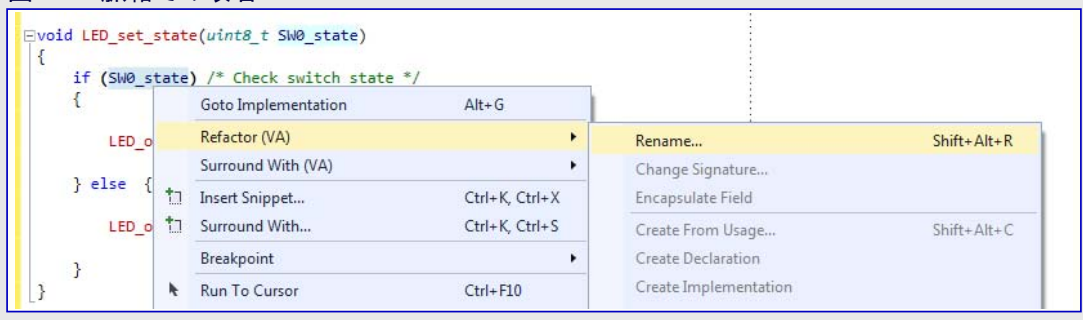
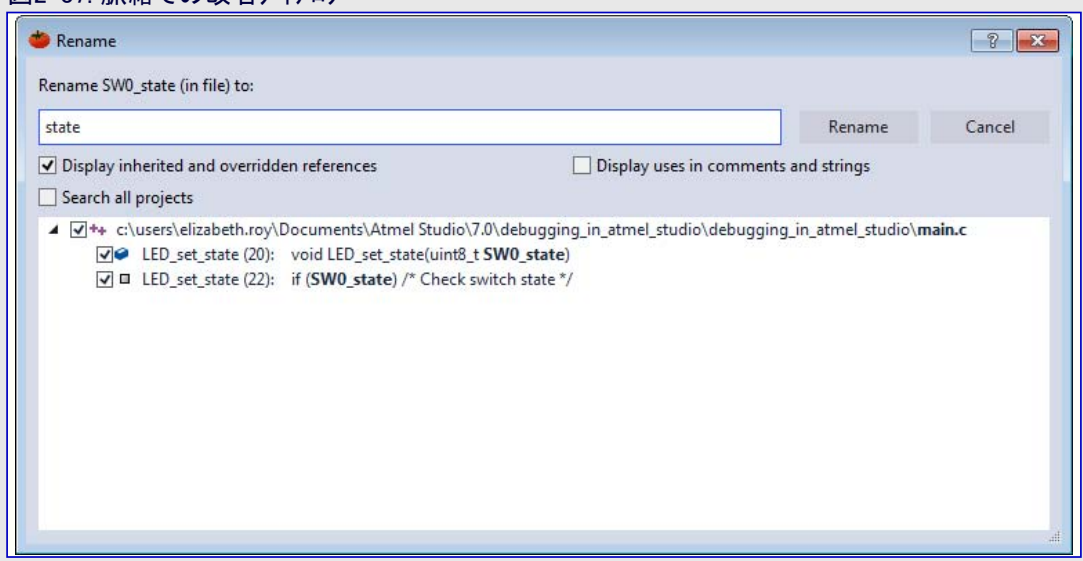


図2-37.で描かれるように、Rename(改名)ダイアログが現れます。SW0_state変数を'state'に改名してください。Microchip Studioは選択されたものと同じ脈絡を持つ変数の全ての存在を検出し、それが一覧で提示され、個別に選択または解除を行うことができます。

図2-37. 脈絡での改名ダイアログ



Rename(改名)をクリックし、コードが変更されるべきです。LED_set_state()の引数と関数内のその全ての参照が改名され、けれどもmain()のSW0_stateへの参照が同じに留まることに気付いてください。

12. 作成した関数をmain()の下に移動して関数定義を作成してください。main.cは今や次のように見えるべきです。

```
#include <avr/io.h>

void LED_on(void);
void LED_off(void);
void LED_set_state(uint8_t state);
uint8_t SW_get_state(void);

int main(void)
{
    PORTB.DIRSET = PIN4_bm;           /* LEDピンを出力として構成設定 */
    PORTB.PIN5CTRL = PORT_PULLUPEN_bm; /* SW0ピンに対してプルアップ許可 */

    while(1)
    {
        uint8_t SW0_state = SW_get_state(); /* スイッチ状態読み込み */
        LED_set_state(SW0_state);          /* LED状態設定 */
    }
}

uint8_t SW_get_state(void)
{
    return !(PORTB.IN & PIN5_bm); /* スイッチ状態読み込み */
}

void LED_off(void)
{
    PORTB.OUTSET = PIN4_bm; /* LEDをOFF */
}


void LED_on(void)
{
    PORTB.OUTCLR = PIN4_bm; /* LEDをON */
}

void LED_set_state(uint8_t state)
{
    if (state)
    {
        LED_on();
    }
    else
    {
        LED_off();
    }
}
```

2.13. AVR®シミュレータ デバッグ

本項は(シミュレータでのみ利用可能な)Cycle Counter(周期計数器)、Stopwatch(ストップウォッチ)のようなAVRシミュレータの鍵となる機能の使用と基本的なデバッグ(中断点(ブレークポイント)設定とコードを通じた段階実行)を実演します。割り込みを模倣する方法も示します。

開始に際しての話題



Studio 7: AVR® MCU Simulator Debugging

In this video:

Studio 7: AVR MCU Simulator Project Setup:

- Modify project from Studio 7 Editor video
- Basic debugging: set breakpoint, step, ...
- Processor view:** Demonstrate use of cycle counter & stop watch
- Dissassembly view:** difference how code compiled
- Simulate IRQ (IO view)

Context:

- Set up 3 options to clear, then set register bit
- LED on when switch pressed (using pin change IRQ).

Program Counter	0x0000028
Stack Pointer	0x3FFD
X Register	0x0000
Y Register	0x3FFF
Z Register	0x0000
Status Register	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Cycle Counter	2
Frequency	1.000 MHz
Stop Watch	2.00 µs
Registers	

```

#include <avr/io.h>

int main(void)
{
    PORTB.DIR &= ~PIN4_bm;
    PORTB.DIR |= PIN4_bm;

    //PORTB.DIRSET = PIN4_bm;
    //PORTB.DIRCLR = PIN4_bm;

    //VPORTB.DIR &= ~PIN4_bm;
    //VPORTB.DIR |= PIN4_bm;

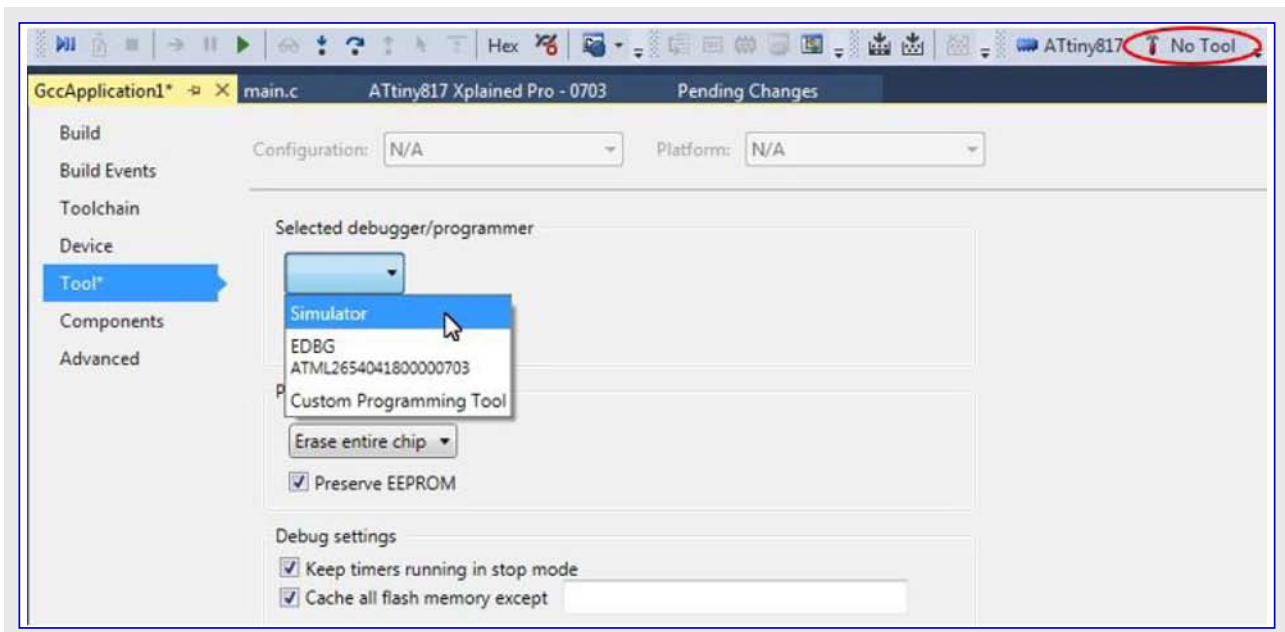
    while (1)
    {
    }
}

```


映像: AVRシミュレータ デバッグ

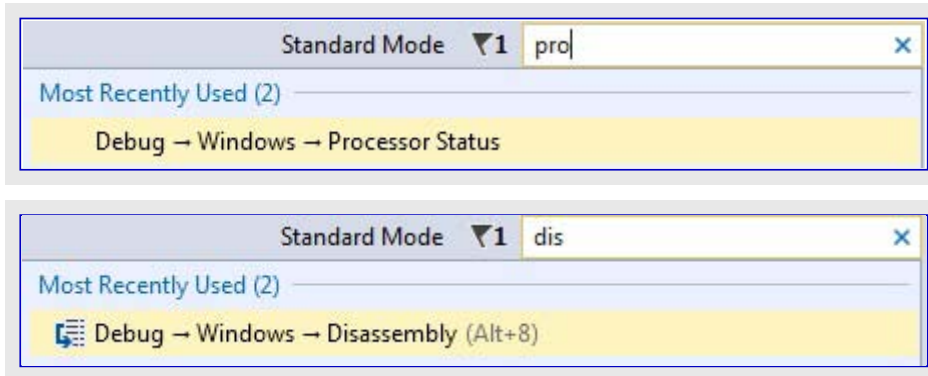
上の映像で使われたコードは「2.12. エディタ: コードの記述と整理 (Visual Assist)」の映像で書かれました。

プロジェクトとシミュレータを関連付けるため、ツールアイコンの  をクリックし、その後に Simulator(シミュレータ)を選んでください。



The screenshot shows the Microchip Studio IDE interface. The 'Tool' dropdown menu is open, showing options: Simulator, EDBG, ATML2654041800000703, and Custom Programming Tool. The 'Simulator' option is selected. In the top right corner of the IDE, the 'No Tool' button is circled in red.

Cycle Counter(周期計数器)とStopwatch(ストップウォッチ)はシミュレータでだけ利用可能です。これらを使うには最初にデバッグ作業を開始するためにStart Debugging and Break(デバッグを開始して中断)  をクリックし、その後に敏速起動バーに'Processor'を入力してEnterを打つことによってProcessor Status(プロセッサ状態)ウィンドウを開いてください(また、これはDebug(デバッグ)⇒Windows(ウィンドウ)⇒Processor Status(プロセッサ状態)下で見つけることができます)。同様に、Disassembly(逆アセンブリ)ウィンドウを開くこともできます。



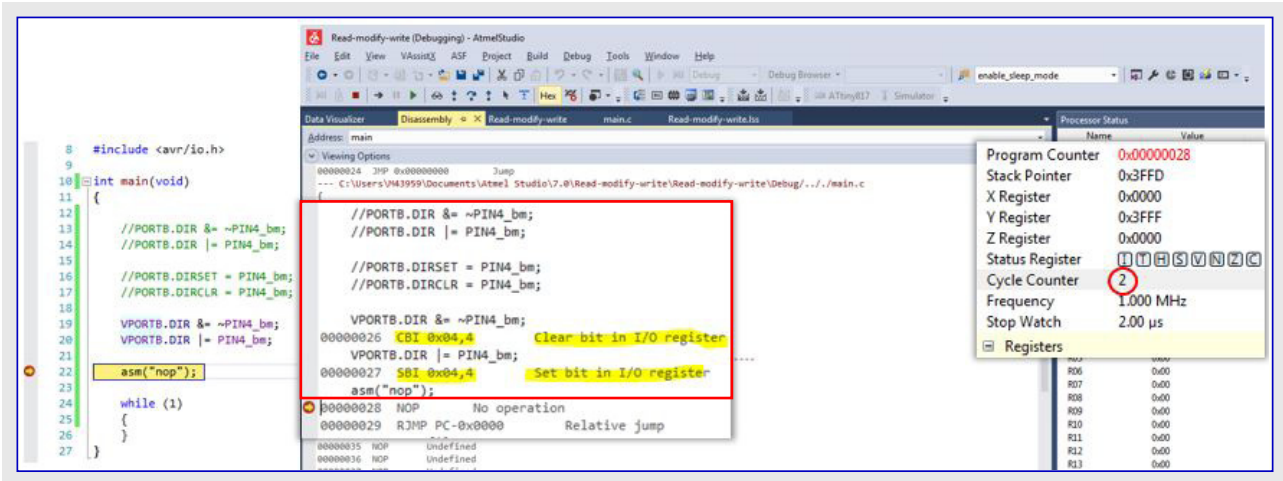
AVRシミュレータは実デバイスを作るのに使われるのと同じRTLコードに基づく模式を使っています。これはバグと遅延の両方に対してCycle Counter(周期計数器)を正確にします。Stopwatch(ストップウォッチ)はFrequency(周波数)に関連され、それは値上のダブルクリックと使いたいクロック周波数を入力することによって設定することができることに注意してください。

Name	Value
Program Counter	0x00000380
Stack Pointer	0x00003821
X Register	0x0002
Y Register	0x3FF1
Z Register	0x3814
Status Register	I T H S V N Z C
Cycle Counter	8186
Frequency	1,000 MHz
Stop Watch	8 186,00 us

実行されつつある命令のアドレス
現在のスタックポインタ値
模倣開始からの経過周期数
周期数と周波数に基づく経過時間

Cycle Counter(周期計数器)は値上をクリックして0を入力することによってリセットすることができます。Processor Status(プロセッサ状態)ウィンドウ内の値はI/O表示部と同様にプログラム中断毎に更新されます。その後に中断点(ブレークポイント)まで走ります。

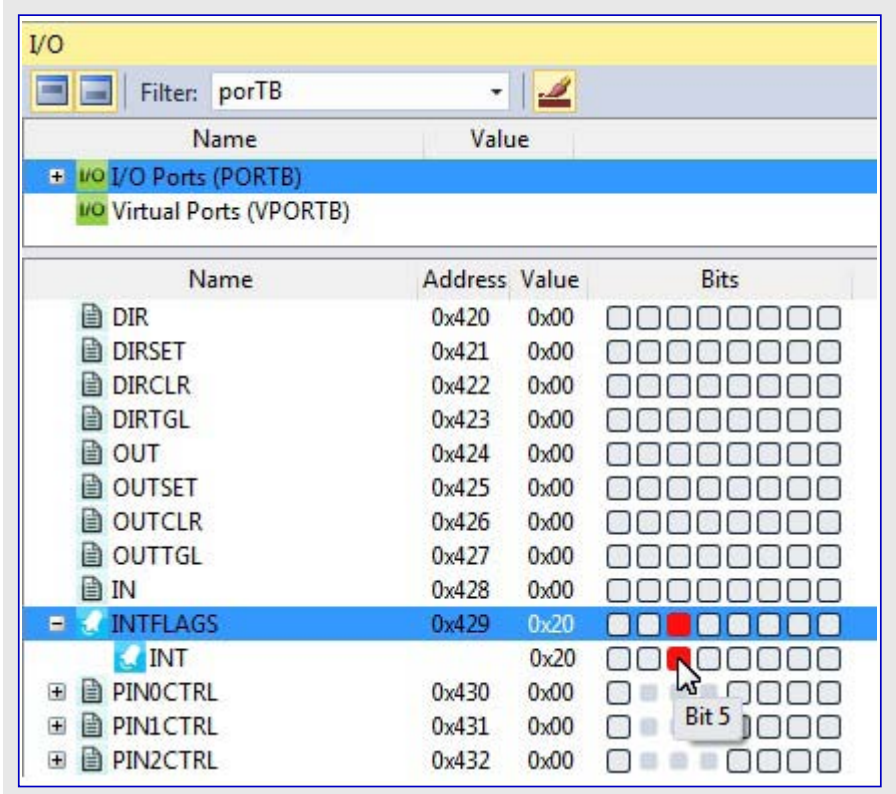
(訳補:周期カウンタの違いとなる)仮想ポートレジスタでのソフトウェア読み-変更-書き(前図)とビット操作命令(次図)間で生成されたアセンブリコードでの違いに気付いてください。



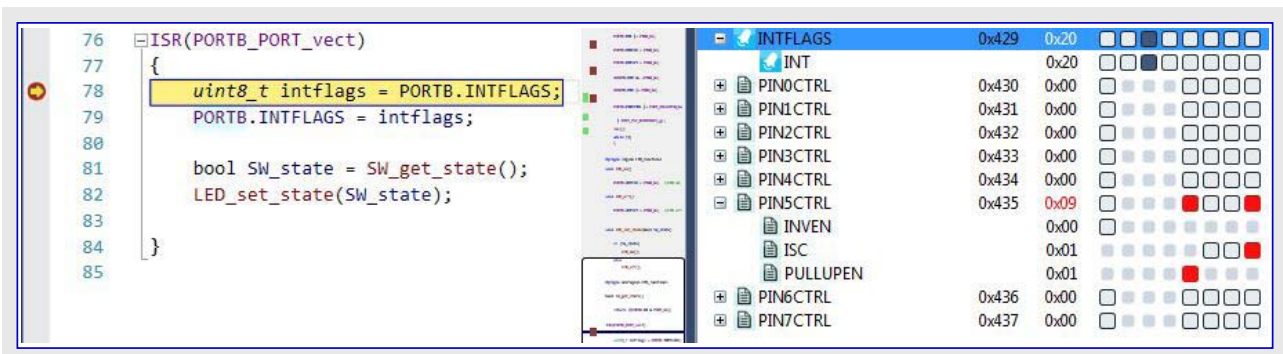
これら3つの方法を比べた結果が右表で要約されます。

方法	周期数	注釈
ソフトウェア読み-変更-書き	10	
ハードウェア読み-変更-書きレジスタ	5	(割り込み安全)非分断命令
仮想ポートでのビットアクセス命令	2	(割り込み安全)非分断命令、実に速い

次に、ピン変化割り込みを模倣したいと思います。デバッグ時にI/O表示部で関連する割り込み要求フラグを設定(1)することによってこれを行うことができます。



下で示されるように割り込み(ISR)に的中します。以降のコードでPORT.PIN5CTRLへ書くことで示されるように、やはり割り込みが許可されるのが必要なことに注意してください。



ピン変化割り込みはI/O表示部でポート入力レジスタに書くことによって起動することもできます。ポート入力レジスタへのビット書き込みはデバイス外圍器の物理ピンへその値を印加するのと同じです。内部ポート論理回路はその後にそれによってこれが構成設定されていた場合に割り込みを起動します。

Microchip Studioの標準デバッグ機能の殆どがシミュレータ使用時に利用可能で、それらの機能はチップ上デバッグ能力が欠けていてハードウェアデバッグを用いてデバッグすることができないデバイスでも利用可能です。この開始の手引きのデバッグ章をご覧ください。

(ATtiny817用に書かれた)AVR[®]シミュレータを実演するのに使われるコード

```
#include <avr/io.h>
#include <stdbool.h>
#include <avr/interrupt.h>

void LED_on();
void LED_off();
bool SW_get_state();
void LED_set_state(bool SW_state);

int main(void)
{
    PORTB.DIR &= ~PIN4_bm;
    PORTB.DIR |= PIN4_bm;

    PORTB.DIRCLR = PIN4_bm;
    PORTB.DIRSET = PIN4_bm;

    VPORTB.DIR &= ~PIN4_bm;
    VPORTB.DIR |= PIN4_bm;

    PORTB.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
    sei();

    while (1)
    {
    }
}

#pragma region LED_functions
void LED_on()
{
    PORTB.OUTCLR = PIN4_bm;          // LEDをON
}

void LED_off()
{
    PORTB.OUTSET = PIN4_bm;         // LEDをOFF
}

void LED_set_state(bool SW_state)
{
    if (SW_state)
    {
        LED_on();
    }
    else
    {
        LED_off();
    }
}
#pragma endregion

bool SW_get_state()
{
```

```

return !(PORTB.IN & PIN5_bm);
}

ISR(PORTB_PORT_vect)
{
    uint8_t intflags = PORTB.INTFLAGS;
    PORTB.INTFLAGS = intflags;


    bool SW_state = SW_get_state();
    LED_set_state(SW_state);
}

```

2.14. デバッグ1: 中断点、段階実行、呼び出しスタック

本項は(下でリンクされる)映像と実践資料の両方としてMicrochip Studioのデバッグ能力を紹介します。主な話題は中断点(ブレイクポイント)、中断点を使う基本的なコード段階実行、呼び出しスタックウィンドウだけでなく、更にコンパイラ最適化設定調整もです。

[開始に際しての話題](#)


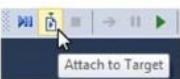


Studio 7: Debugging – 1

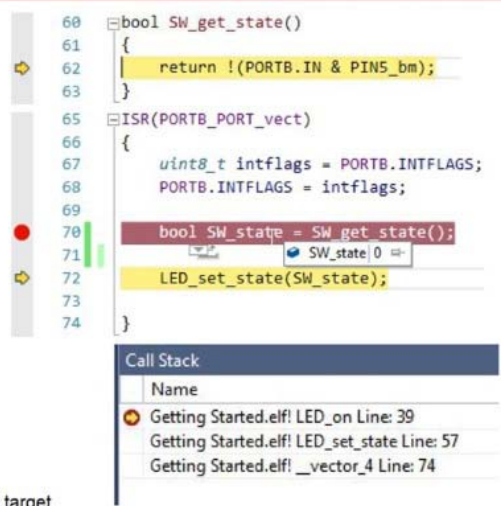
In this video:
Studio 7: Debugging 1

Context:
Debug project from Studio 7 Editor video

Features Covered:

- Basic break points
- Code stepping 
- Breakpoints window
- Call stack
- Project compiler optimization
- Attach to target 

Launch a debug session on the selected target
without RESET or uploading a new application.



```

60 bool SW_get_state()
61 {
62     return !(PORTB.IN & PIN5_bm);
63 }
64
65 ISR(PORTB_PORT_vect)
66 {
67     uint8_t intflags = PORTB.INTFLAGS;
68     PORTB.INTFLAGS = intflags;
69
70     bool SW_state = SW_get_state();
71     LED_set_state(SW_state);
72 }
73
74

```

Call Stack

Name
Getting Started.elf! LED_on Line: 39
Getting Started.elf! LED_set_state Line: 57
Getting Started.elf! _vector_4 Line: 74

映像: [Microchip Studio デバッグ-1](#)

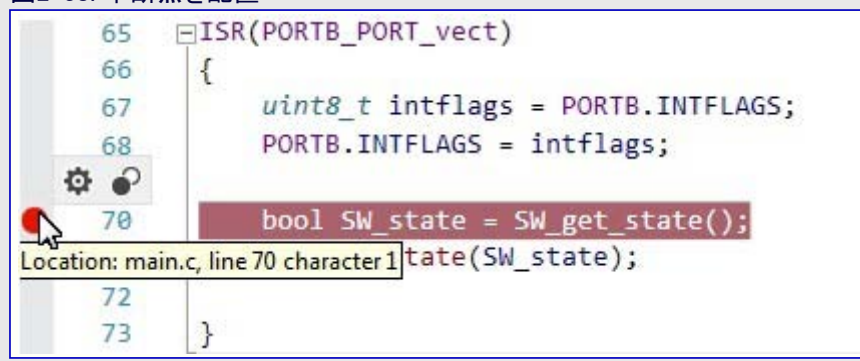
「2.12. エディタ: コードの記述と整理 (Visual Assist)」項で作成されたのと同じコードが使われます。



すべきこと: 中断点を配置してプロジェクト内の全ての中断点の一覧を調べてください。

1. 図2-38.で示されるように、スイッチ状態を得る行に中断点を設定してください。

図2-38. 中断点を配置



情報: 中断点は以下によってコードの行に配置することができます。

- エディタ ウィンドウ左端の灰色バーをクリック。
- 最上部のメニュー バーで、**Debug(デバッグ)**⇒**Toggle Breakpoint(中断点ON/OFF)**へ行く。
- キーボードでの**F9**押下による。

2. ▶でデバッグ作業を開始してください。中断点はXplained Proキット上のスイッチ(SW0)が押された時に行き当たります。中断点に行き当たった時に実行が停止され、実行矢印は中断点が配置されたコードの行が正に実行しようとしていることを示します。図2-39をご覧ください。

図2-39. 中断点に的中した時の実行停止

```

65  ISR(PORTB_PORT_vect)
66  {
67      uint8_t intflags = PORTB.INTFLAGS;
68      PORTB.INTFLAGS = intflags;
69
70      bool SW_state = SW_get_state();
71      LED_set_state(SW_state);
72
73  }

```

助言: 現在開いていないファイル内で中断点が的中した場合、Microchip Studioは一時区画でファイルを開きます。デバッグ作業で的中した中断点を含むファイルは常にフォーカスを持ちます。

3. プログラムの論理の殆どが割り込みが処理される時にだけ扱われるため、今やプログラムの論理的な流れを調べることが可能です。スイッチが押されてその後に開放された場合で割り込みが的中した時に、関数が返すスイッチの状態は何ですか?。仮定はスイッチ押下が割り込みを起動すること、スイッチが押下として設定されること、従ってLEDがONになることです。

コード段階実行はこの過程を調べるのに使うことができます。コード段階実行に使われるキー鈕は下表で説明され、最上部のメニューバーまたは**Debug(デバッグ)**メニューで見つけてください。対応する機能とキーボードショートカットは下図で略述されます。

図2-40. コード段階実行用のMicrochip Studio鈕



表2-4. Microchip Studio鈕機能(コード段階実行)

鈕	機能	キーボードショートカット
	関数呼び出し内段階実行	F11
	外側段階実行	F10
	関数外へ段階実行	Shift + F11
	カーソルまで走行	Ctrl + F10
	システムリセット発行	

すべきこと: 割り込みが的中した時にスイッチが押されてその後に開放された場合でどの状態が返されるかを見つけ出してください。割り込みを起動したスイッチ押下のため押下として設定され、故にLEDがONと言う仮定は正しいですか?。

関数呼び出し内段階実行が最初に使われ得ます。SW_get_state()関数を行くため、関数から戻った後の次の行へ移動するのに関数呼び出し外へ段階実行を使うことができます。中断点からの外側段階実行は直接この同じ点になります。LEDがONにされか否かを決定するのにLED_set_state(SW_state)関数内を更に段階実行することができますことに注意してください。けれども、今や0に設定されている、即ち、LEDがOFFされていることを知るために、単にSW_state変数上にマウスカーソルを浮かせることができます。更に段階実行することによってこれを確認してください。

図2-41. マウス浮かしを使うSW_stateの値調査

```

60  bool SW_get_state()
61  {
62      return !(PORTB.IN & PIN5_bm);
63  }
64
65  ISR(PORTB_PORT_vect)
66  {
67      uint8_t intflags = PORTB.INTFLAGS;
68      PORTB.INTFLAGS = intflags;
69
70      bool SW_state = SW_get_state();
71      LED_set_state(SW_state);
72
73  }
74

```

情報: スwitch押下による下降端によって中断点が起動されたとは言え、SW_get_state()関数を呼ぶ時にだけスイッチの状態が記録されます。この行を外側段階実行する時にスイッチが押されたままの場合に1を読むことを確認してください。

1. プログラム上の中断点を把握するためのウィンドウまたは表示部が必要とされます。敏速起動バーはMicrochip Studioユーザーインターフェースメニューの検索を実行します。これは図2-42.と図2-43.の2つの図を比べることにより、下で実演されます。敏速起動バーでの各的中がDebug(デバッグ)メニュー内の'break'関連入り口からであることに注意してください。

図2-42. 敏速起動バーでの'break'検索

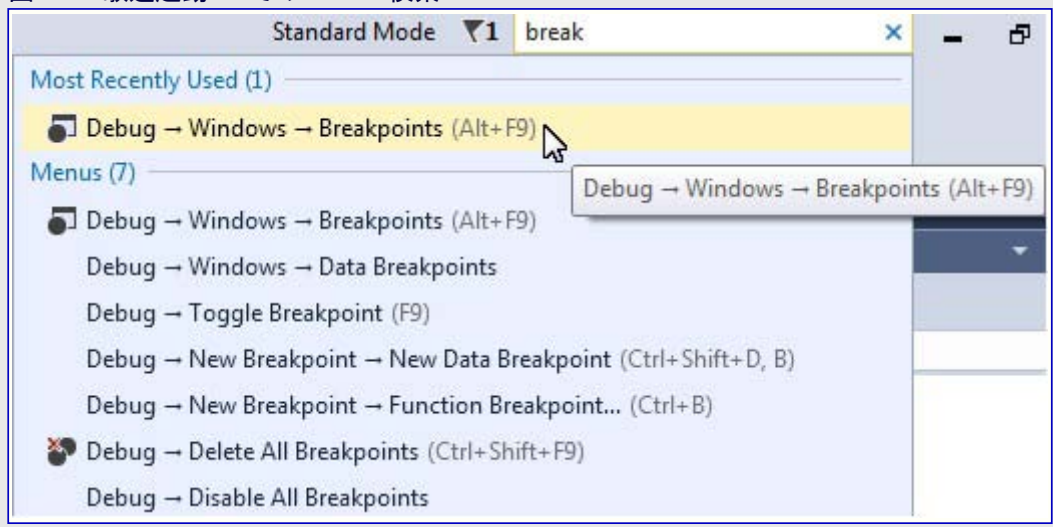
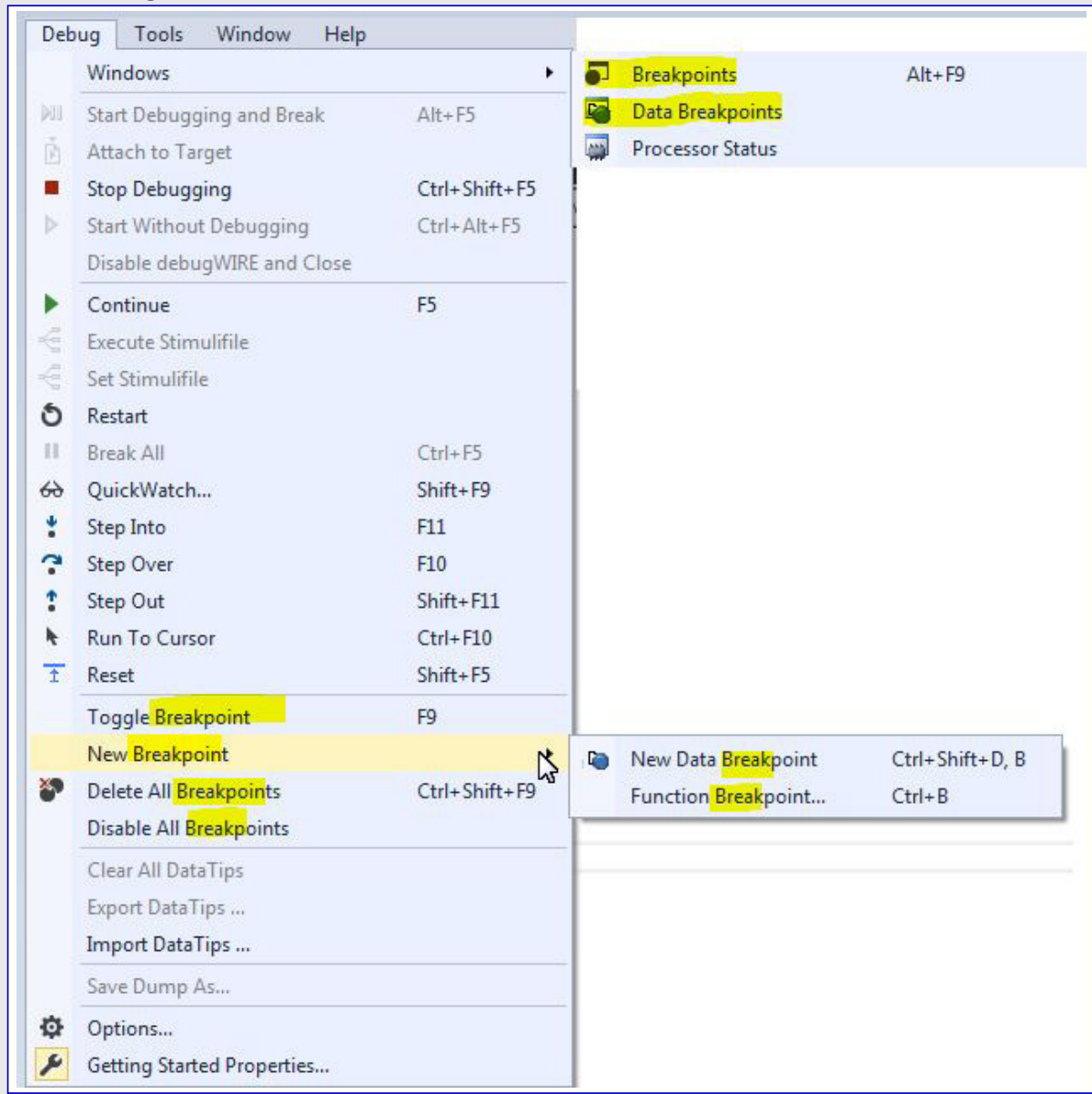



図2-43. Debugメニューでの'break'の中



先頭の結果(Debug(デバッグ))⇒Windows(ウィンドウ)⇒Breakpoints(中断点))でのクリックによってBreakpoints(中断点)ウィンドウを開いてください。図2-44.で描かれるように、中断点ウィンドウは現在の的中数と共にプロジェクト内の全ての中断点を一覧にします。

 **助言:** 中断点は一覧内の中断点傍のチェック枠でチェックを外すことによって一時的に禁止することができます。


 **助言:** 図2-45.部で実演されるように、逆アセンブリ表示部はソースコードと一緒に便利に表示することができます。

図2-44. Breakpoints(中断点)ウィンドウ

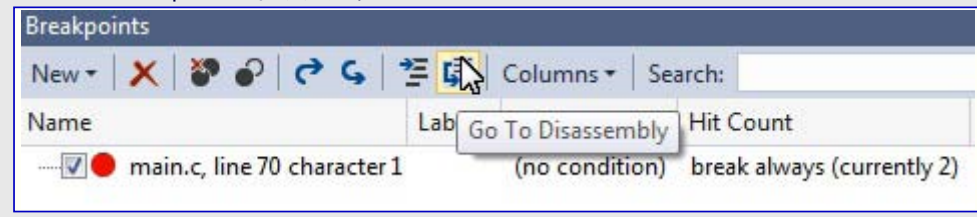
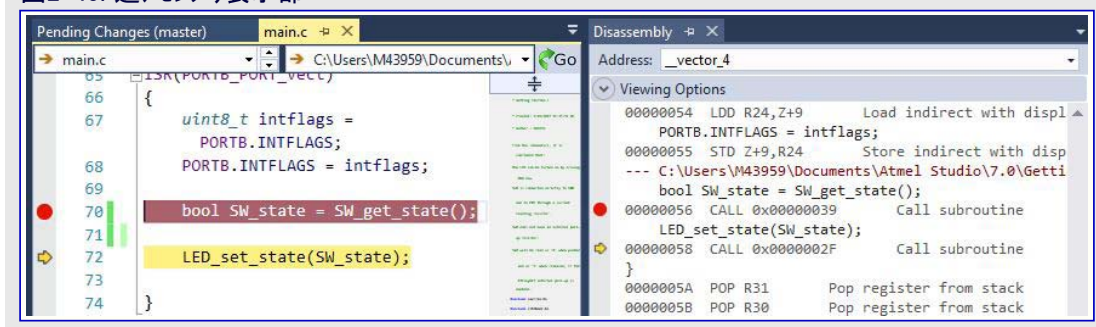



図2-45. 逆アセンブリ表示部

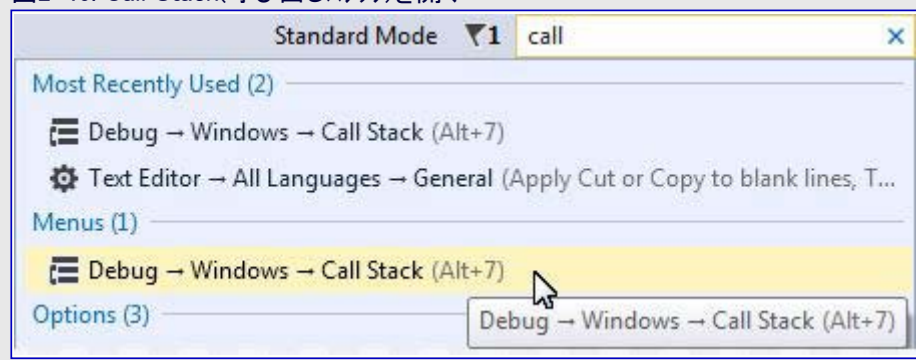


 **すべきこと:** 呼び出しスタックと最適化が禁止された時のその影響を調べてください。

1. 前項から続いて、LED_on()関数に中断点を設定し、その後にそれが的中するように中断点を起動してください。
2. 図2-46.で表されるように、敏速起動バーで'call'を入力してDebug(デバッグ)⇒Windows(ウィンドウ)⇒Call Stack(呼び出しスタック)を選ぶことによってCall Stack(呼び出しスタック)ウィンドウを開いてください。

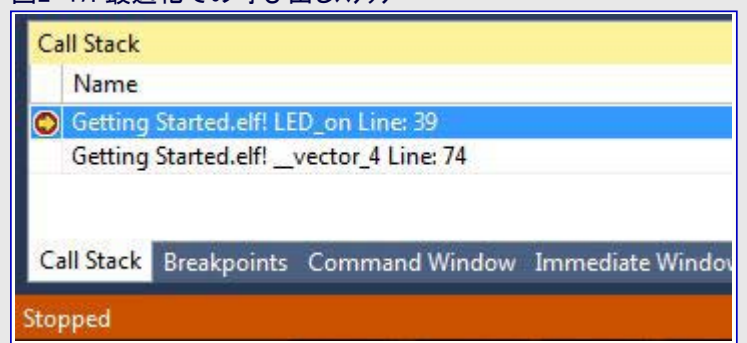
注: このウィンドウを開くにはデバッグ作業を活性(有効)にする必要があります。

図2-46. Call Stack(呼び出しスタック)を開く



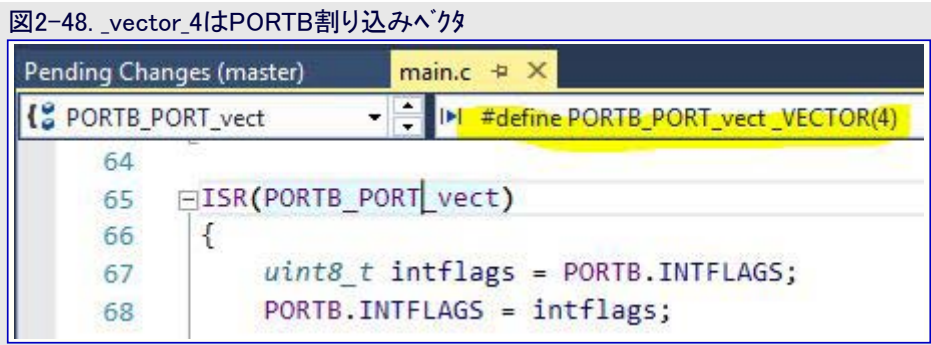
3. それはコードがどう書かれたかなので、呼び出しスタックはLED_on()の呼び出し元としてLED_set_state()を示すことが予測されます。けれども、呼び出しスタックウィンドウでは(図2-47.でのように)呼び出し元として_vector_4が一覧にされ、これはコンパイラの最適化のためです。

図2-47. 最適化での呼び出しスタック

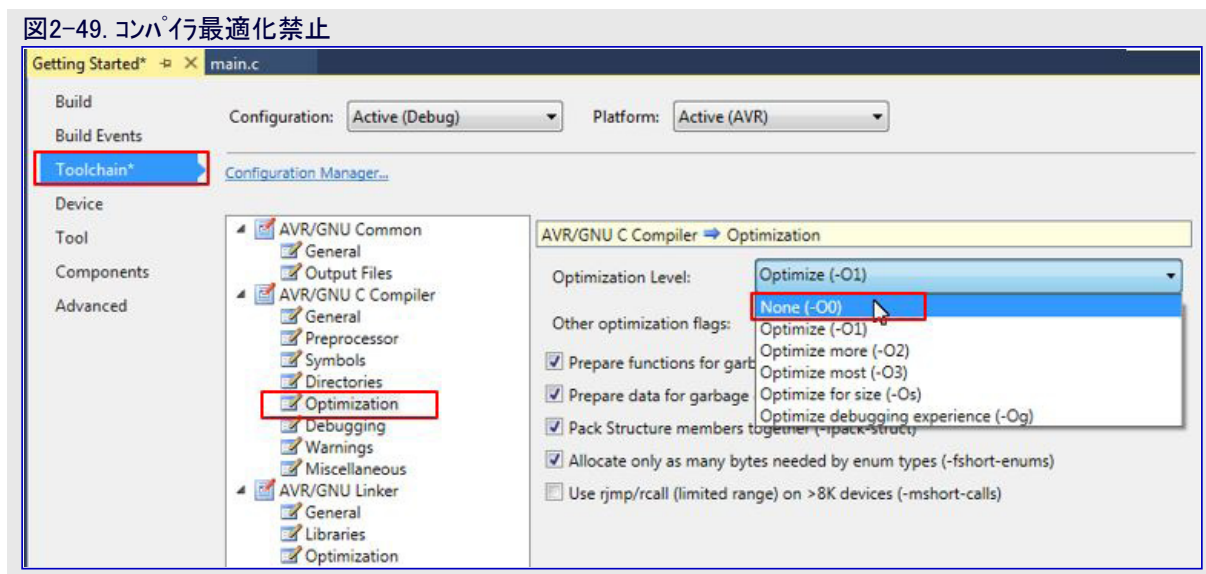


情報: 呼び出し指示はコンパイル最適化のために異なります。このコードは理解するのが比較的簡単で、例えばコンパイルで最適化され、予想されるものを微妙に変更されたとしても、何が起きているかを理解することが可能です。もっと複雑なプロジェクトではバグを探し出すのに時にはコンパイルの最適化を禁止することが役立つかもしれません。

注: 呼び出しスタックが最初に何故_vector_4から来ることを示すのを知るため、図2-48.で示されるように、PORTB_PORT_vectをクリックして定義に対する前後関係領域を覗いてみてください。



4. Stop Debugging(デバッグ停止) 鈕をクリック、またはShift+F5を押すことによってデバッグを停止してください。
5. Project(プロジェクト)⇒<プロジェクト名> properties(プロパティ)へ行くか、またはAlt+F7を押すことによってプロジェクト設定を開いてください。図2-49.のように、左メニューでToolchain(ツールチェーン)タブへ行ってください。
6. AVR/GNU C Compiler(コンパイラ)⇒Optimization(最適化)下で、引き落としメニューを使ってOptimization LevelをNone (-O0)(なし)に設定してください。

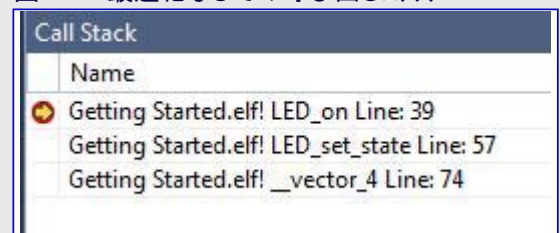


警告 コンパイラ最適化禁止はメモリ消費増加に帰着し、実行タイミングでの変化に帰着し得ます。これはデバッグ時間が重要なコードの時に考慮されることが重要で有り得ます。

7. 新しいデバッグ作業を開始してLED_on()内でコード実行を中断してください。
8. 呼び出しスタックを観察してください。図2-50.で示されるように、コードが実際にどう書かれたかを忠実にしてLED_on()の呼び出し元としてLED_set_state()を一覧にしましょう。

助言: Microchip Studioはコンパイルされたコードをできるだけソースコードに繋げようとしますが、コンパイラの最適化はこれを困難にします。コンパイラの最適化禁止は、中断点がデバッグ中に無視されたように思えたり、実行の流れがコード段階実行中に従うのが難しい場合に役立ち得ます。

図2-50. 最適化なしでの呼び出しスタック



結果: 呼び出しスタックは今や最適化許可の有りとなしの両方で調査されました。

デバッグ1に使われるコード

```

/*
LEDはスイッチが押された時にONにされ、LEDは(ピン変化割り込み経由で)ONにされます。予期せぬプロジェクト中断を避けるため、
目的対象へ取り付け実演するために書かれたMY_mistake()が注釈にされます。

回路図から以下が断定されます。:
LEDはPB4をLowに駆動することによってONにされます。
SW0はGNDに直接と電流制限抵抗を通してPB5に接続されます。
SW0は外部プルアップ抵抗を持ちません。
SW0はATtiny817の内部プルアップが許可された場合に、押下時に'0'、開放時に'1'として読みます。
*/

#include <avr/io.h>
#include <stdbool.h>
#include <avr/interrupt.h>

void LED_on();
void LED_off();
bool SW_get_state();
void LED_set_state(bool SW_state);

int main(void)
{
    PORTB.DIRSET = PIN4_bm;
    PORTB.OUTSET = PIN4_bm;
    PORTB.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
    sei();

    while (1)
    {
    }
}

#pragma region LED_functions
void LED_on()
{
    PORTB.OUTCLR = PIN4_bm;    // LEDをON
}

void LED_off()
{
    PORTB.OUTSET = PIN4_bm;    // LEDをOFF
}

void LED_set_state(bool SW_state)
{
    if (SW_state)
    {
        LED_on();
    }
    else
    {
        LED_off();
    }
}
#pragma endregion

bool SW_get_state()
{
    return !(PORTB.IN & PIN5_bm);
}

```

```

/*
void My_mistake()
{
    while(1)
    {
        asm("nop");
    }
}
*/

ISR(PORTB_PORT_vect)
{
    uint8_t intflags = PORTB.INTFLAGS;
    PORTB.INTFLAGS = intflags;
    //My_mistake();

    bool SW_state = SW_get_state();


    LED_set_state(SW_state);
}

```

2.15. デバッグ2: 条件付きと活動付きの中断点

本項は(下でリンクされる)映像と実践資料の両方としてMicrochip Studioでのもっと高度なデバッグの話題を網羅します。主な話題はコード内の変数の変更方法、条件付きと活動付きの中断点(ブレークポイント)だけでなく、更にメモリ表示部もです。

[開始に際しての話題](#)



Studio 7: Debugging – 2

In this video:

Studio 7: Debugging 2

Context”

Project from Studio 7 Editor video (polled), added logic to SW_get_state():

- SW_get_state() -> SW_get_states_logic()

Features Covered:

- **Watch:** view & modify variables
- **Conditional Breakpoints**

To do: In SW_get_states_logic() break only every 5th edge count, && if edge was rising;
- **Action Breakpoints**

To do: Log various state variables to output window.

```

uint8_t SW_get_states_logic(void)
{
    static uint8_t SW0_prv_state = 0;
    static uint8_t SW0_edge_count = 0;

    /* Read the current SW0 state */
    uint8_t SW0_cur_state = !(PORTB.IN & PINS_bm);
    if (SW0_cur_state != SW0_prv_state) /* Check for edges */
    {
        SW0_edge_count++;
    }
}

```

78 Breakpoint Settings X

Location: main.c, line: 78, Character: 1. Must match source

Conditions

Conditional Expression - Is true ((SW0_edge_count%5)==0)&&SW0_cur_state

Add condition

Output

Show output from: Debug

```

Prv state:0, Cur_state:0, Edge count:0
Prv state:0, Cur_state:0, Edge count:0
Prv state:0, Cur_state:0, Edge count:0
Prv state:0, Cur_state:0, Edge count:0
Prv state:0, Cur_state:0, Edge count:0

```

Autos Locals Watch 1 Call Stack Breakpoints Output Error List

映像: [デバッガー-2](#)

すべきこと: コードで変数の内容を調査して変更するのにMicrochip Studioを使ってください。

© 2021 Microchip Technology Inc.とその子会社

使用者の手引き

DS50002718D – 60頁

1. 使われるコード(下参照)は「2.12. エディタ: コードの記述と整理 (Visual Assist)」項で開発されたものと同じです。SW_get_state()関数が以下のコードで置き換えられています(戻り値の型が変わっていることにも注意してください)。

```
uint8_t SW_get_state(void)
{
    static uint8_t SW0_prv_state = 0;
    static uint8_t SW0_edge_count = 0;

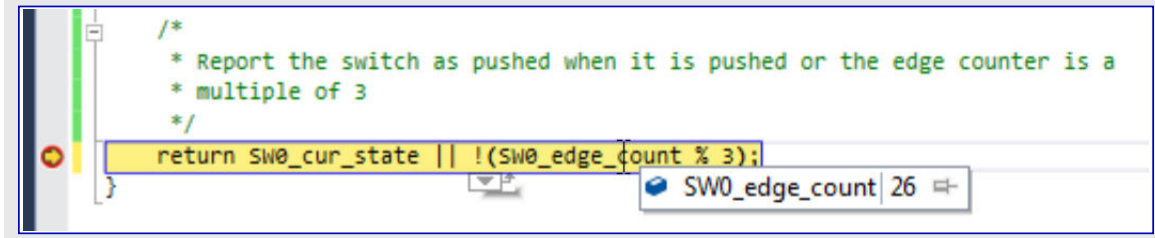
    uint8_t SW0_cur_state = !(PORTB.IN & PIN5_bm); /* 現在のSW0状態読み込み */
    if (SW0_cur_state != SW0_prv_state)           /* 端調査 */
    {
        SW0_edge_count++;
    }
    SW0_prv_state = SW0_cur_state;                /* 直前の状態を保持 */

    /*
    * スイッチが押されているか、端計数器が3の倍数の時に押下として報告
    */
    return SW0_cur_state || !(SW0_edge_count % 3);
}
```

情報: このコードはSW0押し釦がどの位押されまたは開放されたかを計数します。return文はSW0_edge_count変数が3の倍数の場合に常に釦押下として報告するように変更されてもいます。

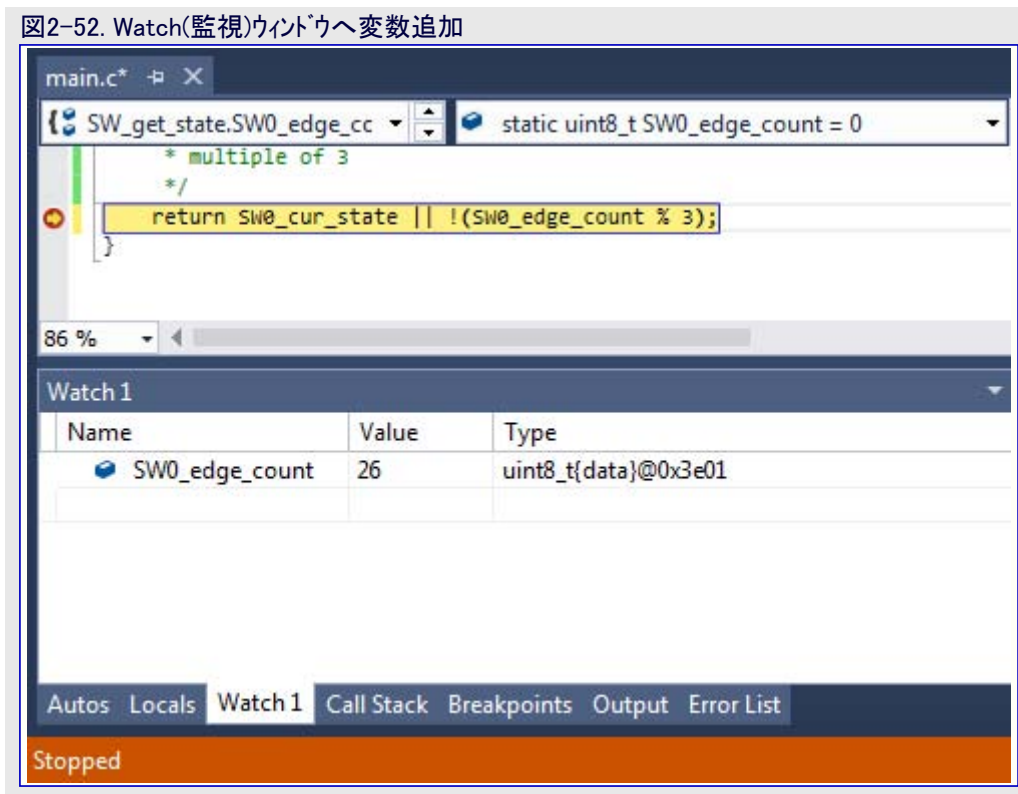
2. 全ての中断点(ブレークポイント)を禁止するためにDebug(デバッグ)⇒Disable All Breakpoints(全中断点禁止)へ行ってください。これはBreakpoints(中断点)ウィンドウで全てのチェック枠が未チェックになることによって反映されるべきです。
3. Start Debugging(デバッグ開始) ▶ 釦をクリックすることによって新しいデバッグ作業を開始してください。
4. キット上のSW0を数回押して、コードへの変更がLEDの動きにどう影響を及ぼすかを観察してください。
5. SW_get_state関数のreturn行に中断点を配置することによって実行を中断してください。
6. 図2-51.で示されるように、現在の値を観察するためにSW0_edge_count変数上にカーソルをかざしてください。

図2-51. 現在値を見るために変数上にカーソルをかざす。



情報: 実行が停止された場所での可視範囲で変数上にカーソルをかざすと、Microchip Studioはポップアップでその変数の内容を提示します。

7. 変数をデータ監視ウィンドウに追加するために、`SW0_edge_count`変数を右クリックして脈絡メニューから**Add Watch**(監視に追加)を選んでください。図2-52.でのように、変数値、データ型、メモリアドレスと共に一覧にされた`SW0_edge_count`変数を持つ**Watch**(監視)ウィンドウが現れるでしょう。



8. 下で記述される手順を使って、**Watch**(監視)ウィンドウ変数を変更してください。`SW0_edge_count`変数に値'3'を割り当ててください。図2-53.で示されるように、値は赤になることによって更新された時を反映します。
- **Watch**(監視)ウィンドウで変数値をダブル クリックしてください。
 - 望む新しい変数の値で入力してください。
 - 確認するためにEnterを押してください。

図2-53. Watch(監視)ウィンドウで新しく更新された変数値

Watch 1		
Name	Value	Type
<code>SW0_edge_count</code>	3	<code>uint8_t{data}@0x3e01</code>

情報: **Watch**(監視)ウィンドウの**Value**(値)列は監視ウィンドウで右クリックして脈絡メニューから**Hexadecimal Display**(16進数表示)を選ぶことによって16進数で表示することができます。

9. デバイスに`SW0_edge_count`の新しい値を評価させるには、全ての中断点を禁止して、▶ をクリックするか、またはF5を押すことによってデバッグ作業を続けてください。`SW0_edge_count`に行われた変更の結果としてLEDがどうONに留まるかを観察してください。

情報: 空の**Name**(名前)領域でクリックして変数名を入力することによって**Watch**(監視)ウィンドウに変数を追加することもできます。この方法は監視ウィンドウでのより良い可読性のために、変数を違うデータ型にキャスト(型変換)することさえ可能です。これは特にポインタとして関数に渡される配列を見ることが必要とされる場合に有用です。

例えば、配列が関数に渡される場合、それはポインタとして関数に渡されます。これはMicrochip Studioに対して配列の長さを知ることを不可能にします。配列の長さが既知で、監視ウィンドウで調べられることが必要なら、以下のキャストを使ってポインタを配列にキャストすることができます。

```
* (uint8_t (*) [ <n> ]) <name_of_array_pointer>
```

ここでの<n>は配列の要素数で、<name_of_array_pointer>は調べられる配列の名前です。

これは監視ウィンドウで空の**Name**(名前)領域で以下を入力することによって`SW0_edge_count`変数でこれを検査することができます。

```
* (uint8_t (*) [ 5 ]) &SW0_edge_count
```


変数へのポインタを得るためにこの場合は'&'シンボルが使われなければならないことに注意してください。

 **結果:** Microchip Studioは今やコード内の変数の内容を調べて変更するのに使われています。

2.15.1. 条件付き中断点

本項は条件付き中断点を配置するのにMicrochip Studioを使うための手引きです。

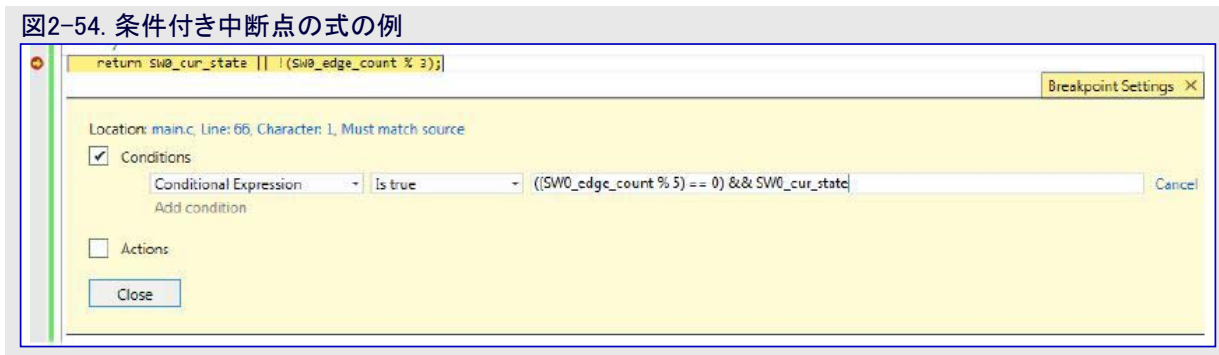
条件付き中断点は指定した条件が一致した場合にだけコード実行を停止するそれらで、或る変数が与えられた値を持つ場合に中断することが必要とされる時に有用で有り得ます。条件付き中断点は中断点が的中した回数に従ってコード実行を停止することにも使われ得ます。


 **すべきこと:** 上昇端の場合にだけ5端計数毎にデバッグ用に実行を停止するためにSW_get_state()の内側に条件付き中断点を配置してください。


1. Breakpoints(中断点)ウィンドウを使ってプロジェクトから全ての中断点を解消してください。
2. 図2-54.のように、SW_get_state()のreturn行に中断点を配置してください。
3. 中断点を右クリックして脈絡メニューからConditions...(条件...)を選んでください。
4. Conditions(条件)テキスト枠に以下を入力してください。


```
((SW0_edge_count % 5) == 0) && SW0_cur_state
```


図2-54. 条件付き中断点の式の例



5. 中断条件を承認するためにEnterを押してください。
6.  ボタンをクリックするか、またはF5を押すことによってデバッグを続けるか、新しいデバッグ作業を開始してください。
7. キット上のSW0を数回押して条件が満たされた時にコード実行がどう停止されるかを観察してください。
8. Watch(監視)ウィンドウで変数値を再検査することによって条件が合致していることを確認してください。

 **警告** 指定した中断条件に一致した時にだけコード実行が完全に停止されるとは言え、Microchip Studioは変数内容を読んで中断条件が一致するかを判断するために中断点的に的中する毎にコード実行を一時的に中断します。従って、条件付き中断点は例え実際の中断条件が決して一致しなくても、実行タイミングでの影響を持ちます。

 **助言:** 中断点が一致した回数に基づいて実行が中断することを必要なら、Hit Count(的中回数)を使ってください。

 **結果:** Microchip Studioは指定した中断条件が満足された時に実行を停止するのに使われています。

2.15.2. 活動付き中断点

本項は活動付き中断点を配置するのにMicrochip Studioを使うための手引きです。

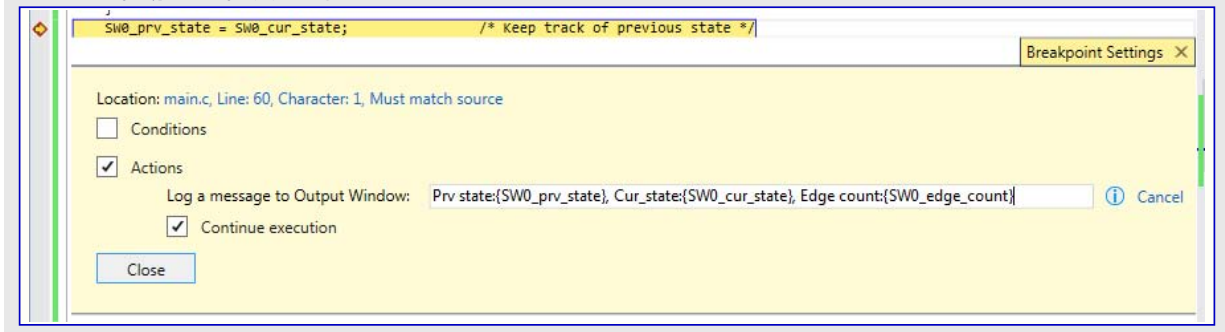
活動付き中断点はコード実行を停止して手動で必要とするデータを記録することなしに変数内容や実行の流れが記録されることを必要な場合に有用で有り得ます。

 **すべきこと:** SW0_cur_state、SW0_prv_state、SW0_edge_countを記録するために活動付き中断点を配置し、関連する変数の状態に対する出力を調べてください。

1. 進行中のデバッグ作業を停止してBreakpoints(中断点)ウィンドウから全ての中断点を解消ください。
2. 図2-55.でのように、'SW0_prv_state = SW0_cur_state;'行に中断点を配置してください。
3. 中断点を右クリックして脈絡メニューからActions...(活動...)を選んでください。
4. Log a message to Output Window(出力ウィンドウにメッセージ記録)テキスト枠に以下を入力してください。

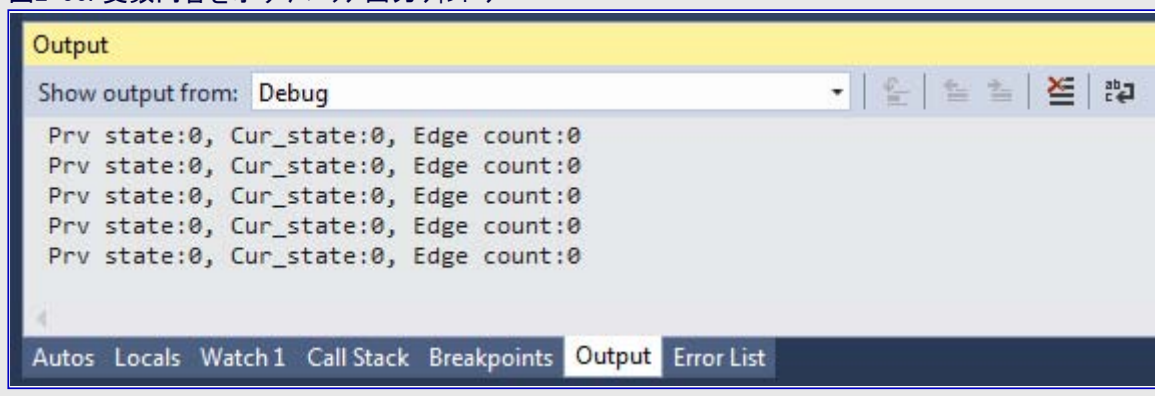
```
Prv state: {SW0_prv_state}, Cur_state: {SW0_cur_state}, Edge count: {SW0_edge_count}
```

図2-55. 活動付き中断点の例



5. 承認するためにEnterを押してください。
6. デバッグ作業を開始してください。
7. Debug(デバッグ)⇒Windows(ウィンドウ)⇒Output(出力)へ行くことによってデバッグ出力ウィンドウを開いてください。図2-56.で示されるようにそれは変数内容を一覧にするでしょう。キットでSW0が押された場合、内容が更新されます。

図2-56. 変数内容を示すデバッグ出力ウィンドウ



警告 活動付き中断点使用時、Microchip Studioは変数内容を読み出すためにコード実行を一時的に停止します。結果として、実行タイミングが影響を及ぼされます。控え目な手法はSW0端検出でだけ実行される'SW0_edge_count++'行に活動付き中断点を配置することでしょう。これはSW0が押された時にだけ一時的に停止させますが、コードの1行によって遅らされるデバッグウィンドウ出力も引き起こします。

助言: 条件が満たされた場合にだけデータを記録するために活動付きと条件付きの中断点を共に使うことができます。

結果: Microchip Studioは活動付き中断点を使って変数データを記録するのに使われています。

2.15.3. (ATtiny817 Xplained Pro用) 使用コード

条件付きと活動付きの中断点に使われるコード

```
#include <avr/io.h>
#include <avr/interrupt.h>

void LED_on();
void LED_off();
uint8_t SW_get_state();
void LED_set_state(uint8_t SW_state);

int main(void)
{
```



```
PORTB.DIRSET = PIN4_bm;
PORTB.OUTSET = PIN4_bm;
PORTB.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
sei();

while (1)
{
}

#pragma region LED_functions
void LED_on()
{
    PORTB.OUTCLR = PIN4_bm;           // LED ON
}

void LED_off()
{
    PORTB.OUTSET = PIN4_bm;          // LED OFF
}

void LED_set_state(uint8_t SW_state)
{
    if (SW_state)
    {
        LED_on();
    }
    else
    {
        LED_off();
    }
}

#pragma endregion LED_functions

uint8_t SW_get_state(void)
{
    static uint8_t SW0_prv_state = 0;
    static uint8_t SW0_edge_count = 0;

    uint8_t SW0_cur_state = !(PORTB.IN & PIN5_bm); /* 現在のSW0状態読み込み */
    if (SW0_cur_state != SW0_prv_state)           /* 端検査 */
    {
        SW0_edge_count++;
    }
    SW0_prv_state = SW0_cur_state;                /* 直前状態の把握 */

    /*
     * スイッチが押されるか、または端計数器が3の倍数の時にスイッチ押下として報告
     */
    return SW0_cur_state || !(SW0_edge_count % 3);
}

ISR(PORTB_PORT_vect)
{
    uint8_t intflags = PORTB.INTFLAGS;
    PORTB.INTFLAGS = intflags;


    uint8_t SW_state = SW_get_state();

    LED_set_state(SW_state);
}
```

2.16. デバッグ3: I/O表示部、メモリ表示部、監視

本項は(下でリンクされる)映像と実践資料の両方としてMicrochip Studioでのもっと高度なデバッグの話題を網羅します。主な話題は構成設定変更保護(CCP:Configuration Change Protected)レジスタと共に作業するためのI/O表示部、EEPROM書き込みを確認するためのメモリ表示部の使い方だけでなく、ポインタを配列としてキャスト(型変換)するためのWatch(監視)ウィンドウの使い方でもです。

開始に際しての話題



Studio 7: Debugging – 3

In this video:

Studio 7: Debugging 3

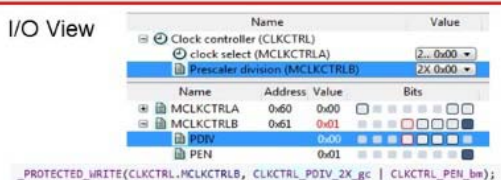
Context:

Project from Debugging 2, add function to save data to eeprom. Change clock freq to 10 MHz.

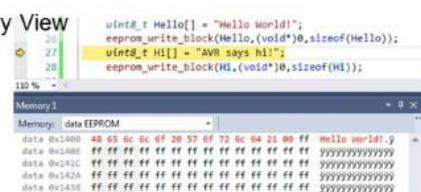
Features Covered:

- **I/O View:**
 - Configuration Change Protect (CCP) registers
- **Memory view:**
 - EEPROM Write (AVR® LibC)
- **Watch:**
 - Cast pointer to array of specified size, so can view in Watch Window


I/O View



Memory View




Watch View




映像: デバッグ-3


2.16.1. I/O表示部

I/O表示部はプロジェクトと関連するデバイスのI/Oメモリ配置の図画的表示を提供します。このデバッグ ツールはデバッグ時に実際のレジスタ内容を表示し、周辺機能構成設定の確認を許します。これは再コンパイルする必要なしにレジスタの内容を変更するのにも使うことができます。

 **すべきこと:** 以下のためにI/O表示部を使ってください。

- デバイスのメモリ配置の概要を取得
- 現在の周辺機能構成設定を調査
- 周辺機能構成設定を変更
- 構成設定変更を確認

1. 全ての中断点を取り去って新しいデバッグ作業を開始してください。
2. **Break All**(全て中断)  鈕を押すことによってコード実行を中断してください。
3. 上部メニューバーから**Debug**(デバッグ)⇒**Windows**(ウィンドウ)⇒**I/O**(入出力)へ行くことによってI/O表示部を開いてください。
4. 周辺機能の一覧を通してスクロールして**I/O Ports (PORTB)**を選んでください。図2-57.で描かれるように、**OUT**レジスタを見つけ、対応する四角が色を変えるように**Bits**(ビット)列でビット4をクリックしてください。PORTB.**OUT**レジスタでビット4をクリックすることはGPIOのPB4ピンでの出力レベルを切り替え、これはATtiny817 Xplained Pro上のLEDを制御します。

 **情報:** I/O表示部は何かのレジスタが変更された後に刷新され、検出された全ての変更が赤で強調されます。


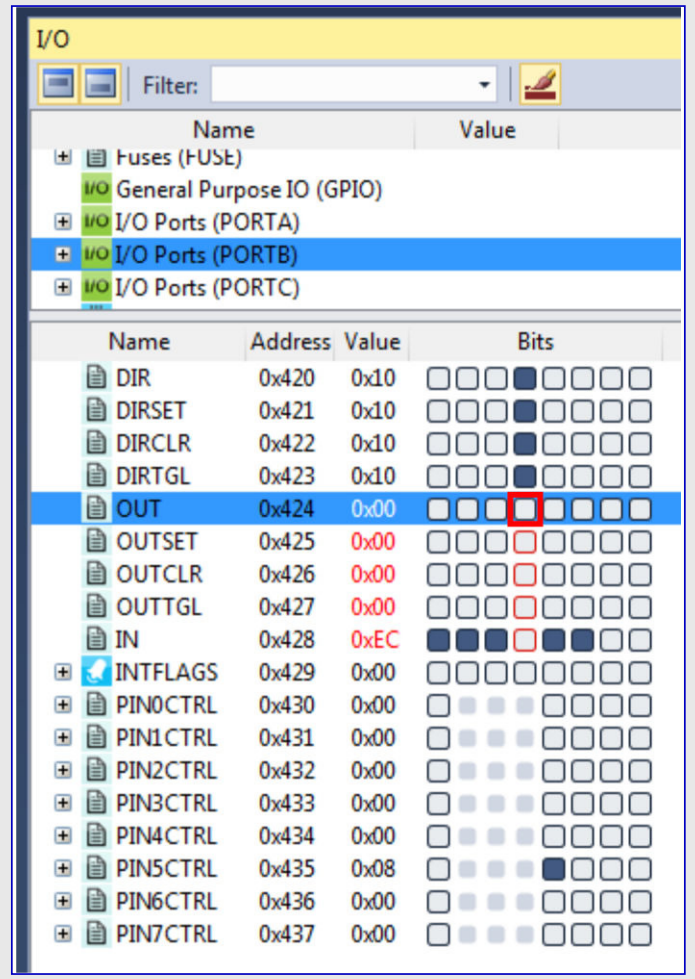


 **助言:** **Value**(値)領域をダブルクリックしてレジスタに割り当てるために望む値を入力することによって複数ビットを同時に変更することができます。

図2-57. I/O表示部を使うレジスタ内のビット値操作



5. I/Oウィンドウでクロック制御器(CLKCTRL)を展開して以下の問いに答えてください。
 - 現在選ばれているクロック元は何ですか? (**clock select**(クロック選択))
 - 構成設定した前置分周器値は何ですか? (**Prescaler division**(前置分周))
 - 主クロック前置分周器は許可ですか? (**MCLKCTRLB.PEN**)

 **結果:** クロック制御器は許可された前置分周器と分周係数6を持つ内部RC発振器から走行するATtiny817既定クロック設定で主クロックが構成設定されるべきです。

 **情報:** 既定クロック構成設定はデバイスが1.8~5.5Vの支援される動作電圧範囲全体に渡ってコードを実行することを保証します。Xplained ProキットはATtiny817に3.3Vで給電します。デバイスのデータシートの「全般動作定格」に従って、デバイスは3.3V供給、10MHzで安全に走行することができます。

6. コードは今や10MHzでATtiny817を走らせるように変更されています。下のようmain()の始めを変更してください。

```
int main(void)
{
    /*
     * 主クロック分周計数を2に設定して主クロック前置分周器許可を保ってください。
     */
    CLKCTRL.MCLKCTRLB = CLKCTRL_PDIV_2X_gc | CLKCTRL_PEN_bm;
```


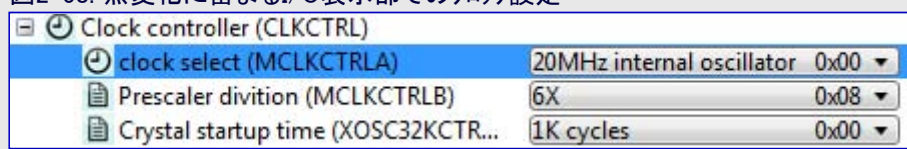
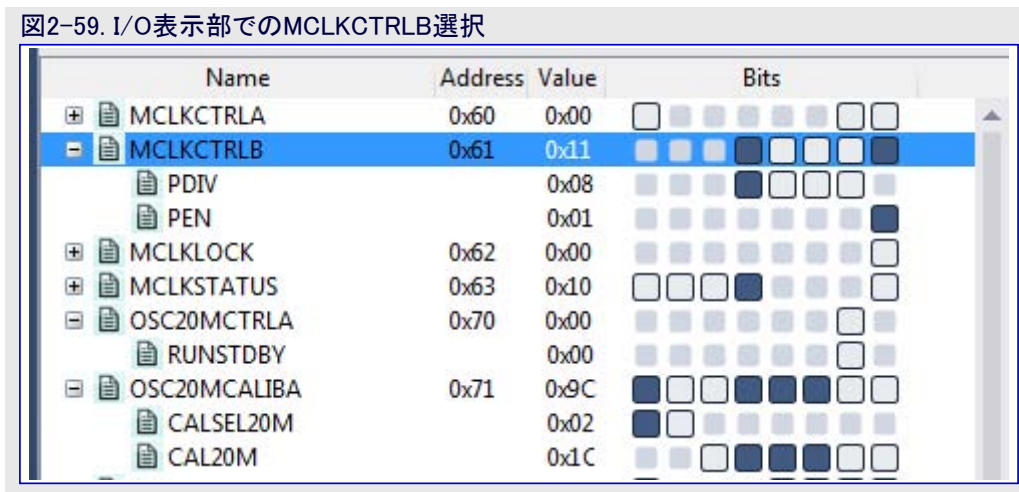
7. プロジェクトを再コンパイルしてデバイスに書き込むために新しいデバッグ作業を開始してください。
8.  をクリックすることによってコード実行を停止してください。図2-58.で描かれるI/O表示部でクロック設定を調べてください。

図2-58. 無変化に留まるI/O表示部でのクロック設定



結果: 問題があります!。前置分周器が無変化のままです。

9. 図2-59.で示されるように、I/O表示部で**MCLKCTRLB**レジスタを選択してください。



10. ウェブに基づくレジスタ説明を提示するためにキーボードで**F1**を押してください。

情報: ウェブに基づくレジスタ説明を使うにはインターネット アクセスが必要とされます。インターネット アクセスが利用不能の場合、ATtiny817 データシートのオフライン版を参照してください。

11. **MCLKCTRLB**レジスタに何かアクセス制限が適用されるかを探してください。

結果: このレジスタは構成設定変更保護(CCP: Configuration Change Protection)機構によって保護されます。重要なレジスタは予期せぬ変更を防ぐために構成設定変更保護されます。これらのレジスタはデータシートで記述されるように、正しい解錠手順に従った場合にだけ変更することができます。

12. たった今追加したコードの行を以下で置き換えてください。

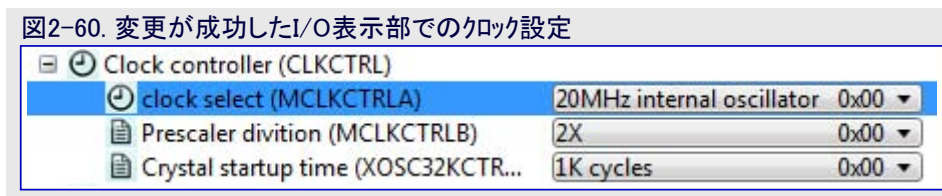
```
_PROTECTED_WRITE(CLKCTRL.MCLKCTRLB, CLKCTRL_PDIV_2X_gc | CLKCTRL_PEN_bm);
```

情報: `_PROTECTED_WRITE()`は保護されたレジスタを解錠するためのタイミング要件が合致することを保証するアセンブリ マクロです。保護されたレジスタを変更する時にこのマクロを使うことが推奨されます。

助言: マクロの実装へ誘導するにはコードのマクロ名を右クリックして**Goto Implementation**(実装へ行く)を選んでください。これはコードのマクロ名にカーソルを置いてキーボードで**Alt+G**を押すことによっても可能です。変数宣言と関数実装に対しても同じ手順を使うことができます。

13. 変更と共にデバイスを書き込むために以前のデバッグ作業を停止して新しい(デバッグ)作業を開始してください。

14. 図2-60.で示されるように、コード実行を中断して前置分周器が今や成功裏に2分周(2X)に設定されていることを確認するのにI/O表示部を使ってください。



助言: **Processor Status**(プロセッサ状態)ウィンドウはAVRコアに対するレジスタ表示ツールです。このツールは上部メニューバーから**Debug**(デバッグ)⇒**Windows**(ウィンドウ)⇒**Processor Status**(プロセッサ状態)へ行くことによって開くことができます。このウィンドウは内部AVRコアレジスタの状態の詳細な表示を提供します。この表示部は全体割り込みが許可されているかを調べる(ステータスレジスタの**IF**ビットを見る)のに使うことができます。

結果: I/O表示部の能力はプロジェクトのバグを見つけて修正するのに使われています。

2.16.2. メリ表示部



すべきこと: ATtiny817のEEPROM先頭に2つの文字列を書き、EEPROM内容を確認するためにメリ表示部を使ってください。

1. '#include <avr/io.h>' 行の後に '#include <avr/eeprom.h>' を追加してください。
2. main()の'while (1)' 繰り返しの前に以下のコードを追加してください。

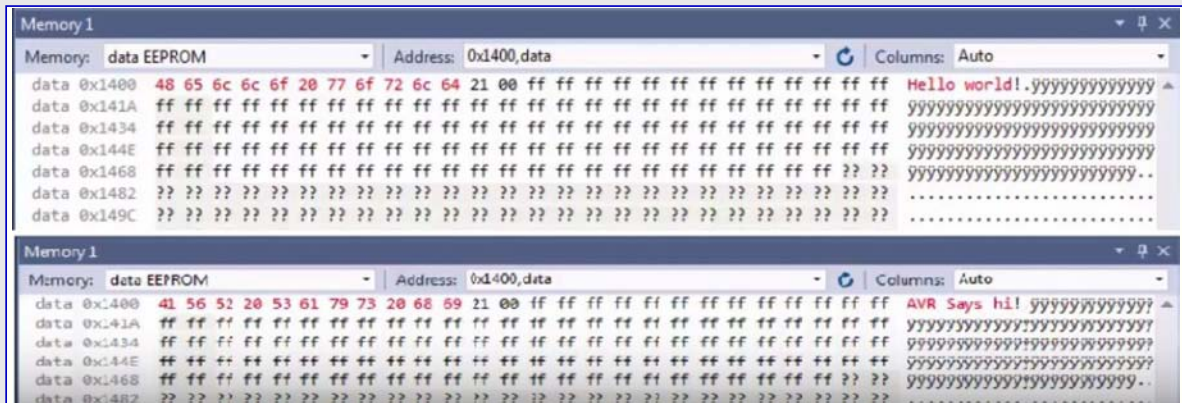
```
uint8_t hello[] = "Hello World";
eeprom_write_block(hello, (void *)0, sizeof(hello));
uint8_t hi[] = "AVR says hi";
eeprom_write_block(hi, (void *)0, sizeof(hi));
```

3. 図2-61.のように最初のeeprom_write_block()呼び出しの隣に中断点を配置してください。



4. 更新したコードでデバイスを書くために新しいデバッグ作業を開始してください。
5. 中断点に的中した後、上部メニューバーからDebug(デバッグ)⇒Windows(ウィンドウ)⇒Memory(メリ)⇒Memory 1(メリ1)へ行くことによってメリ ウィンドウを開いてください。
6. eeprom_write_block()呼び出し上を段階実行するためにキーボードでF10を押し、EEPROM書き込みを確認してください。
7. メリ表示部を使って(次の)書き込みを確認する前に次のEEPROM書き込みを実行することをATtiny817に許してください。表示部は各々は各間隔で図2-62.のように現れるべきです。

図2-62. EEPROM書き込み後のメリ表示部更新



助言: メリ表示部ツールはプログラム メリを含む他のAVRメモリ区部の内容を調べるのにも使うことができます。これはブートローダをデバッグする時に有用で有り得ます。



結果: EEPROMの内容は各eeprom_write_block()呼び出し後に更新されます。更新された内容は赤で強調され、EEPROM内容のASCII解釈が書かれた文字列と一致します。従って、EEPROM書き込み後のその内容がメリ表示部を使って確認されました。

2.16.3. 監視ウィンドウ

これは「2.15. デバッグ2: 条件付きと活動付きの中断点」項をもっと詳細に網羅しますが、Watch(監視)ウィンドウでポインタを配列としてキャスト(型変換)する方法の要旨がここで繰り返されます。



情報: 空のName(名前)領域でクリックして変数名を入力することによってWatch(監視)ウィンドウに変数を追加することもできます。この方法は監視ウィンドウでのより良い可読性のために、変数を違うデータ型にキャスト(型変換)することさえ可能です。これは特にポインタとして関数に渡される配列を見ることが必要とされる場合に有用です。

例えば、配列が関数に渡される場合、それはポインタとして関数に渡されます。これはMicrochip Studioに対して配列の長さを知ることを不可能にします。配列の長さが既知で、監視ウィンドウで調べられることが必要なら、以下のキャストを使ってポインタを配列にキャストすることができます。

```
*(uint8_t (*) [<n>]) <name_of_array_pointer>
```

ここでの<n>は配列の要素数で、<name_of_array_pointer>は調べられる配列の名前です。

これは監視ウィンドウで空のName(名前)領域で以下を入力することによってSW0_edge_count変数でこれを検査することができます。

```
*(uint8_t (*) [5])&SW0_edge_count
```

変数へのポインタを得るためにこの場合は‘&’シンボルが使われなければならないことに注意してください。



結果: Microchip Studioは今やコード内の変数の内容を調べて変更するのに使われています。

デバッグ3に使うコード

```
#include <avr/io.h>
#include <avr/eeprom.h>

void LED_on(void);
void LED_off(void);
void LED_set_state(uint8_t state);
uint8_t SW_get_state(void);
uint8_t SW_get_state_logic(void);

int main(void)
{
    PORTB.DIRSET = PIN4_bm; /* LEDピンを出力として構成設定 */
    PORTB.PIN5CTRL = PORT_PULLUPEN_bm; /* SW0ピンに対してプルアップ許可 */

    _PROTECTED_WRITE(CLKCTRL.MCLKCTRLB, CLKCTRL_PDIV_2X_gc | CLKCTRL_PEN_bm);

    uint8_t Hello[] = "Hello World!";
    save(Hello, sizeof(Hello));
    uint8_t Hi[] = "AVR says hi!";
    save(Hi, sizeof(Hi));

    while(1)
    {
        uint8_t SW0_state = SW_get_state_logic(); /* スイッチ状態を読み込み */
        LED_set_state(SW0_state); /* LED状態を設定 */
    }
}

void save(const uint8_t* to_save, uint8_t size)
{
    eeprom_write_block(to_save, (void*)0, size);
}

uint8_t SW_get_state()
{
    return !(PORTB.IN & PIN5_bm);
}
```

```
uint8_t SW_get_state_logic(void)
{
    static uint8_t SW0_prv_state = 0;
    static uint8_t SW0_edge_count = 0;

    uint8_t SW0_cur_state = !(PORTB.IN & PIN5_bm); /* 現在のSW0状態を読み込み */
    if (SW0_cur_state != SW0_prv_state)           /* 状態調査 */
    {
        SW0_edge_count++;
    }
    SW0_prv_state = SW0_cur_state;                /* 直前の状態を把握 */

    /*
     * スイッチが押されているか、または端計数器が3の倍数の時にスイッチ押下として報告
     */
    return SW0_cur_state || !(SW0_edge_count % 3);
}

void LED_off(void)
{
    PORTB.OUTSET = PIN4_bm;                       /* LEDをOFF */
}

void LED_on(void)
{
    PORTB.OUTCLR = PIN4_bm;                       /* LEDをON */
}

void LED_set_state(uint8_t state)
{
    if (state)
    {
        LED_on();
    }
    else
    {
        LED_off();
    }
}
```

3. プロジェクト管理

3.1. 序説

Microchip StudioはAVR/Arm基盤に対して応用を書いてデバッグするための統合開発環境(IDE: Integrated Development Environment)です。現在コード書き環境として、完全なIDE環境で内包されるAVRアセンブラと何れかの外部AVRGCC/ARMGCCコンパイラを支援します。

IDEとしてのMicrochip Studioの使用は様々な利点を与えます。

1. より速い誤り追跡を許す、同じ応用ウィンドウでの編集とデバッグ
2. 例えその間にコードが編集されたとしても、作業間で中断点(ブレークポイント)が保存されて復元されます。
3. プロジェクト外項目管理は便利で可搬にされます。

3.1.1. 解決策箱 (Solution Container)

AVR Studio 5で”解決策(solution)”の概念が導入されます。解決策は様々なプロジェクトを含むかもしれない入れ物です。プロジェクトは解決策の外側に存在することができません。(。cprojまたは.asmprojの拡張子の)プロジェクト ファイルを開こうとする場合、解決策が作成されます。これは例えば同じ解決策内にブートローダ プロジェクトと様々な応用プロジェクトを保持することを許します。実際には解決策が.atslnファイルとして格納されます。一般に、解決策に追加されるプロジェクトは.atslnファイルが属すフォルダの内側で独立したフォルダに置かれます。

3.1.2. プロジェクトの保存と開く

全てのプロジェクトはGCCプロジェクトに対して.cpproj拡張子、8ビット アセンブラプロジェクトに対して.asmproj拡張子を持つ選んだ名前の下に保存されます。使用者はfile(ファイル)メニューの最近使ったプロジェクト一覧、またはProject(プロジェクト)メニューのOpen project(プロジェクトを開く)下のどちらかからプロジェクトを再開することができます。

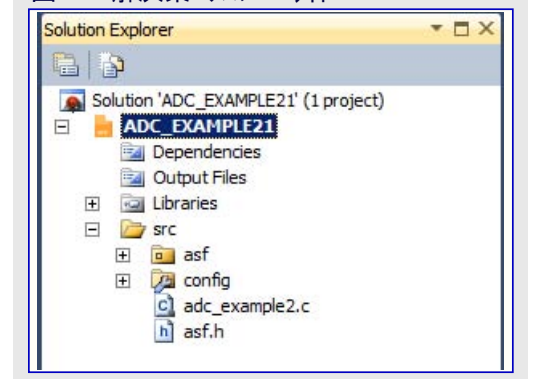
3.1.3. プロジェクト出力表示

プロジェクトの構築、アセンブル、またはコンパイル後、操作結果が構築出力ウィンドウで示されます。何れかの異常が起きた場合、使用者はメッセージをダブル クリックすることができ、それはソース ウィンドウで対応する行上に印を置きます。

3.1.4. 解決策エクスプローラ (Solution Explorer)

解決策エクスプローラは解決策またはプロジェクトに於いて項目を見て項目管理作業の実行を許します。解決策またはプロジェクトの環境の外側でファイルに取り組むためにMicrochip Studioのエディタの使用も許します。既定でそれはMicrochip Studio GUIの右側に現れます。

図3-1. 解決策エクスプローラ枠



3.1.5. ツールバー アイコン

樹状表示で選ばれた項目に特有の釘が解決策エクスプローラで現れます。

- 樹状表示で選んだ項目に対して適切なプロパティユーザーインターフェースを表示します。
- プロジェクトと隠されたそれらに於いて、除かれたそれらを含む全てのプロジェクト項目を見せます。

3.1.6. 階層的な表示

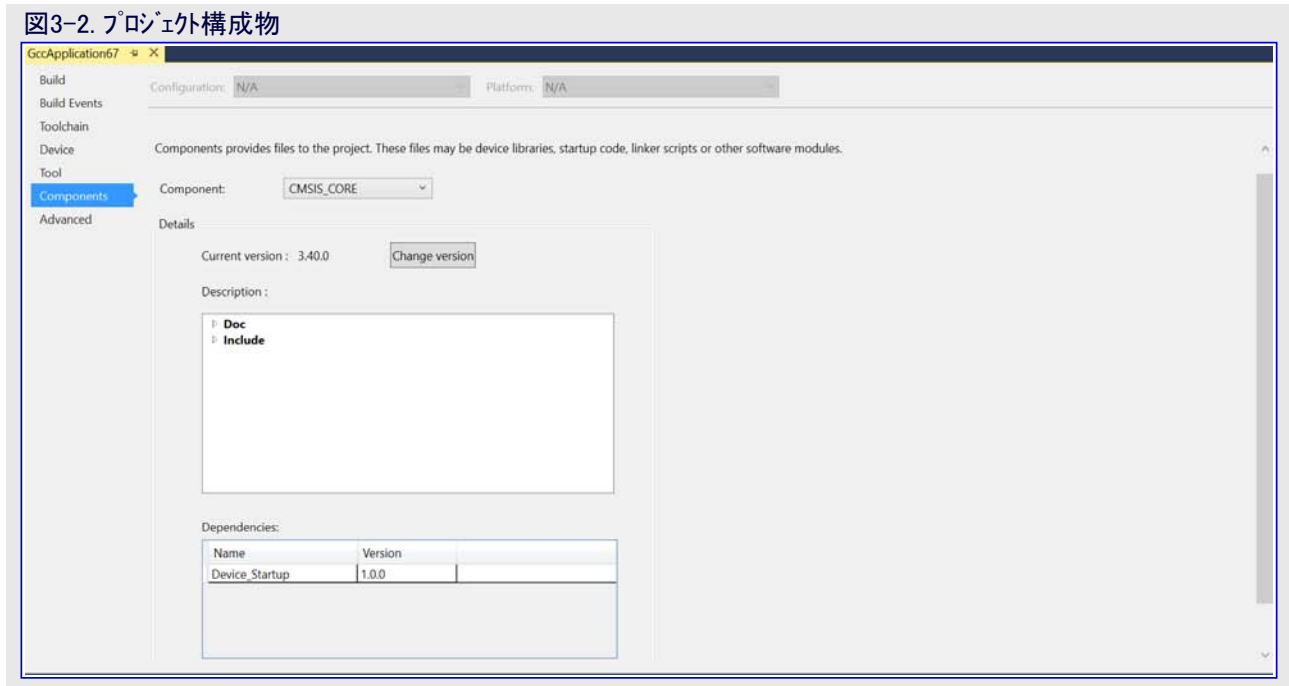
単一解決策とそれの全てのプロジェクトが階層表示で現れます。これは同時に様々なプロジェクトに取り組み、それと同時に全てのプロジェクト外と項目の経緯を保つことを許します。改訂制御下であるプロジェクト項目の更新状態を合図するために、(AnkhSVNなどのような)殆どのソース制御システム拡張子は項目アイコンに置き換えアイコンも追加します。

3.1.7. 項目管理命令

解決策エクスプローラは各々のプロジェクトや解決策の項目に対する様々な管理命令を支援します。その特定項目に対して利用可能な命令を持つメニューを得るにはどれかの項目上を右クリックしてください。

3.1.8. プロジェクト構成物

プロジェクトはデバイス特有構成物の組を含みます。これは始動コード、リンクスクリプト、他の支援ライブラリを含みます。構成物はコードの小さな断片または何れかのプロジェクトに含まれる他の支援ファイルです。



プロジェクトに含まれる構成物はComponent(構成物)引き落としメニューで一覧にされます。引き落としメニューからの構成物選択は構成物の版番号、構成物が寄与するファイル、構成物が持つ依存性を示します。

構成物の版はChange version(版変更)鈕をクリックすることによって変更することができます。

3.1.8.1. 版変更

構成物プロジェクトに追加される時に版管理されます。使う版を変更するにはこのダイアログを使ってください。

構成物の版を選ぶ時に2つの任意選択があります。

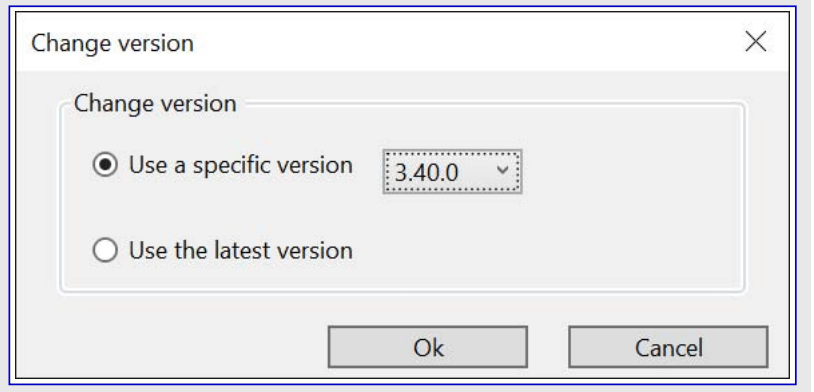
Use a specific version(特定版使用) : プロジェクトを構成物の特定版に固定化します。

Use the latest version(最新版使用) : 利用可能な構成物の最も最近の版を選びます。

構成物はMicrochip Studio内のデバイス一括の一部です。これらのデバイス一括はDevice Pack Manager(デバイス一括管理部)を用いて管理されます。

関連リンク [6.1. デバイス一括管理部](#)

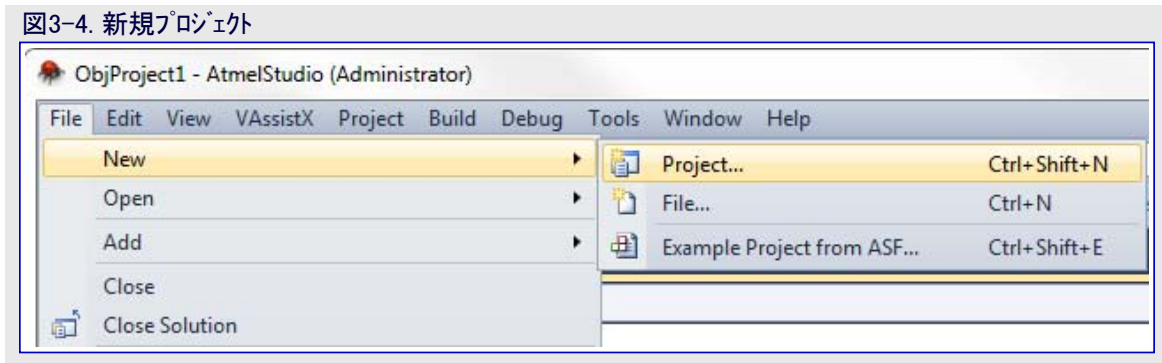
図3-3. 版変更



3.2. GCCプロジェクト

3.2.1. 新規プロジェクト ウィザード

メニューからFile(ファイル)⇒New(新規)を選んでください。下のダイアログが現れます。始動ウィザードは新しいプロジェクトを開始するための任意選択も持ちます。



プロジェクト形式

現在は様々なプロジェクト形式がProject Type(プロジェクト形式)枠で利用可能です。AVR基板の使い方を通して案内するための - AVR基板例、あなたがAVRツールで自身の製品を作成する場合の - 使用者基板プロジェクト、GNUコンパイラでの基板と無関係のプロジェクトの - 一般AVR GCCプロジェクト。それは支援されるソースコード形式のどれをも含むかもしれない、AVRアセンブラプロジェクトと一般的なAVR解決策を作成することも可能です。

助言: プロジェクトは支援されるオブジェクトファイルを読み込むことによって作成することもできます。そのようなプロジェクトの作成を望む場合、File(ファイル)⇒Open file(ファイルを開く)メニューを使うべきです。

プロジェクト名と初期ファイル

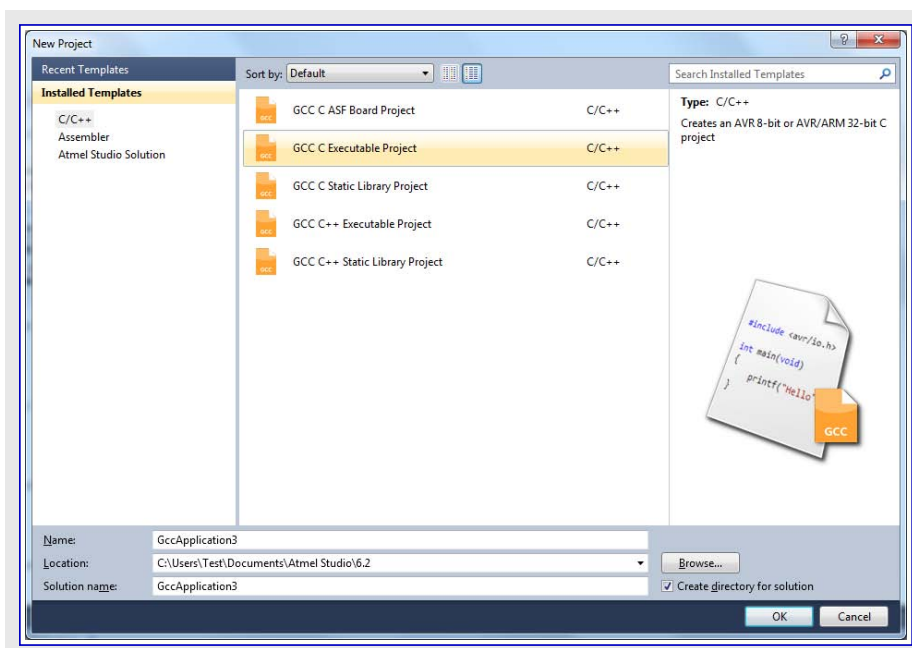
プロジェクト名を入力してください。自動的に生成されるプロジェクト主ファイルは既定によって同じ名前(ASMまたはC)で名付けられます。望むなら、この名前を変更することができます。プロジェクト名を持つ新しいフォルダを作成するために枠をチェックすることが可能です。この枠は既定によってチェックされません。

Solution(解決策)引き落としメニューで新しい解決策を作成するか、または既存コードを再使用するかを選ぶことができます。Solution Name(解決策名)領域に解決策名を入力してください。

プロジェクトの名前と形式に満足なら、OKを押してデバッグ基盤選択段階へ進んでください。今の所で基盤未定義のままにすることもできますが、その後にデバッグ作業の始めでデバッグ基盤とデバイスを選ばなくてはなりません。「3.4. アセンブラプロジェクト」と「4.17. オブジェクトファイル形式」もご覧ください。

3.2.2. AVRデバイス用新規GCCプロジェクト開始

1. Project(プロジェクト)メニューからNew Project(新しいプロジェクト)を選ぶことによって新しいプロジェクトを作成してください。これはProject Wizard(プロジェクト ウィザード)を開きます。

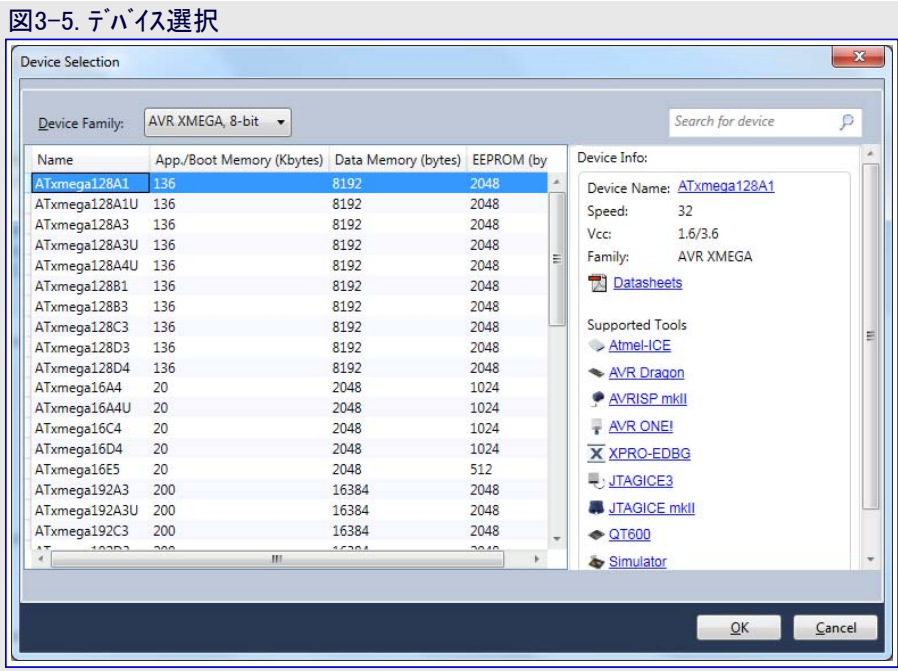


- 雛形としてC/C++⇒GCC C Executable Project(実行可能プロジェクト)を選び、その後にプロジェクト名を指定し、場所を選び、プロジェクトに対する解決策名を書いてください。既定によってプロジェクトと同じ名前を持つファイルが作成され、プロジェクトに追加されます。それは空のmain()関数を含みます。初期ファイルの名前の変更を望むなら、後で主ファイル名を単に編集してください。この設定で満足された時にOKを押してください。
- 雛形としてC/C++⇒GCC C Static Library Project(静的ライブラリプロジェクト)を選び、その後にプロジェクト名を指定し、場所を選び、プロジェクトに対する解決策名を書いてください。これはコードを再使用するための良い方法である静的ライブラリ(LIB)プロジェクトを作成します。

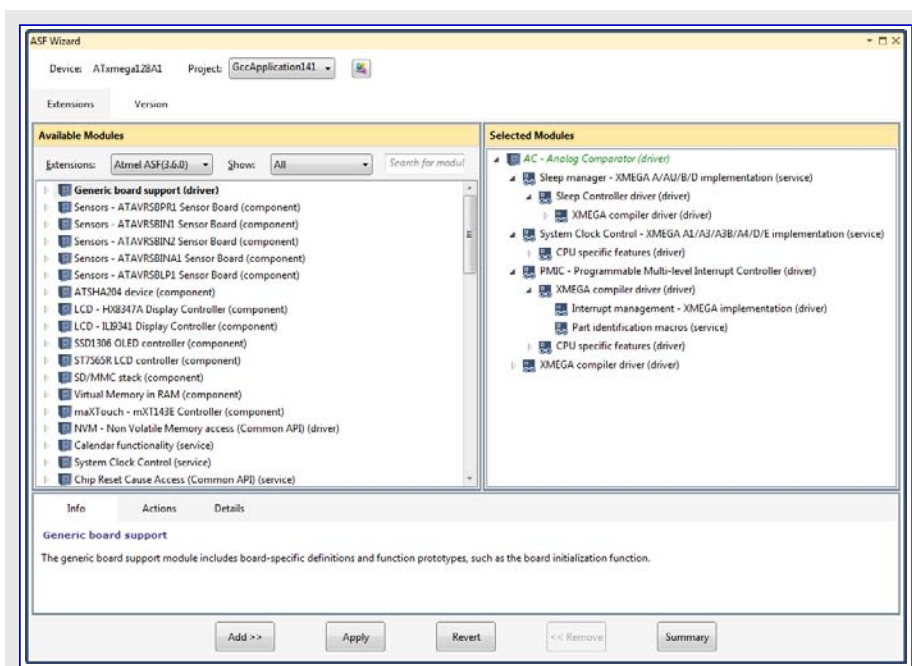


助言: 静的ライブラリプロジェクトについてもっと学ぶには「3.2.6. 新規GCC静的ライブラリプロジェクト開始」項をご覧ください。

- デバイス選択表が現れます。プロジェクトに対して適切な目的対象基盤を選んでください。始めるためにATxmega128A1デバイスを選ぶことができます。



- プロジェクト樹形が構成設定されます。段階2で作成された初期ファイルがプロジェクト節点に追加されていることに気付いてください。また、初期ファイルがエディタで開かれます。
- 応用の開発と検証を楽にするため、Project(プロジェクト)⇒ASF Wizard(ASFウィザード)から呼び出されるドライバ選択ウィザードを使うこともできます。



ASFウィザードに於いて現在の構築構造と基板に対するプロジェクトで使用を望むドライバ、構成物、サービスを選ぶことができます。

7. そこで、開いたエディタ ウィンドウ内に以下のコードを書いてください。

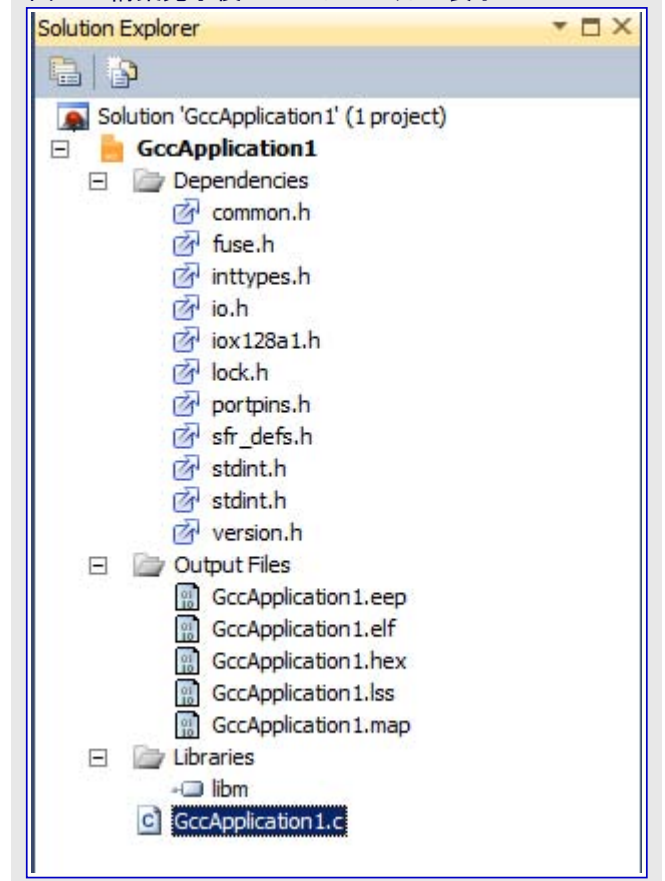
```
#define MAXINT 200000

int main(void)
{
    unsigned int t=1000, k=0, l=5, pn=2;
    unsigned int primes[t];
    primes[0]=2;
    primes[1]=3;

    while (pn < t || primes[pn] < MAXINT)
    {
        for ( k = 0; k <= pn; k++)
        {
            if (1 % primes[k] == 0)
            {
                goto otog;
            }
            else
            {
                if (k == pn)
                    primes[pn++]=1;
            }
        }
    }
    otog:
        l += 2;
}
return 0;
}
```

8. プロジェクトを構築してください。

図3-6. 構築完了後のGCCプロジェクトの表示



Dependencies (依存性)

全てのインクルードされたファイルがここで一覧にされます。エディタでそれを開くにはどれかのファイル上をダブル クリックしてください。

Output Files (出力ファイル)

全ての出力ファイルがこの項目下で表示されます。

Libraries (ライブラリ)

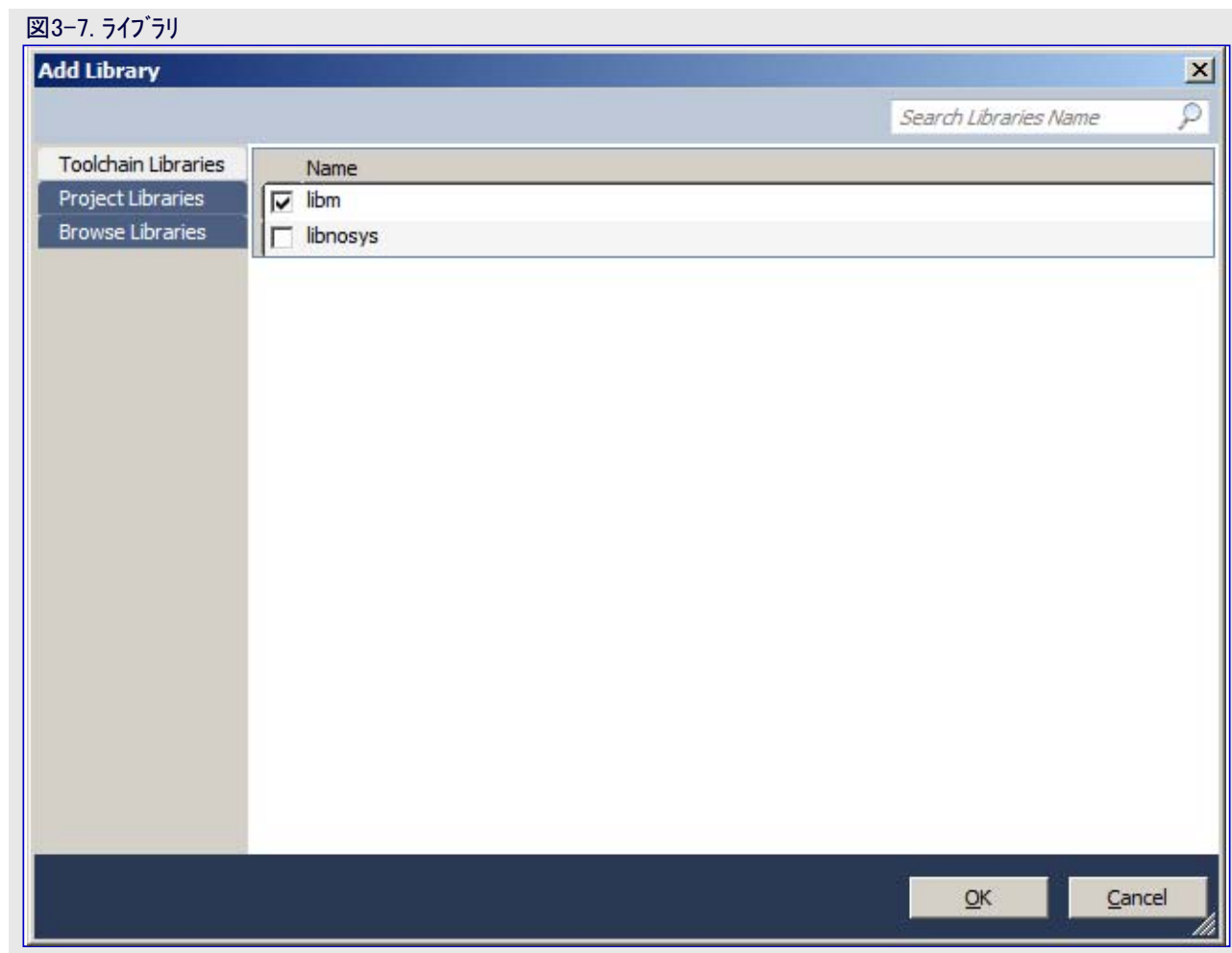
全ての静的ライブラリ、ツールチェーン ライブラリ、他のライブラリ ファイルがこの項目下で表示されます。



助言: ライブラリ任意選択についてもっと知るには「3.2.3. ライブラリ任意選択」項をご覧ください。

3.2.3. ライブラリ任意選択

全ての静的ライブラリ、ツールチェーン ライブラリ、他のライブラリ ファイルがこの項目下で表示されます。



3.2.3.1. Toolchain Libraries (ツールチェーン ライブラリ)

ツールチェーン ライブラリはここで一覧にされます。

ライブラリ一覧を形成するためにツールチェーンによって提供されたライブラリ検索パスが列挙されます。

3.2.3.2. Project Libraries (プロジェクト ライブラリ)

現在の解決策で利用可能なプロジェクトが列挙され、静的ライブラリがここで一覧にされます。

3.2.3.3. Browse Libraries (ライブラリ閲覧)

他のライブラリを閲覧することができます。

3.2.3.4. プロジェクト ライブラリの追加方法



助言: 現在の解決策内に静的ライブラリ プロジェクトを持つことを確実にしてください。

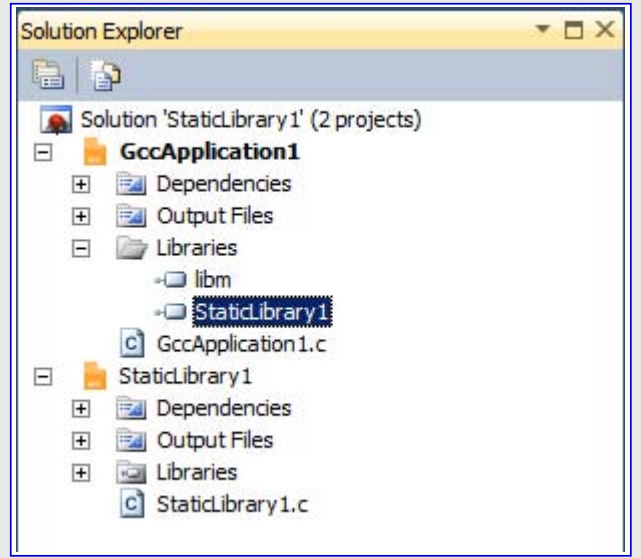
“Add Library(ライブラリ追加)”ウィザードを呼び出すためにプロジェクトまたはプロジェクト内のライブラリ節点で右クリックしてください。

Project Libraries(プロジェクト ライブラリ)タブを選んでください。ここは一覧にされた現在の解決策内の全ての静的ライブラリを見ます。

追加したい静的ライブラリを選んでください。

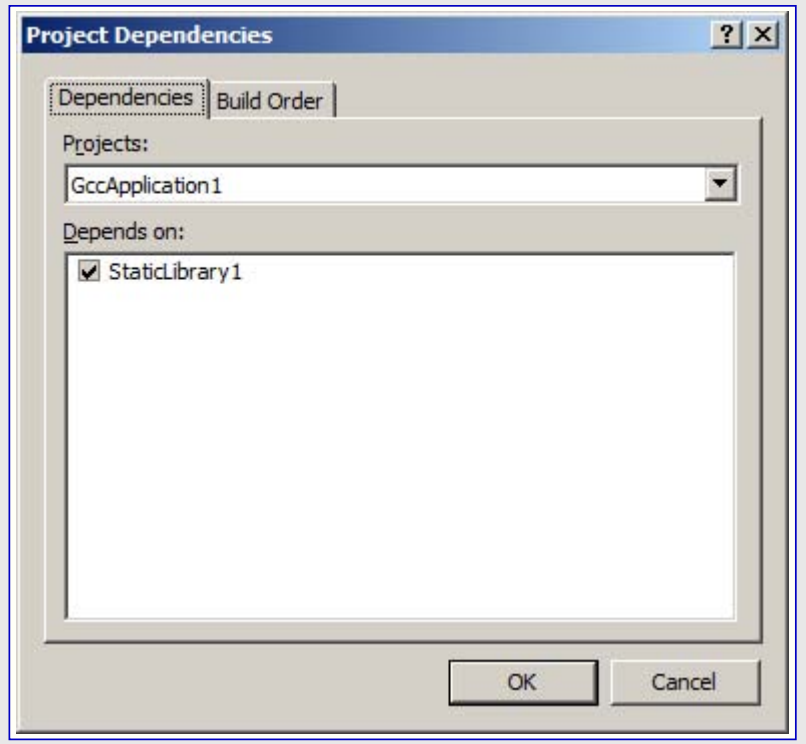
OKをクリックしてください。

図3-8. ライブラリ追加後のプロジェクトの表示



またProject(プロジェクト)⇒Project Dependencies(プロジェクト依存性)で追加した静的ライブラリを見ます。

図3-9. ライブラリ追加後のプロジェクト依存性の表示



3.2.3.5. ツールチェーン ライブラリの追加方法

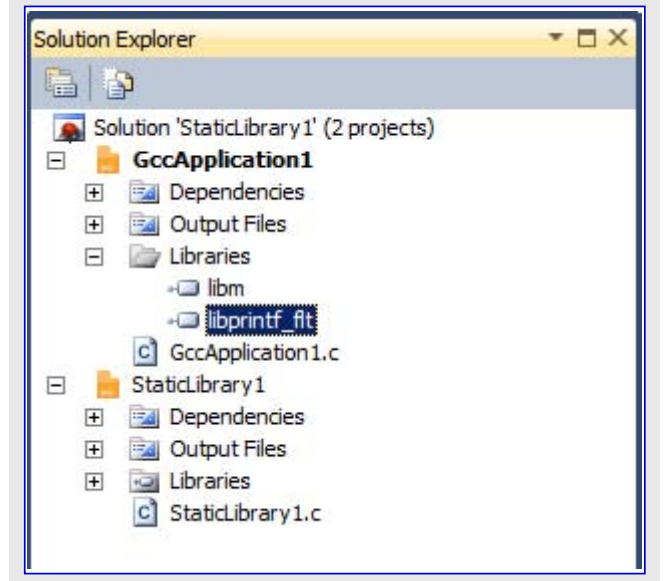
“Add Library(ライブラリ追加)”ウィザードを呼び出すためにプロジェクトまたはライブラリ節点で右クリックしてください。

Toolchain Libraries(ツールチェーン ライブラリ)タブを選んでください。ここはプロジェクトに対して現在選択されたツールチェーンに関する利用可能なツールチェーンライブラリを見ます。

追加したいライブラリを選んでください。

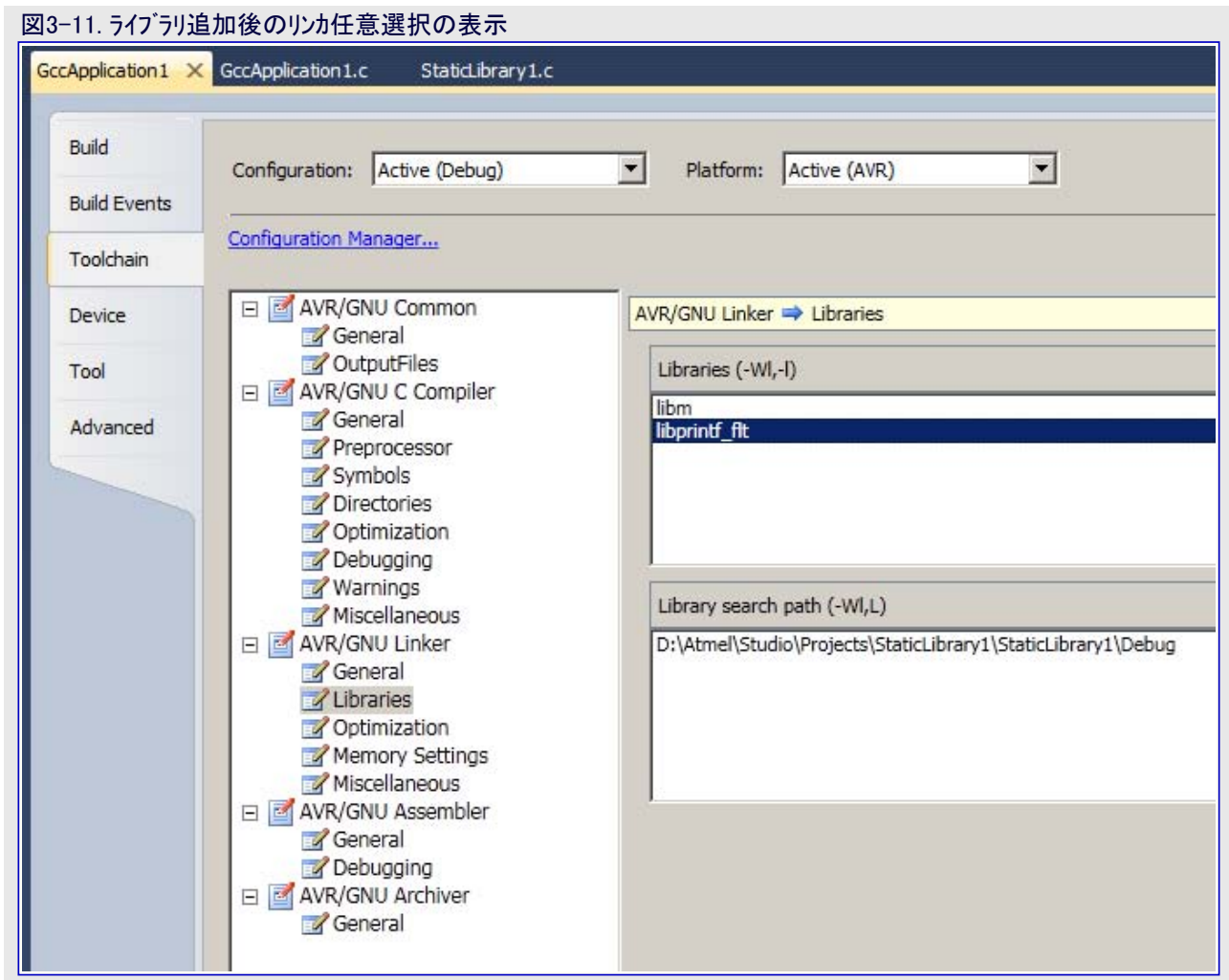
OKをクリックしてください。

図3-10. ライブラリ追加後のプロジェクトの表示



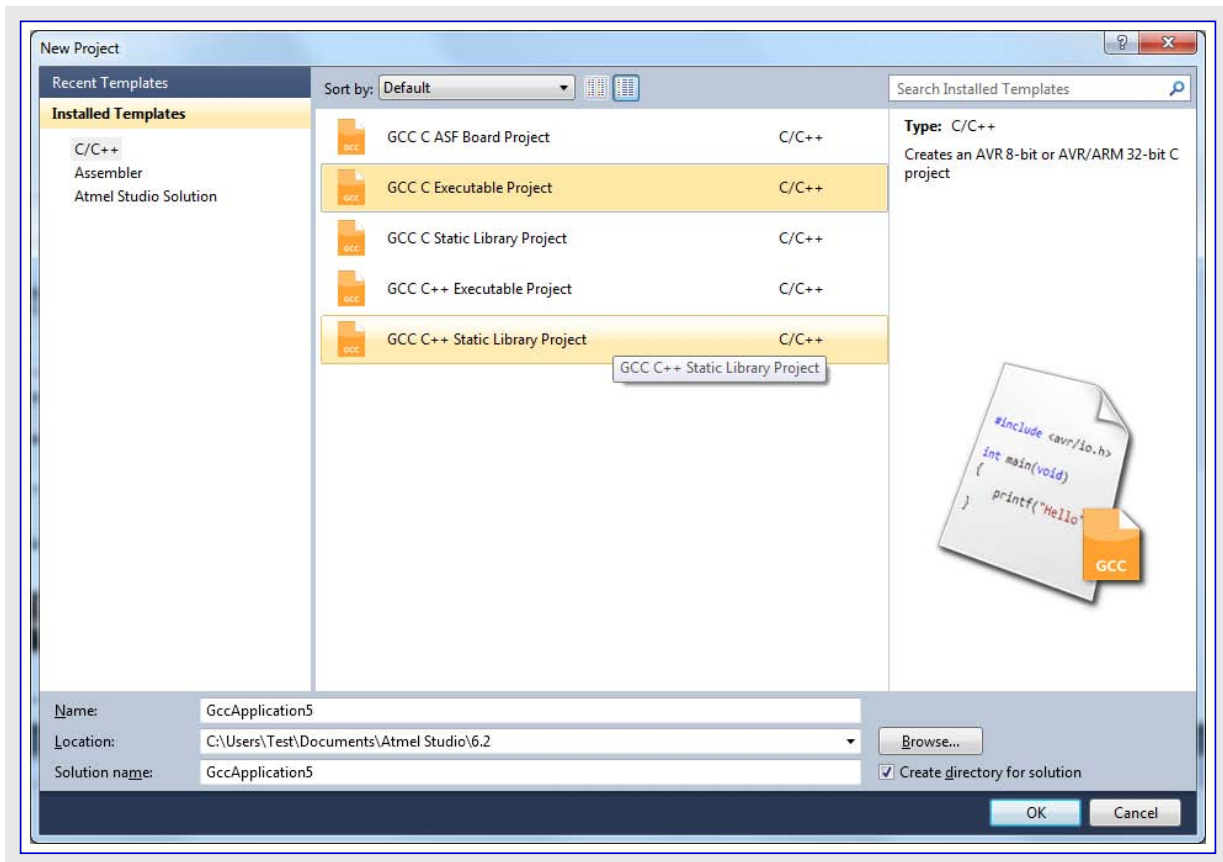
ツールチェーン リンカ設定で新しく追加されたライブラリを見ることもできます。

図3-11. ライブラリ追加後のリンカ任意選択の表示




3.2.4. SAMデバイス用新規GCCプロジェクト開始

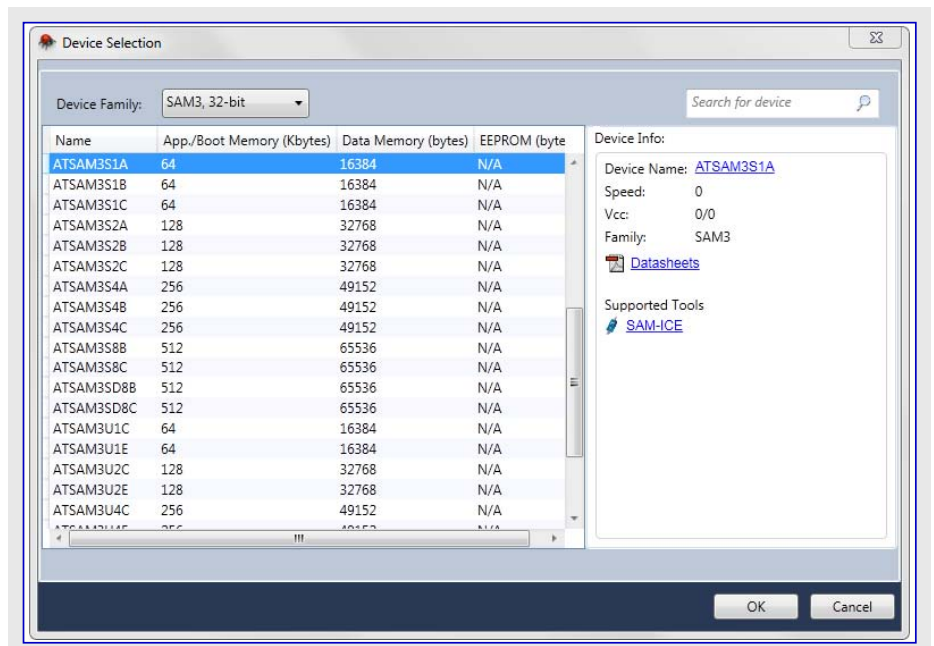
1. Project(プロジェクト)メニューからNew Project(新しいプロジェクト)を選ぶことによって新しいプロジェクトを作成してください。これはProject Wizard(プロジェクト ウィザード)を開きます。



2. 雛形としてC/C++⇒GCC C Executable Project(実行可能プロジェクト)を選び、その後にプロジェクト名を指定し、場所を選び、プロジェクトに対する解決策名を書いてください。既定によっていくつかのデバイス特有の関数とライブラリを含むプロジェクトにいくつかの始動ファイルが追加されます。この設定で満足された時にOKを押してください。
3. 雛形としてC/C++⇒GCC C Static Library Project(静的ライブラリプロジェクト)を選び、その後にプロジェクト名を指定し、場所を選び、プロジェクトに対する解決策名を書いてください。これはコードを再使用するための良い方法である静的ライブラリ(LIB)プロジェクトを作成します。

 **助言:** 静的ライブラリプロジェクトについてもっと学ぶには「3.2.6. 新規GCC静的ライブラリプロジェクト開始」項をご覧ください。

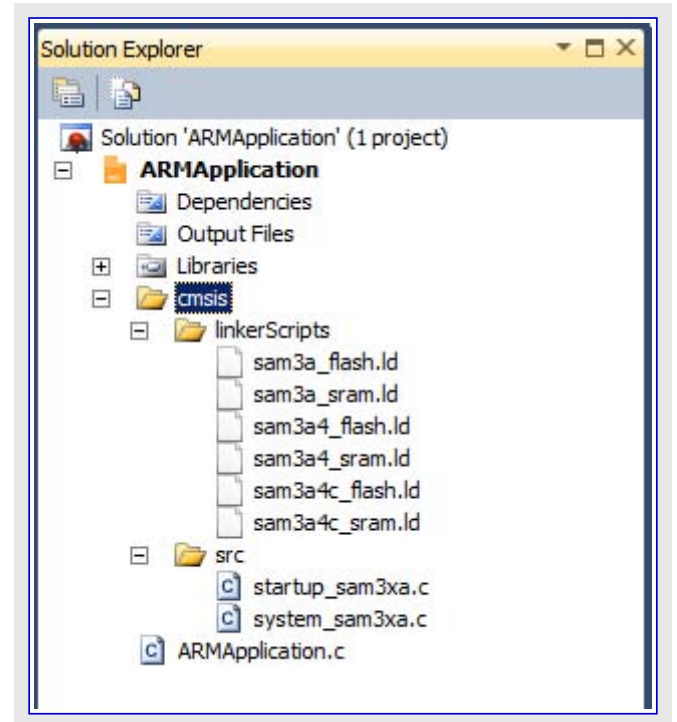
4. デバイス選択表が現れます。プロジェクトに対してデバイス系統をSAM3またはSAM4として選び、目的対象基盤を選んでください。始めるためにATSAM3S1Aデバイスを選ぶことができます。



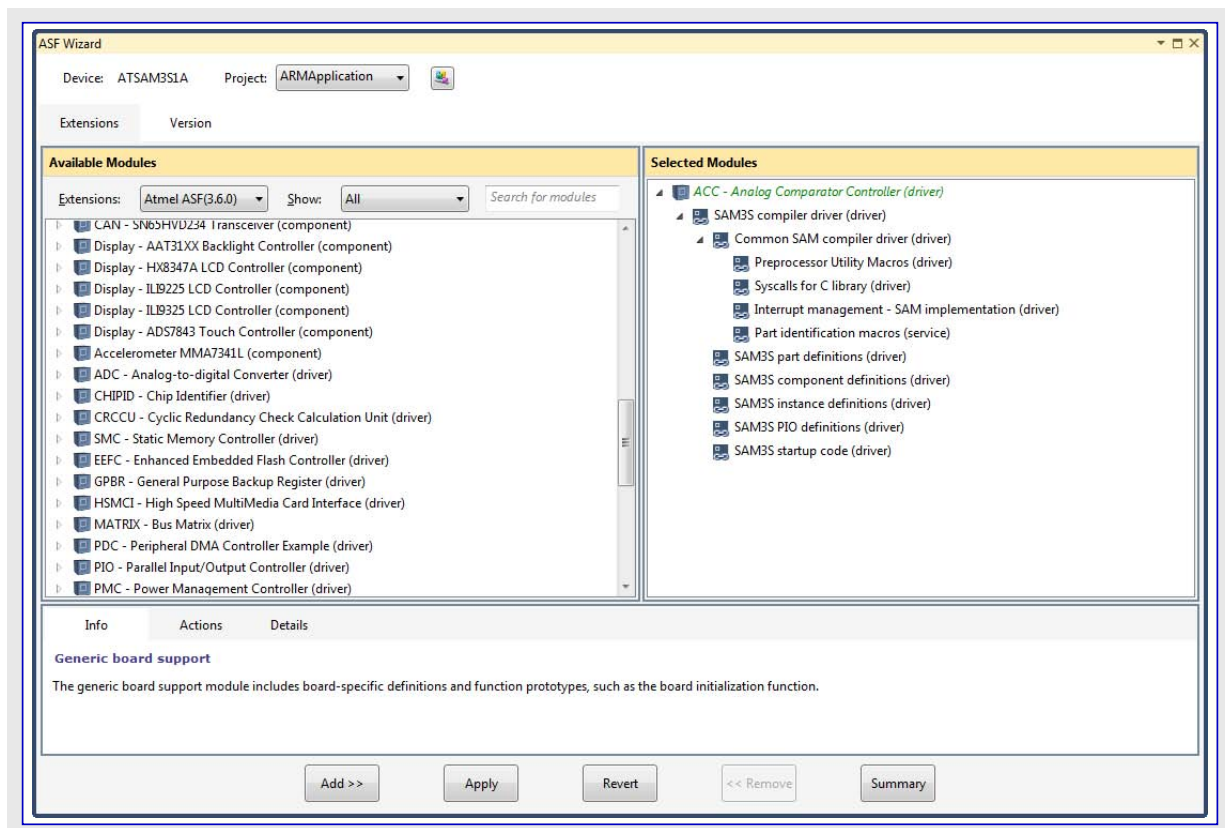
5. プロジェクト樹形が構成設定されます。段階2で作成された初期ファイルがプロジェクト節点に追加されていることに気付いてください。また、`main()`関数を含む初期ファイルがエディタで開かれます。以下は次のように作成されたファイルの一覧です。

- プロジェクトと同じ名前を持つファイルが作成され、既定によってプロジェクトに追加されます。これは`main()`関数を含みます。
- 始動ファイル(`startup*.c`)は”`cmsis¥src`”ディレクトリで利用可能です。これは全ての周辺機能に対する既定割り込み処理部を含みます。
- ”`cmsis¥src`”で利用可能なシステムファイル(`system*.c`)は始動で呼ばれるシステムレベルの初期化関数を提供します。
- デバイスに基づく適切な項を持つリンクスクリプトがプロジェクトフォルダ内の”`cmsis¥LinkerScripts`”に作成されます。
- `cmsis`フォルダ内のどれかのファイルを削除してそれを戻したいと望む場合、またはデバイスを変更した場合、適切なファイルを得るためにプロジェクトを右クリックして”`CMSIS Update from Microchip`(MicrochipからCMSIS更新)”をクリックしてください。

注: 他の選択が全くない場合を除き、`startup*.c`と`system*.c`のファイルの内容を変更することは推奨されません。これらの始動、システム、リンクスクリプトはArm静的ライブラリプロジェクトに対して作成されません。



6. 応用の開発と検証を楽にするため、Project(プロジェクト)⇒ASF Wizard(ASFウィザード)から呼び出されるドライバ選択ウィザードを使うこともできます。



ASFウィザードに於いて現在の構築構造と基板に対するプロジェクトで使用を望むドライバ、構成物、サービスを選ぶことができます。

7. そこで、開いたエディタ ウィンドウ内に以下のコードを書いてください。

```
#define MAXINT 200000

int main(void)
{
    unsigned int t=1000, k=0, l=5, pn=2;
    unsigned int primes[t];
    primes[0]=2;
    primes[1]=3;

    while (pn < t || primes[pn] < MAXINT)
    {
        for ( k = 0; k <= pn; k++)
        {
            if (1 % primes[k] == 0)
            {
                goto otop;
            }
            else
            {
                if (k == pn)
                    primes[pn++]=1;
            }
        }
    }
otop:
    l += 2;
}
return 0;
}
```

8. プロジェクトを構築してください。

3.2.5. コード編集

導入の以下の部分について前に見たのと同じコードを再利用します。

```
#define MAXINT 200000

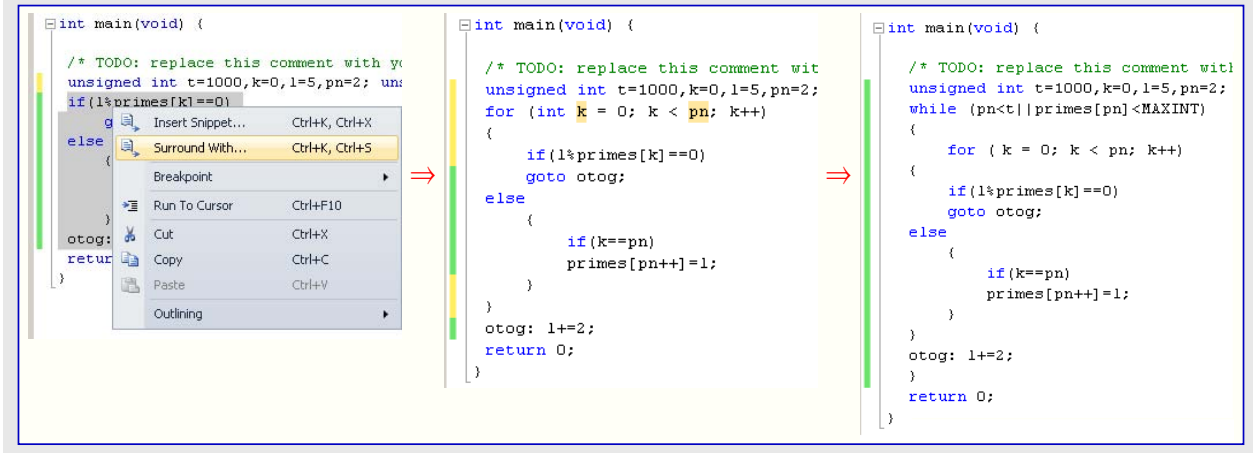
int main(void)
{
    unsigned int t=1000, k=0, l=5, pn=2;
    unsigned int primes[t];
    primes[0]=2;
    primes[1]=3;

    while (pn < t || primes[pn] < MAXINT)
    {
        for ( k = 0; k <= pn; k++)
        {
            if (1 % primes[k] == 0)
            {
                goto otop;
            }
            else
            {
                if (k == pn)
                    primes[pn++]=1;
            }
        }
    }
otop:
    l += 2;
}
return 0;
}
```

Micrchip StudioはMicrchipと第三者のプラグインによって、より一層豪華にされる豊かなエディタを持ちます。Microchip StudioはCソースコードの断片用の自動コード生成能力を持ちます。これを使うには(for,while,ifなどのような)条件構造で囲いたいコードの部分を選んで右クリックしてください。

コード断片を使って核となるソースに部品を追加することができます。いくつかの断片では変数名と脱出条件がIDE内の媒介変数で、故に1つの実体だけが変更されるような場合にその断片内で全ての実体も変更し、そのようなものは”for”繰り返しの場合です。

表3-1. “Surround With”使い方



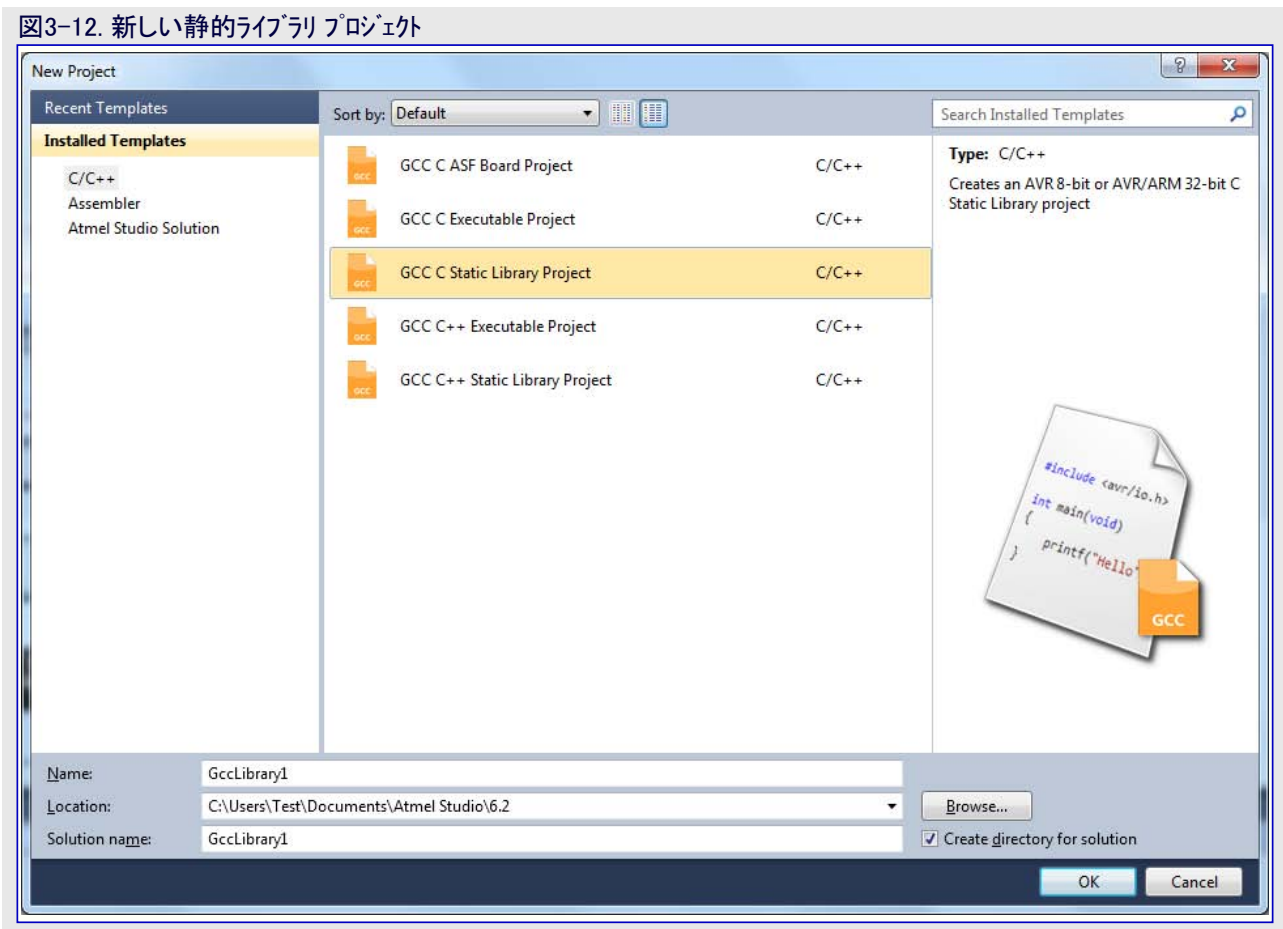
3.2.6. 新規GCC静的ライブラリプロジェクト開始

3.2.6.1. 静的ライブラリの理由

静的ライブラリ(LIB)はコード再利用のための良い方法です。全てのプログラムで同じルーチン/関数を再作成するのではなく、使用者はそれらを一度書いて、その機能が必要な応用から参照します。

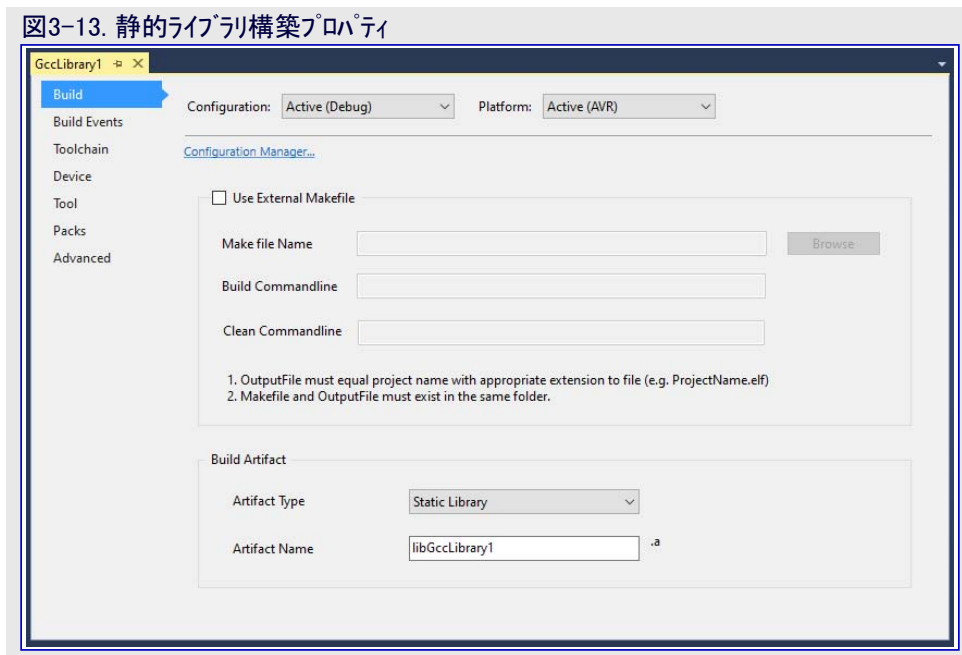
3.2.6.2. 新規静的ライブラリプロジェクト作成

図3-12. 新しい静的ライブラリプロジェクト



静的ライブラリプロジェクトを作成するためにOKをクリックしてください。プロジェクトと同じ名前を持つ既定ソースファイルが解決策に追加されます。あなたはその後あなたのルーチン/関数を書いてコンパイルするかもしれません。このプロジェクトに新しいファイルやヘッダファイルを追加することもできます。

メニューのProject(プロジェクト)⇒「あなたのプロジェクト名」 Properties(あなたのプロジェクト名のプロパティ)でプロジェクト プロパティを開いてください。このメニュー項目は静的ライブラリ プロジェクトが開いている時にだけ利用可能です。Build(構築)プロパティ ページを選んでください。ここでArtifact Type(成果物の種類)がStatic Library(静的ライブラリ)として選ばれるのを見ます。



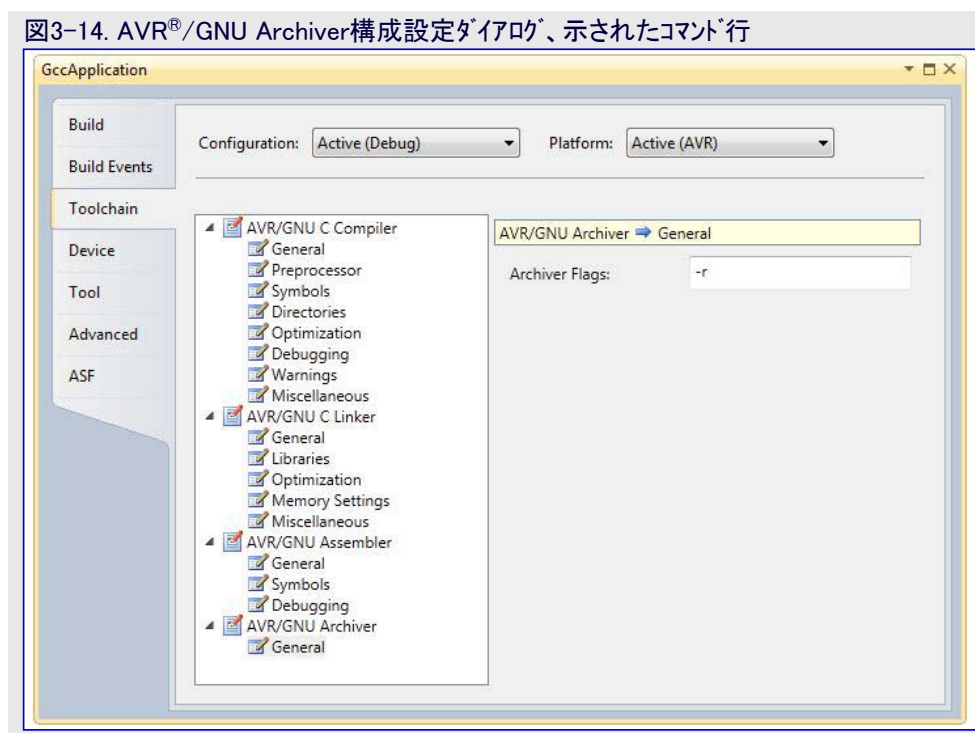
Build(構築)メニューからBuild Solution(解決策構築)を選ぶことによってプロジェクトをコンパイルしてください。これは他のプログラムによって使うことができる静的ライブラリを作成します。

3.2.6.3. 静的ライブラリ プロジェクト任意選択 (AVR®/GNU Archiver)

AVR/GNU Archiver(纏め部)(avr-ar)はライブラリとしても知られる単一纏めファイル内にオブジェクト ファイルの集合を結合します。

メニューのProject(プロジェクト)⇒「あなたのプロジェクト名」 Properties(あなたのプロジェクト名のプロパティ)でプロジェクト プロパティを開いてください。このメニュー項目は静的ライブラリ プロジェクトが開いている時にだけ利用可能です。静的ライブラリ任意選択を構成設定するために、Toolchain(ツールチェーン)プロパティ タブをクリックしてください。

ツールチェーン プロパティ ページに於いて、AVR/GNU Archiver(纏め部)がActive(活性)で許可されたことを見ます。静的ライブラリ プロジェクトに対してAVR/GNU Linkerが禁止されることも見るかもしれません。



上のGeneral(全般)任意選択内のArchiver Flags(纏め部フラグ)でAVR/GNU Archiver(纏め部)フラグを設定することができます。そこで、Build(構築)メニューからBuild Solution(解決策構築)を選ぶことによってプロジェクトを保存してコンパイルしてください。

3.2.7. GCCプロジェクト外任意選択と構成設定

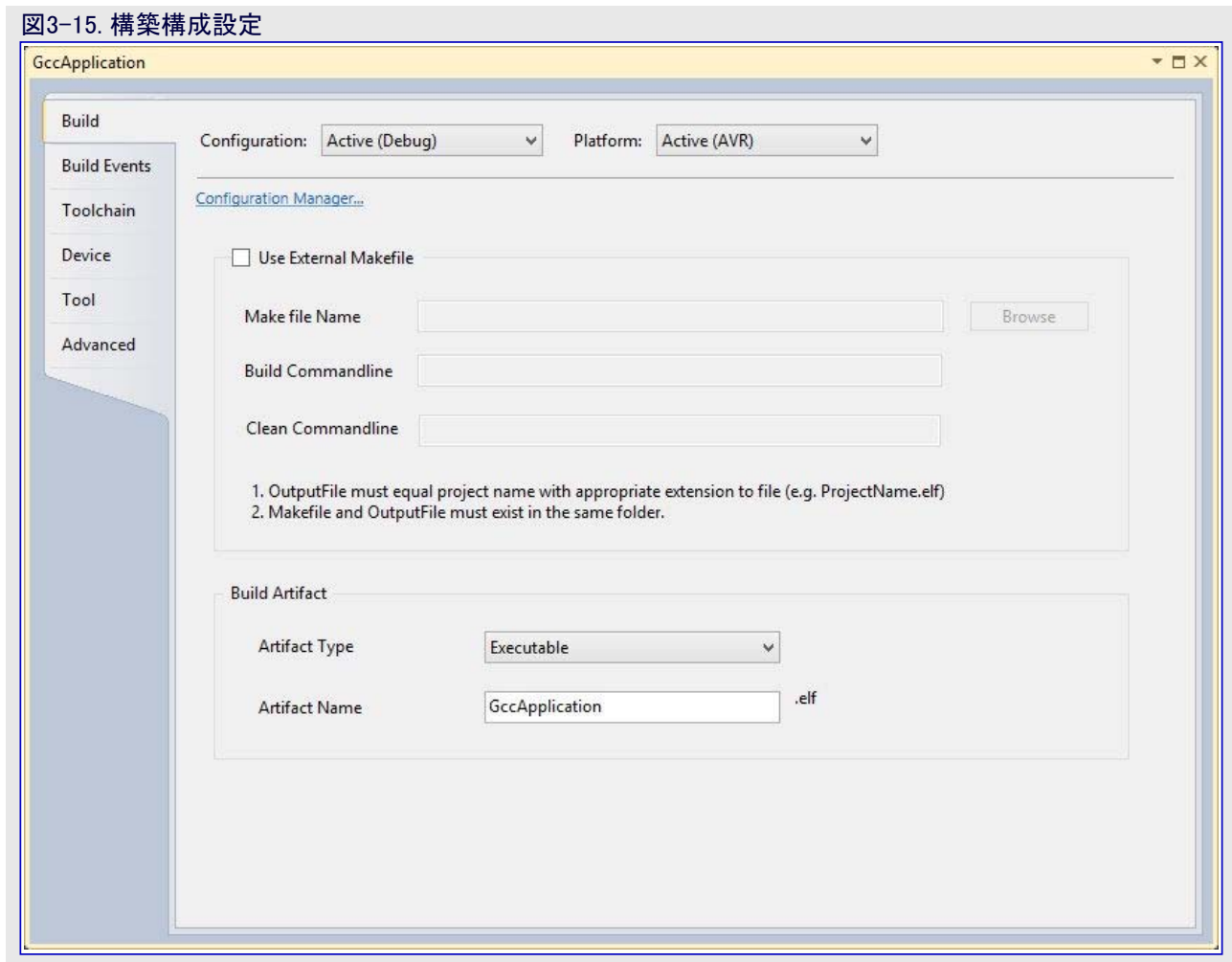
プロジェクト任意選択と構成設定はSolution(解決策)エクスプローラ⇒Project Properties(プロジェクト プロパティ)上を右クリックするか、またはAlt+Enterを押下することのどちらかによって構成設定することができます。

これは7つのタブを持つプロジェクト プロパティウィンドウを呼び出します。

タブが構成設定指定のプロパティを支援する場合、そのタブは次の2つの引き出しメニューを持ちます。Configuration(構成設定)領域は変更するためのプロジェクト構成設定を定義します。既定によって、デバッグ(Debug)と公開(Release)の各々のプロジェクトで2つの構成設定が提供されます。Platform(基盤)領域はAVRを構成設定します。タブが構成設定と無関係なプロパティを支援する場合、Configuration(構成設定)とPlatform(基盤)の領域は禁止されます。

注: 変更された時に必ずプロジェクト ファイルでの変更を更新するためにFile(ファイル)メニューまたはツールバーから”Save All(全てを保存) (Ctrl Shift S)”を使ってください。

3.2.7.1. 構築任意選択



Build(構築)タブのページで、あなたのプロジェクトに対して外部メークファイルの使用を望むかどうかを構成設定することができます。その場合、単にUse External Makefile(外部メークファイル使用)チェック枠をチェックしてメークファイルの正しいパスを選ぶためにBrowse(検索)してください。

Build Commandline(構築コマンド行)はプロジェクトに対して構築が呼び出される時の外部メークファイルのために提供されます。既定構築目的の対象は”all(全て)”です。

Clean Commandline(解消コマンド行)はプロジェクトに対して解消が呼び出される時の外部メークファイルのために提供されます。既定楷書目的の対象は”clean(解消)”です。

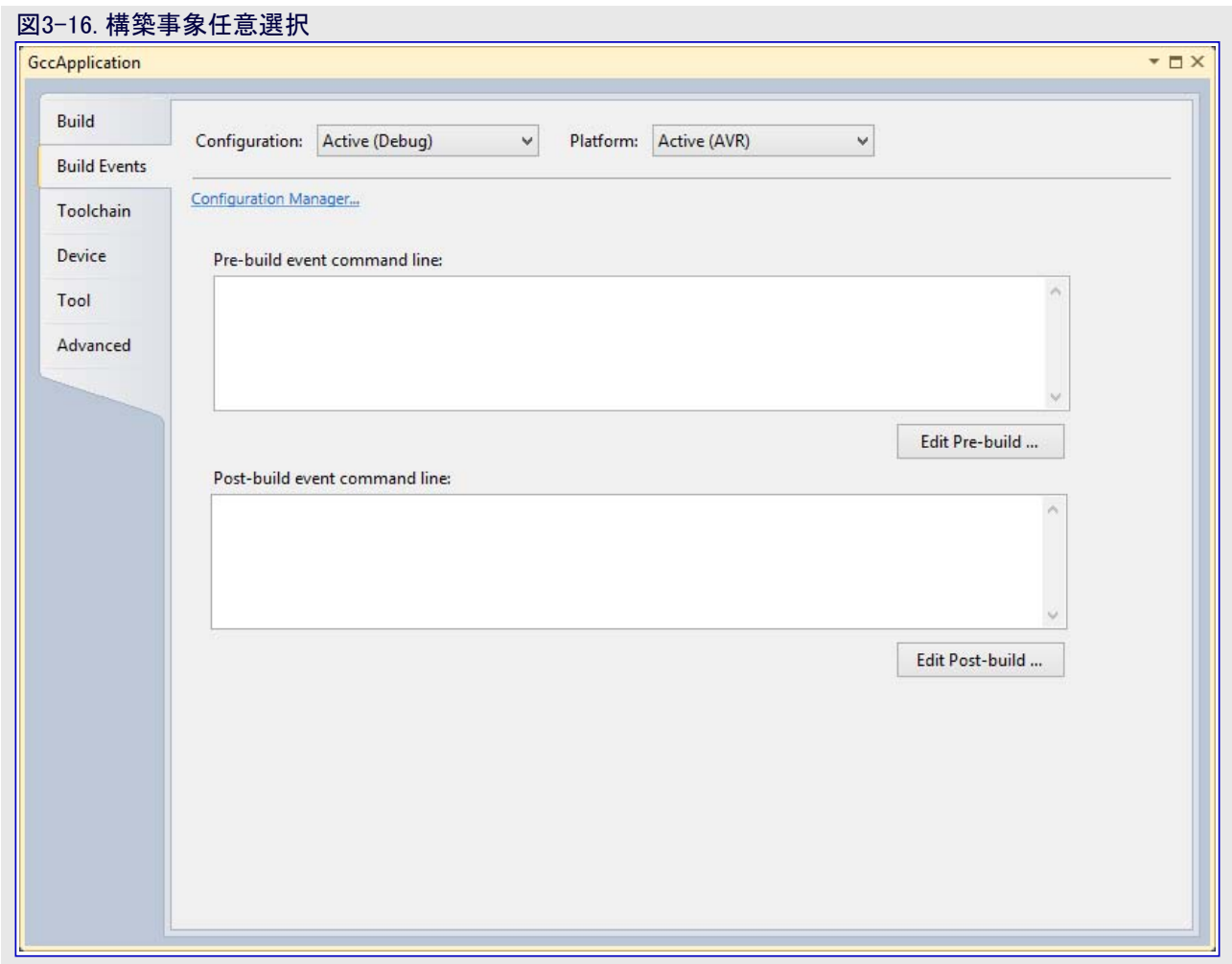
外部メークファイル構成設定の他にも、構築に対して応用の形式を指定することもできます。この任意選択はArtifact Type(成果物の種類)コンボ枠を用いて選ぶことができるExecutable(実行可能)またはStatic Library(静的ライブラリ)です。

注: 独自のメークファイルは以下のようなこれらの条件を満たさなければなりません。

1. 目的対象名はプロジェクト名と同じでなければなりません。
2. メークファイルと目的対象は(NTFSSリンクでも参照することができる)同じフォルダに存在しなければなりません。

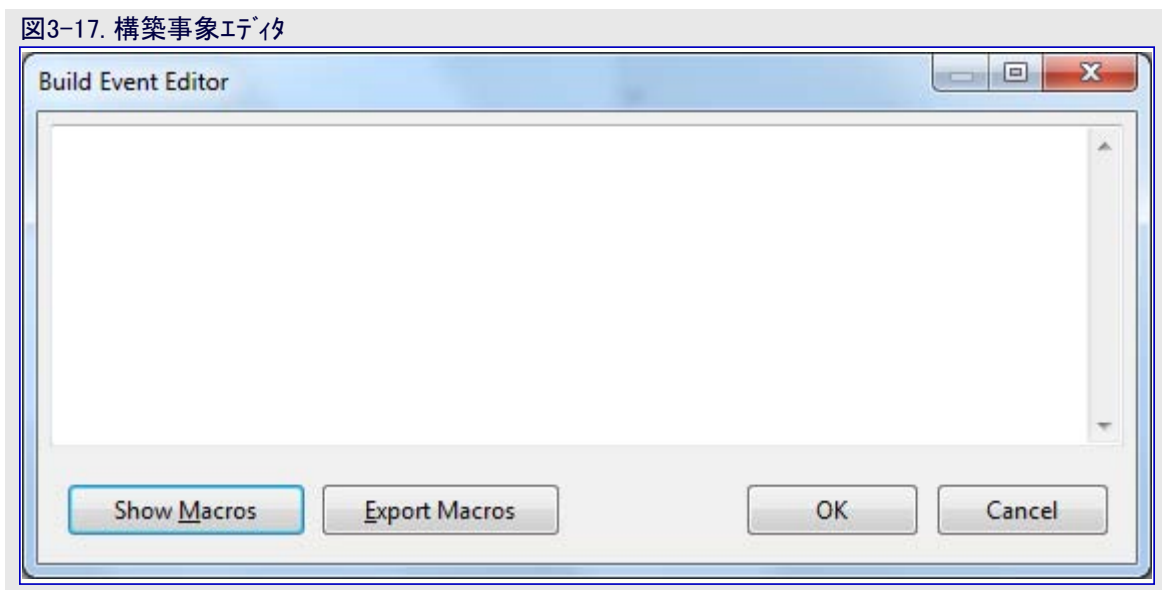
3.2.7.2. 構築事象

図3-16. 構築事象任意選択



Build Events(構築事象)タブはPre-build(構築前)とPost-build(構築後)のスクリプトによって開始される各構成設定に対して計画された事象の一覧を含みます。これらの事象はEdit Pre-build...(構築前編集)またはEdit Post-build...(構築後編集)の釦のどちらかをクリックすることによって追加、削除、または編集することができます。これらの釦をクリックすることで、以下のダイアログであなたの命令を手動で追加すべきです。現公開版の時点で、他の利用可能な応用とのリンクとしてそれら内で宣言された環境変数と値を使うことが可能です。その機能を使うにはShow Macros(マクロ表示)釦を押してください。

図3-17. 構築事象エディタ



マクロ

コマンド行編集枠で挿入するのにマクロ/環境変数の一覧を表示するために編集枠を広げてください。

マクロ表

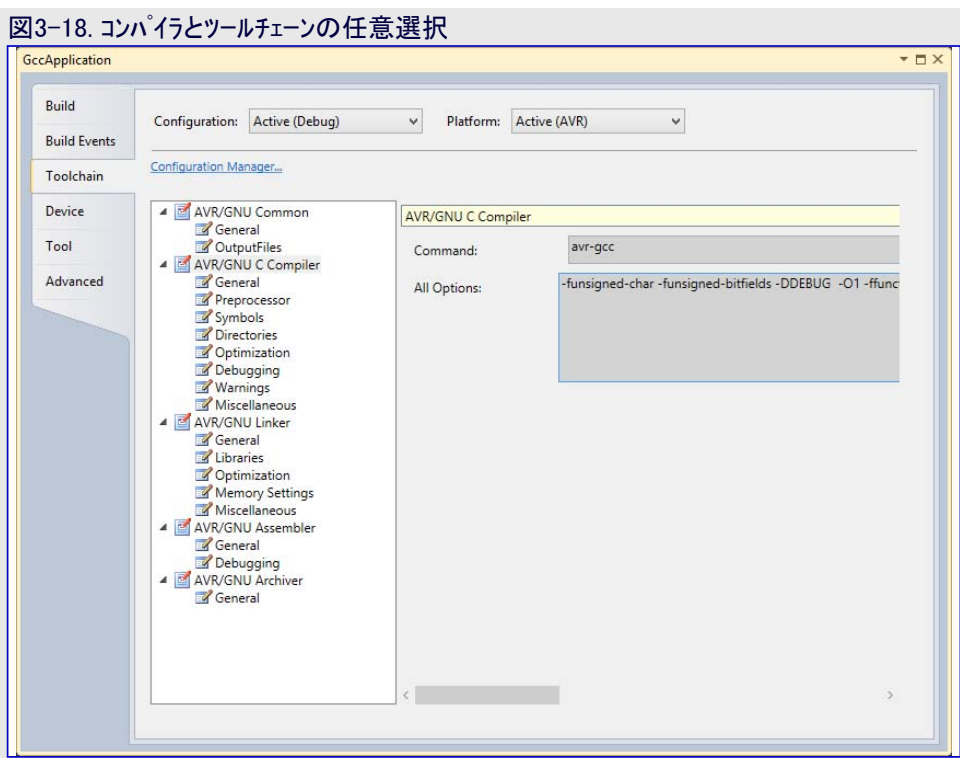
利用可能なマクロ/環境変数とその値を一覧してください。コマンド行編集枠内へ挿入するのに一度に1つのマクロだけを選ぶことができます。MSBuildはプロジェクトファイルとMSBuildバイナリについての情報を格納する予約されたプロパティの組を提供します。これらのプロパティは編集枠で一覧にされるかもしれません。

Microchip Studioに特有な説明については下のマクロ/環境変数をご覧ください。

表3-2. Microchip Studio構築マクロ表




マクロ	説明
\$(AVRSTUDIO_EXE_PATH)	(ドライブとパスで定義される)Microchip Studioのインストールディレクトリ
\$(SolutionDir)	(ドライブとパスで定義される)解決策のディレクトリ
\$(SolutionPath)	(ドライブ、パス、基本名、ファイル拡張子で定義される)解決策の絶対パス名
\$(SolutionFileName)	解決策の名前
\$(SolutionName)	解決策の基本名
\$(SolutionExt)	解決策のファイル拡張子。ファイル拡張子の前の'.'を含みます。
\$(Configuration)	現在のプロジェクト構成設定の名前。例えば、"Debug"
\$(Platform)	現在の目的対象基盤の名前。例えば、"AVR"
\$(DevEnvDir)	(ドライブとパスで定義される)Microchip Studioのインストールディレクトリ
\$(ProjectVersion)	プロジェクトのバージョン
\$(ProjectGuid)	プロジェクトの固有識別子
\$(avrdevice)	現在選ばれているデバイスの名前
\$(avrdeviceseries)	選ばれているデバイスの系統。Microchip Studioによって内部的に使用
\$(OutputType)	現在のプロジェクトが実行可能型または静的ライブラリ型かを定義
\$(Language)	現在のプロジェクトの言語。例えば、C、CPP、またはアセンブラ
\$(OutputFileName)	(基本ファイル名として定義される)構築用基本出力ファイルの名前
\$(OutputFileExtension)	構築用基本出力ファイルのファイル拡張子。ファイル拡張子の前の'.'を含みます。
\$(OutputDirectory)	出力ファイルディレクトリの絶対パス
\$(AssemblyName)	構築用基本出力のアセンブリ名
\$(Name)	プロジェクトの基本名
\$(RootNamespace)	プロジェクトの基本名
\$(ToolchainName)	ツールチェーンの名前
\$(ToolchainFlavour)	ツールチェーン コンパイラの名前

3.2.7.3. コンパイラとツールチェーンの任意選択



AVR® GNU C コンパイラ任意選択

表3-3. AVR® GNU Cコンパイラ任意選択

任意選択	説明
全般任意選択	
-mcall-prologues	プロローグとエピローグの機能にサブルーチンを使用
-mno-interrupts	割り込み禁止なしでスタックポインタを変更
-funsigned-char	既定char型は符号なし
-funsigned-bitfield	既定ビット領域は符号なし
プロセッサ任意選択	
-nostdinc	システムディレクトリを検索しない。
-E	前処理(プリプロセッサ)のみ
シンボル任意選択	
ソース内のシンボルを定義(-D)または定義解除(-U)することができます。新しいシンボル宣言は下のインターフェース鉤を用いて追加、変更、再整理をすることができます。	
<ul style="list-style-type: none"> •  新しいシンボル追加。これと以下の全アイコンはMicrochip Studioインターフェースの他の部分で同じ意味で再利用されます。 •  シンボル削除 •  シンボル編集 •  解析順に於いてシンボルを上に移動 •  解析順に於いてシンボルを下に移動 	
インクルードディレクトリ	
インクルードされるヘッダと定義指令の全てを含み、シンボルと同じインターフェースを用いて編集することができます。	
最適化任意選択	
最適化レベル(引き落としメニュー) -O0, -O1, -O2, -O3, -Os	最適化なし、速度最適化(レベル1~3)、大きさ最適化
他の最適化フラグ(手入力方式)	ここで基盤と必要条件に対して指定する最適化フラグを書くべきです。
-ffunction-sections	ゴミ回収用関数準備。関数が決して使われないなら、メモリがかき集められます。
-fpack-struct	構造体メンバを共に一括化
-fshort-enums	列挙型によって必要とされるのと同数のバイトだけを割り当て
-mshort-calls	8Kバイトを超えるデバイスでRJMP/RCALL制限範囲命令を使用
デバッグ任意選択	
デバッグレベル(引き落としメニュー) none, -g1, -g2, -g3	ソースコードで残されるまたは挿入されるコードとヘッダを追跡してデバッグするレベルを指定
他のデバッグ任意選択(書式領域)	基本構造特有デバッグ任意選択
警告メッセージ出力任意選択	
-Wall	全ての警告
-Werror	警告を異常に格上げ
-fsyntax-only	構文だけ検査
-pedantic	GNUに対する適合性検査、非標準プログラミング実行で警告揭示
-pedantic-errors	同上、加えて警告を異常に格上げ
その他の任意選択	
他のフラグ(書式領域)	他のプロジェクト特有フラグを入力
-v	詳細
-ansi	ANSIプログラムを支援
-save-temps	一時ファイルを削除しない

3.2.7.3.1. AVR® GCCリンク任意選択

表3-4. AVR® GCCリンク任意選択

任意選択	説明
-Wl -nostartfiles	標準ファイルを使わない。
-Wl -nodefault	既定ライブラリを使わない。
-Wl -nostdlib	始動または既定ライブラリなし
-Wl -s	全てのシンボル情報を省く。
-Wl -static	静的にリンク
ライブラリ任意選択	
ライブラリ -Wl, -l (書式領域)	 の鉤を用いて、ここでライブラリ名の追加、優先付け、編集ができます。
ライブラリ検索パス -Wl, -L (書式領域)	上と同じインターフェースでリンクが動的にリンクされるライブラリを探す所のパスの追加、優先付け、編集ができます。
最適化任意選択	
-Wl, -gc-sections	未使用領域ゴミ回収
--rodata-writable	書き込み可能空間に読み込み専用データを配置
-mrelax	分岐緩和
その他の任意選択	
他のリンクフラグ (書式領域)	他のプロジェクト特有フラグを入力

3.2.7.3.2. AVR®アセンブラ任意選択

表3-5. AVR®アセンブラ任意選択

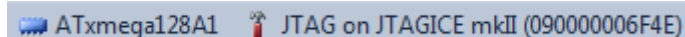

任意選択	説明
最適化任意選択	
アセンブラフラグ (書式領域)	その他のアセンブラフラグ
インクルードパス (書式領域)	ここで基本構造と基盤特有のインクルードしたファイルへのパスの追加、優先付け、編集ができます。
-v	アセンブラ出力で版番号を公表
デバッグ任意選択	
デバッグレベル(引き落としメニュー) -Wa -g1, -Wa -g2, -Wa -g3	シンボルデバッグとソース挿入デバッグのレベルを定義

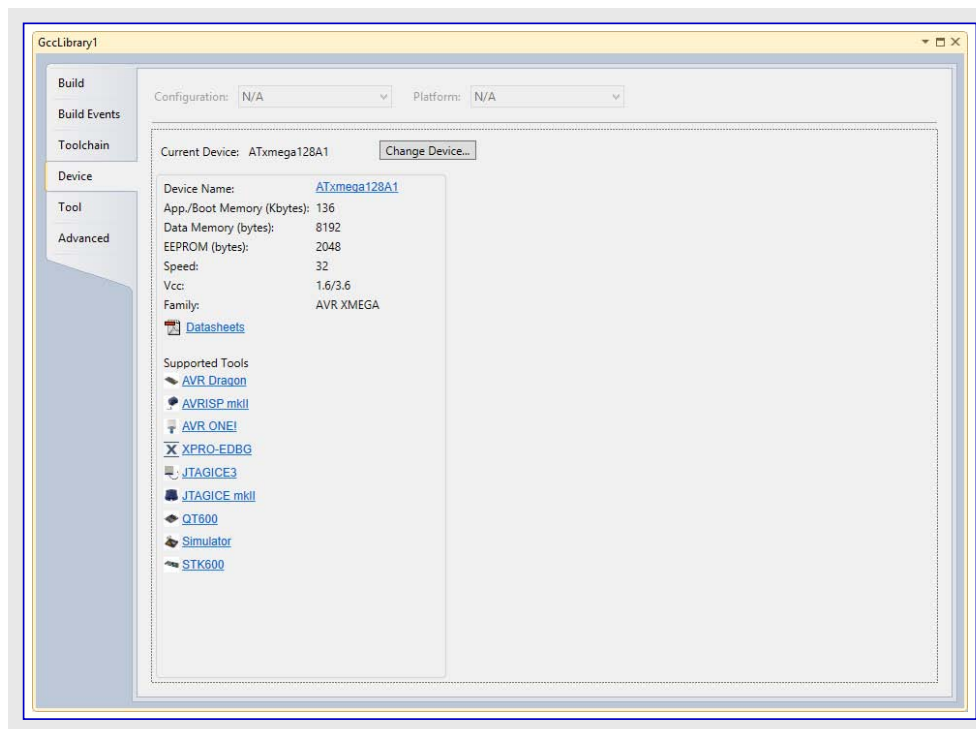
3.2.7.4. デバイス任意選択

このタブは現在のプロジェクトに対するデバイスを選んで変更することを許し、デバイス選択部と同じです。図3-5をご覧ください。



助言: 編集集中に素早くこのタブを得るにはDevice and Debugger(デバイスとデバッグ)ツールバーでDevice鉤をクリックしてください。

 ATxmega128A1  JTAG on JTAGICE mkII (090000006F4E)



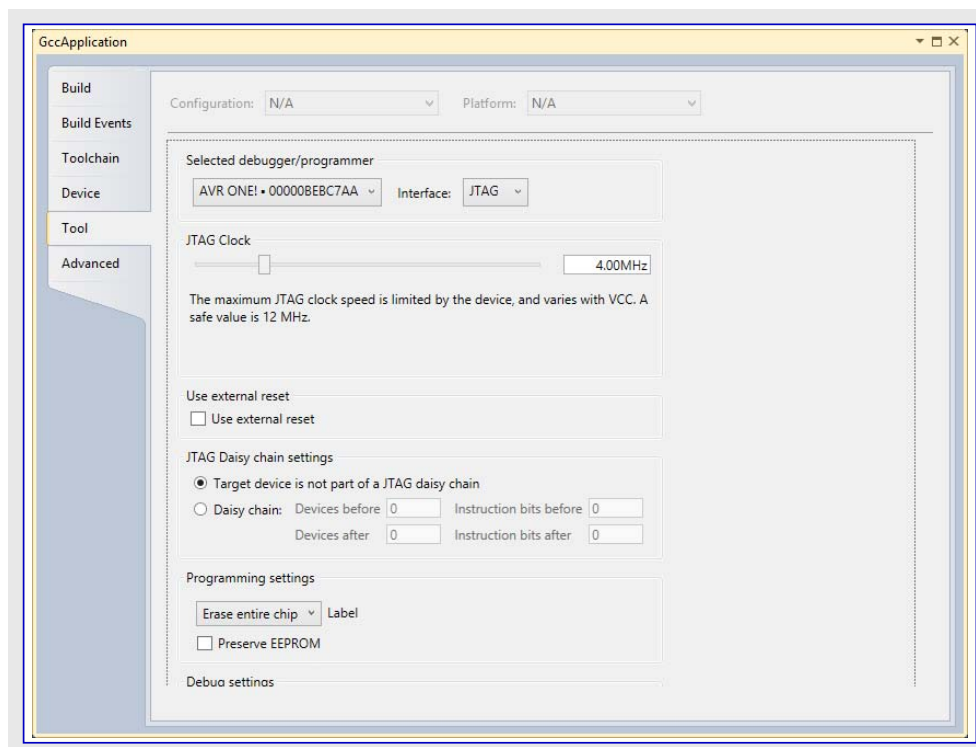
3.2.7.5. ツール任意選択

このタブは現在のプロジェクトに対してデバッグ基盤を選んで変更することを許します。



助言: 編集集中に素早くこのタブを得るには **Device and Debugger** (デバイスとデバッグ) ツールバーで **Device** 釘をクリックしてください。

ATxmega128A1 JTAG on JTAGICE mkII (090000006F4E)



引き落とし一覧からツール/デバッグを選んでください。現在の選択が示されます。

引き落とし一覧からインターフェースを選んでください。現在の選択が示されます。

注: 現在のデバイス選択については有効なツールとインターフェースだけが示されます。

更なる属性は選んだツールとインターフェースに依存します。

JTAG

プログラミング インターフェースとしてJTAGを選んだ場合、クロック速度、外部リセット使用、デバッグチェーン設定がおそらく利用可能です。これはツールとデバイスに依存します。

JTAG Clock (JTAGクロック)

JTAGクロックはツールがそれでデバイスのクロック駆動を試みる最大速度です。クロック範囲は各種ツールとデバイスに対して異なります。制限がある場合、それらはクロック摺動子の下のメッセージで述べられます。

クロックは手動(全ツール)、自動(SAM-ICEのみ)、適応(SAM-ICEのみ)に設定することができます。

Use external reset (外部リセット使用)

チェックされた場合、ツールはデバイスへの接続を試みる時に外部リセット線をLowに引きます。

JTAG daisy chain settings (JTAGデバッグチェーン設定)

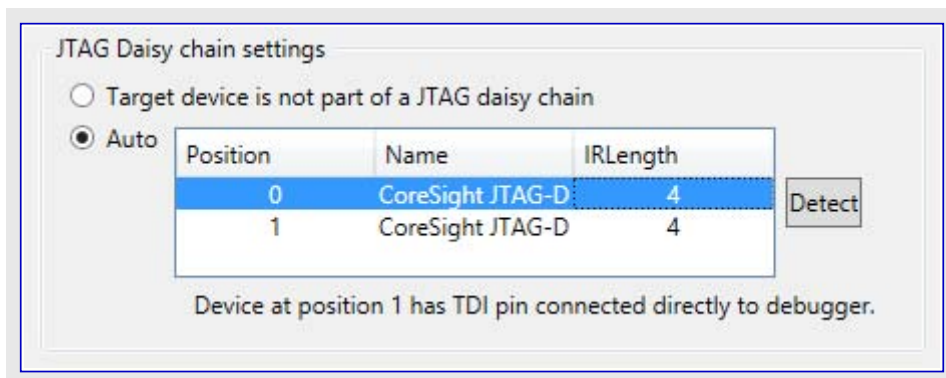
プログラミングのためにデバイスに関連するJTAGデバッグチェーン設定を指定してください。

Target device is not part of a JTAG daisy chain : 目的対象デバイスがデバッグチェーンの一部でない時にこの任意選択を選んでください。

Daisy chain – 手動 : 基板上システムでプログラミングする場合にJTAGデバッグチェーンの手動構成設定を許します。

- **Devices before** – 目的対象デバイスに先行するデバイス数を指定してください。
- **Instruction bits before** – 目的対象デバイスに先行する全デバイスの命令レジスタの総(ビット)数を指定してください。
- **Devices after** – 目的対象デバイスに後続するデバイス数を指定してください。
- **Instruction bits after** – 目的対象デバイスに後続する全デバイスの命令レジスタの総(ビット)数を指定してください。

Daisy chain – Auto : JTAGデバッグチェーンでデバイスを自動的に検出します。JTAGデバッグチェーンでデバイスを選ぶことを許します。自動検出はSAMデバイスに対してだけ支援されます。



PDI

PDIインターフェースはPDIクロック速度の1つの設定だけを持ちます。

PDI Clock(PDIクロック)はツールがそれでデバイスのクロック駆動を試みる最大速度です。クロック範囲は各種ツールとデバイスに対して異なります。制限がある場合、それらはクロック摺動子の下のメッセージで述べられます。

クロックは全てのツールで調整することができず、故に空の**Interface settings**(インターフェース設定)ページが提示されます。

Programming and debug settings (プログラミングとデバッグの設定)

引き落としメニューに於いてプログラミング/デバッグ周回の間で消去されるべきメモリの部分を指定することが可能です。

- **Skip programming** – プログラミングが全く起こるべきでないことを指定します。ツールはメモリ内の以前のプログラムに結び付きます。
- **Erase only program area** – メモリのプログラム領域だけが消去されることを指定します。
- **Erase entire chip** – チップ全体が消去されることを指定します。

“**Preserve Eeprom**(EEPROM保護)”任意選択はデバッグ作業開始時にEEPROMデータが書かれるべきかどうかを決めます。これに応じて**EESAVE**ヒューズは設定と解除が行われます。

デバッグ作業の開始でデバイスが書かれる時に、既定の動作はデバイスの内容を消去(利用可能ならばチップ消去)することです。これは“**Programming settings**(プログラミング設定)”下の引き落とし枠から違う任意選択を選ぶことによって変更することができます。

Keep timer running in stop mode (停止動作で計時器走行維持)

チェックされると、プログラムで設定された計時器は例えばプログラムが中断点(ブレークポイント)で中断するか、または停止されても走行を保ちます。

この設定は一定のツール/インターフェースに対してだけ許されます。

Override Vector Table Offset Register (ベクタ表変位レジスタ上書き)

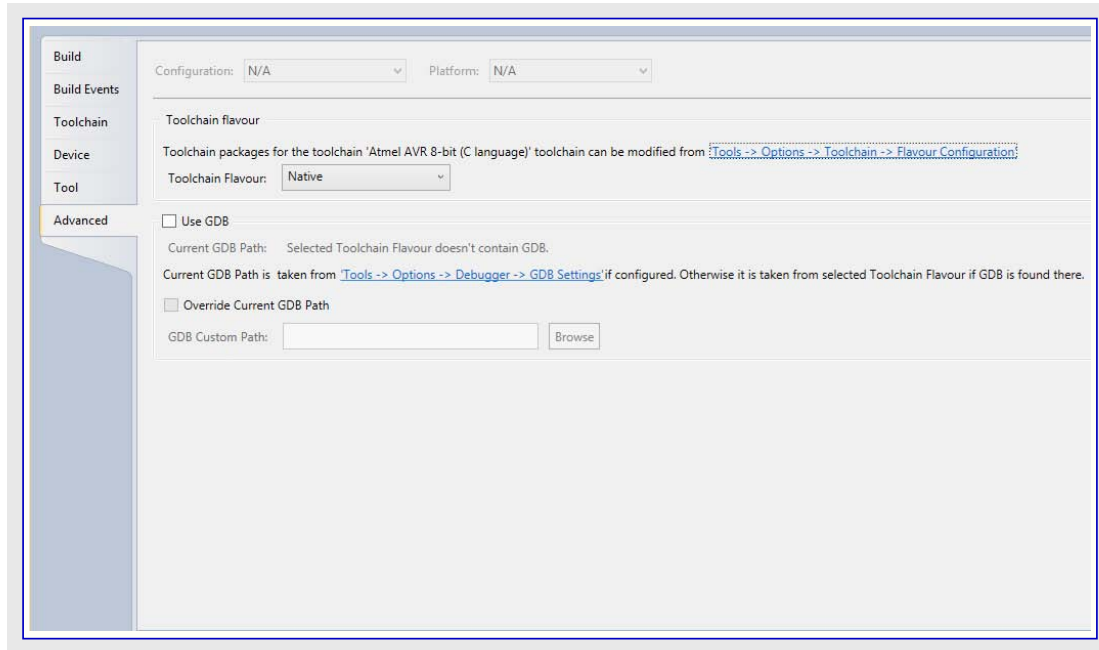
リセットで、指定したアドレスからPCとSPを設定します。

注: このページで選んだツールは**Start Without Debugging**(デバッグなしで開始)命令でも使われます。これはデバッグ能力を持たないツールも選ぶことができる理由です。より多くの情報については「[4.5. デバッグなしでの開始](#)」をご覧ください。

3.2.7.6. 高度な任意選択

3.2.7.6.1. Toolchain Flavour設定 (ツールチェーン味付け設定)

Toolchain Flavour(ツールチェーン味付け)で構成設定されたツールチェーンのパスがプロジェクトの構築に使われます。現在のプロジェクトに新しいツールチェーン味付けを追加することが可能です。新しい味付けの構成設定と追加については「[10.3.11. ツールチェーン](#)」をご覧ください。

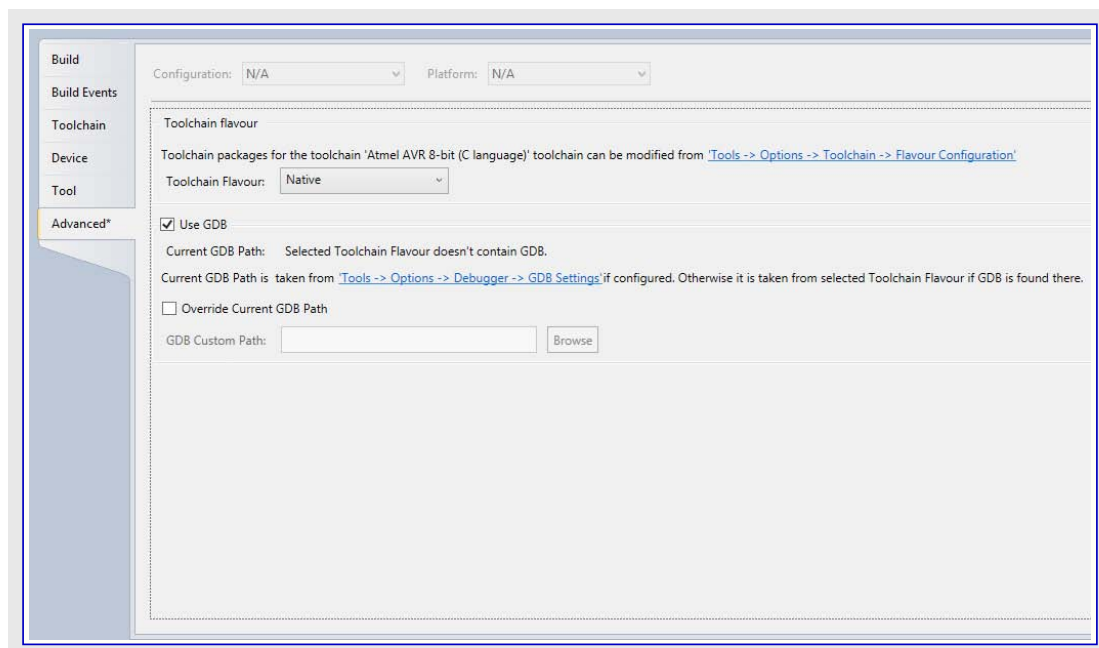


3.2.7.6.2. Use GDB (GDBを使用)

本項はGDBが現在のプロジェクトに対して使われなければならないかどうかを選ぶことを許します。現在のGDBパスは以下によって計算されます。

1. 現在のGDBパスはTools(ツール)⇒Options(任意選択)⇒Debugger(デバッグ)⇒GDB Settings([10.3.12. GDB設定](#))から取られます。
2. さもなければ、そこでGDBが見つかるなら、選ばれたツールチェーン味付けから取られます。

現在のGDBパスは”Advanced(高度な)”プロジェクトプロパティ頁で”Override Current GDB Path(現在のGDBパス上書き)”任意選択を使う時に上書きされます。下図をご覧ください。



注: “Use GDB(GDB使用)”任意選択はArmに基づくデバイスに対して既定によって許可され、GDBが許可される場合、AVR 32ビットデバイスに対して以下の警告も示されます。



3.2.7.7. 他のメモリ形式でのELFファイル作成

全てのメモリ形式に対するデータを含むELFファイルをGCCコンパイラで作ることができます。データ領域は以下のコード例で示されるようにコードで指定されます。

3.2.7.7.1. tinyAVR[®]、megaAVR[®]、XMEGA[®]デバイス用ELFファイル作成

このコードは(ATtiny4/5/9/10を除く)tinyAVR、megaAVR、XMEGAデバイスにメモリ領域を追加する方法を示します。この例はATxmega128A1用です。他のデバイスについてはそれに応じて変更されなければなりません。

```
#include <avr/io.h>

// ATxmega128A1用データ例
const char eepromdata[] __attribute__((section (".eeprom"))) = "Hello EEPROM";
// ヒューズ値の順番は下位から上位へ。ヒューズ バイト0に0xA2が書かれ、バイト1に0x00、以下同様
const uint8_t fusesdata[] __attribute__((section (".fuse"))) = {0xA2, 0x00, 0xFF, 0xFF, 0xFF, 0xFF};
const uint8_t lockbits[] __attribute__((section (".lockbits"))) = {0xFC};
const char userdata[] __attribute__((section (".user_signatures"))) = "Hello User Signatures";

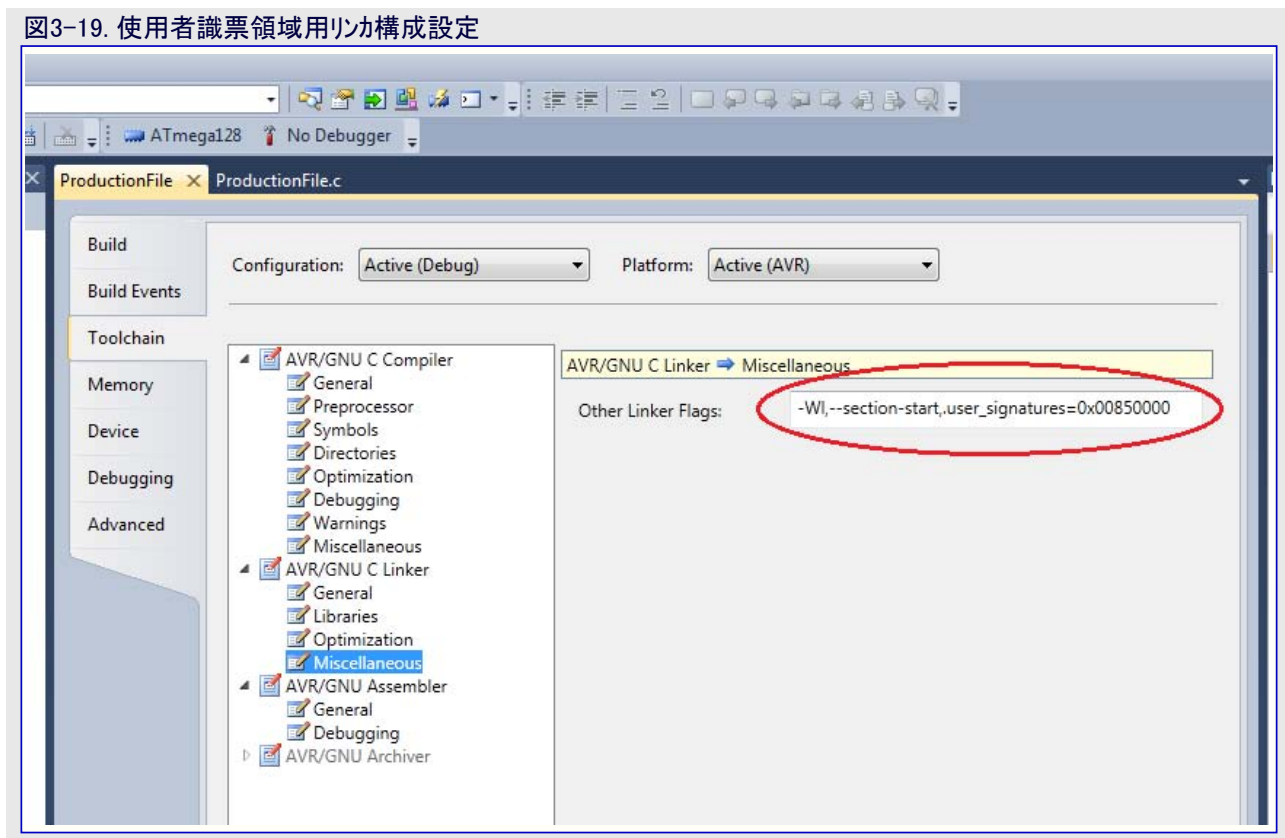
// 以下のツールチェーンプロジェクト任意選択を設定することを忘れないでください。
// AVR/GNU⇒Miscellaneous下に:
//
// -Wl,--section-start,.user_signatures=0x00850000

int main(void)
{
    while(1)
    {
        // すべきこと: あなたの応用コードを書いてください。
    }
}
```

使用者識票領域用リンク構成設定

使用者識票(User Signatures)領域は次に示されるように特有のリンク構成設定を持たなければなりません。これはELFファイル内で使用者識票領域に対する正しいアドレス変位(オフセット)を得ることが必要です。他のメモリ領域は自動的に正しいアドレスを得ます。

図3-19. 使用者識票領域用リカ構成設定



3.2.7.7.2. ATtiny4/5/9/10用ELFファイル作成

このコードはATtiny10に対してメモリ領域を追加する方法を示します。

```
#include <avr/io.h>

typedef struct _tagConfig
{
    unsigned char f1;
} Config;

typedef struct _tagLock
{
    unsigned char f1;
} Lock;

typedef struct _tagSig
{
    unsigned char f1;
    unsigned char f2;
    unsigned char f3;
} Signature;

Config __config __attribute__((section(".config"))) =
{
    f1 : 0xfb, // CKOUT設定
};

Lock __lock __attribute__((section(".lock"))) =
{
    f1 : 0xfc, // 更なる書き込みと検証を禁止
};

Signature __sig __attribute__((section(".signature"))) =
{
```

```

f1 : 0x03,
f2 : 0x90,
f3 : 0x1e,
};

int main(void)
{
    while(1)
    {
        // すべきこと:: あなたの応用コードを書いてください。
    }
}

```

3.2.7.7.3. UC3用ELFファイル作成

下の例はUC3デバイスでユーザーページにデータを追加する方法を示します。

```

#include <avr/io.h>

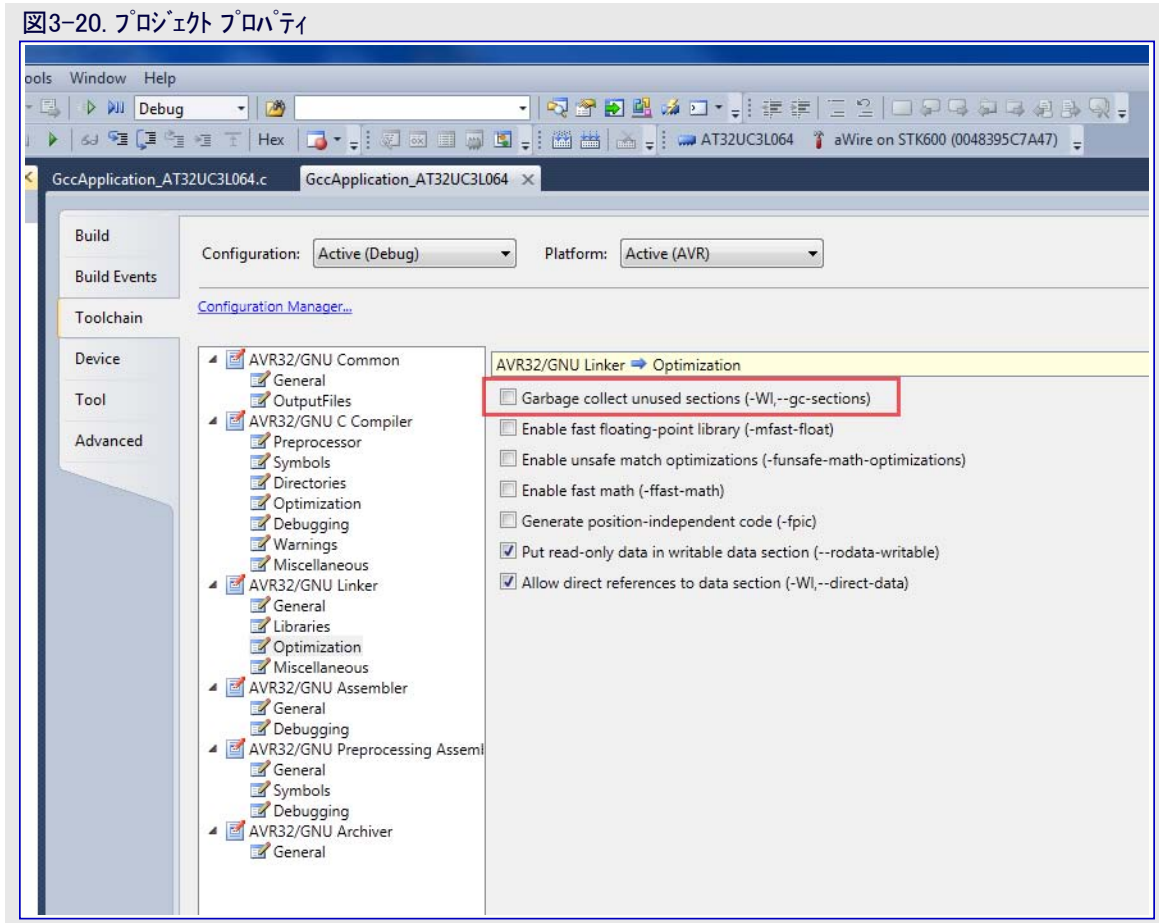
const char userdata[] __attribute__((section(".userpage"))) = "Hello Page";

int main(void)
{
    while(1)
    {
        // すべきこと:: あなたの応用コードを書いてください。
    }
}

```

3.2.7.7.4. プロジェクト プロパティ

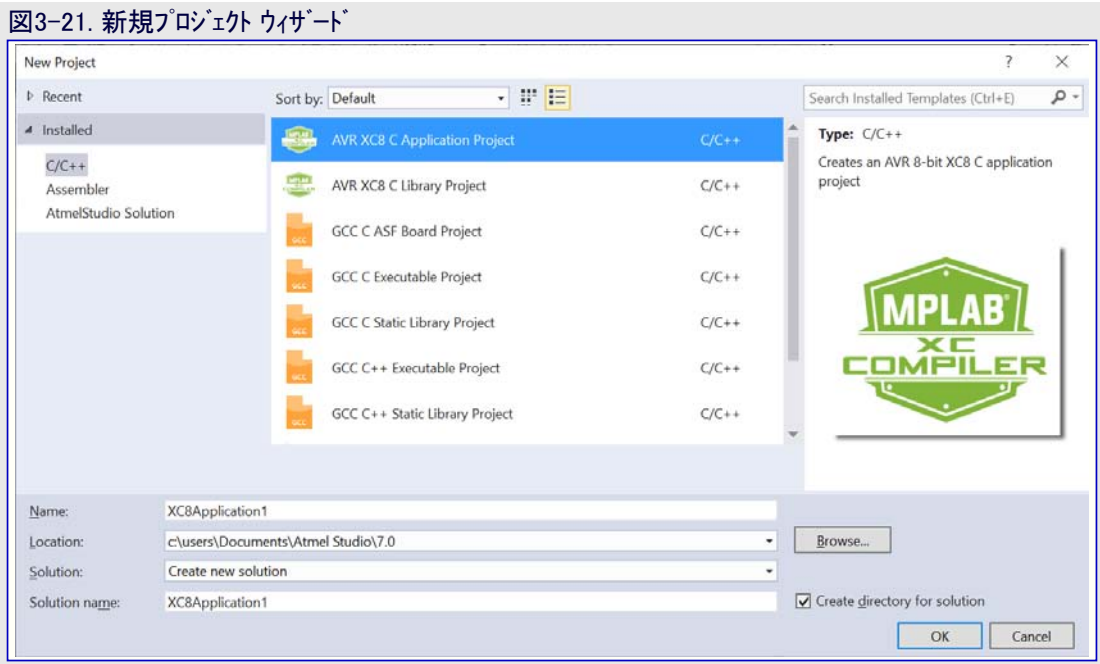
メモリ領域が定義されているが応用コードで参照されない場合、Project Properties(プロジェクト プロパティ)⇒Linker(リンカ)⇒Optimization(最適化)で”Garbage collect unused sections(未使用領域コミ回収)”任意選択が非チェックにされなければなりません。さもなければ、リンカはELFファイルにその領域を含めません。



3.3. XC8プロジェクト

3.3.1. AVR®デバイス用新規XC8応用プロジェクト開始

1. Microchip StudioのメニューからFile(ファイル)⇒New(新規)⇒Project...(プロジェクト)を選ぶことによってProject Wizard(プロジェクト ウィザード)を開いてください。

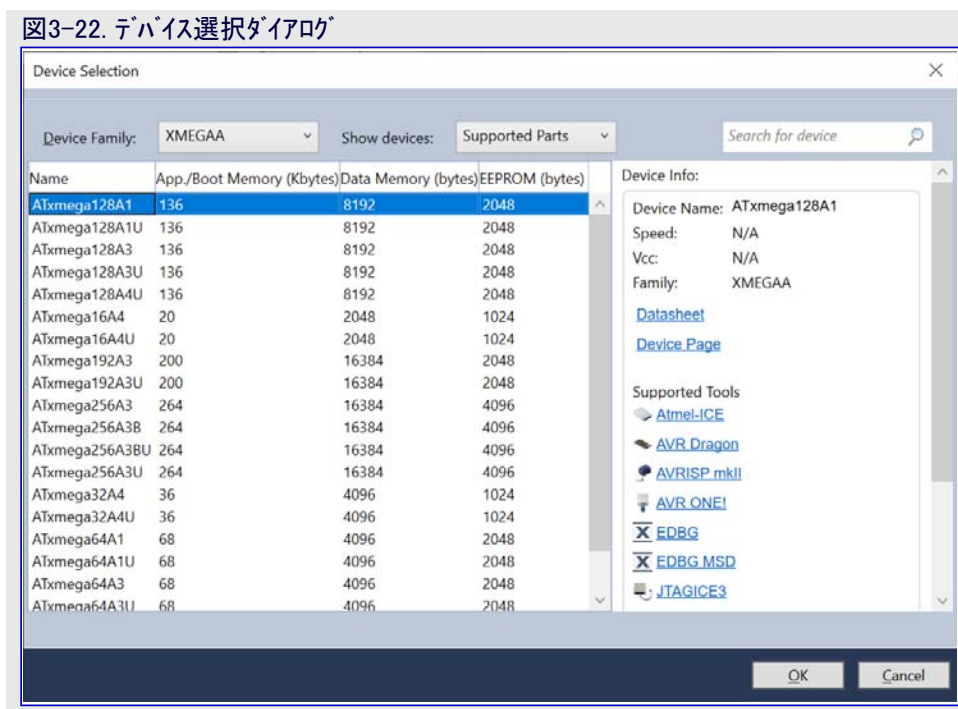


2. 雛形としてC/C++(C/C++)⇒AVR XC8 C Application Project(AVR XC8 C 応用プロジェクト)を選び、その後にName(プロジェクト名)を指定し、Location(場所)を選び、Solution name(解決策名)を提供してください。既定により、空のmain()関数を持つソースファイルがプロジェクトに追加されます。進めるためにOKを押してください。
3. ライブラリプロジェクトを作成するには雛形としてC/C++(C/C++)⇒XC8 C Library Project(XC8 C ライブラリプロジェクト)を選び、その後にName(プロジェクト名)を指定し、Location(場所)を選び、Solution name(解決策名)を提供してください。myfunc()関数を持つソースファイルがプロジェクトに追加されます。進めるためにOKを押してください。



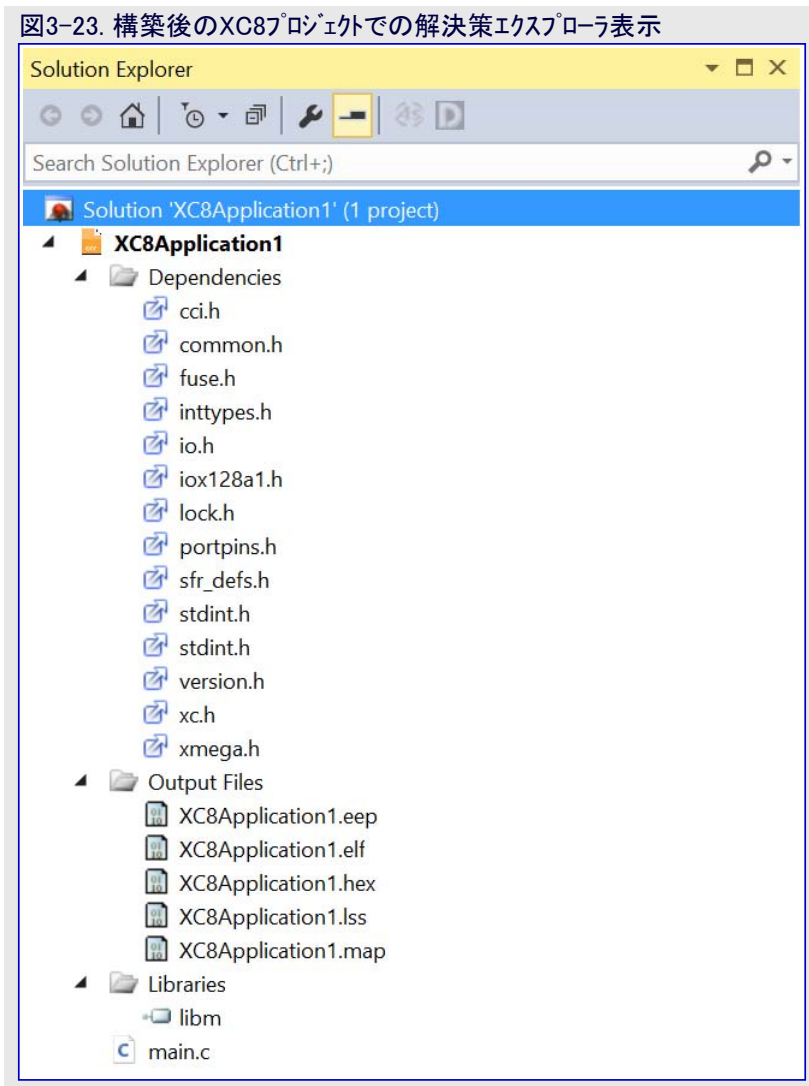
助言: XC8ライブラリプロジェクトについてもっと学ぶには「3.3.3. 新規XC8ライブラリプロジェクト開始」項をご覧ください。

4. デバイス選択ダイアログが現れます。あなたのプロジェクトに対して適切な目的対象デバイスを選んでください。



5. プロジェクト作成を完了するためにOKをクリックしてください。

6. プロジェクト項目で右クリックしてプロジェクトを構築するためにBuild(構築)を選んでください。



Dependencies (依存性)

構築処理で使われた全てのヘッダファイルがこの項目下で表示されます。エディタで開くにはそのファイル上をダブルクリックしてください。

Output Files (出力ファイル)

全ての出力ファイルがこの項目下で表示されます。

Libraries (ライブラリ)

全てのライブラリファイルがこの項目下で表示されます。



助言: ライブラリ任意選択についてもっと知るには「3.3.2. ライブラリ任意選択」項をご覧ください。

3.3.2. ライブラリ任意選択

3.3.2.1. プロジェクト ライブラリの追加方法



助言: 現在の解決策内にライブラリプロジェクトを持つことを確実にしてください。

応用プロジェクトにライブラリを追加するための手順:

1. “Add Library(ライブラリ追加)”ウィザードを呼び出すためにプロジェクトまたはプロジェクト内のLibraries(ライブラリ)節点で右クリックしてください。
2. Project Libraries(プロジェクト ライブラリ)タブを選んでください。現在の解決策内の全てのライブラリプロジェクトがここで一覧にされます。
3. 追加したいライブラリプロジェクトを選んでください。
4. 完了するためにOKをクリックしてください。

図3-24. ライブラリ追加状況メニュー

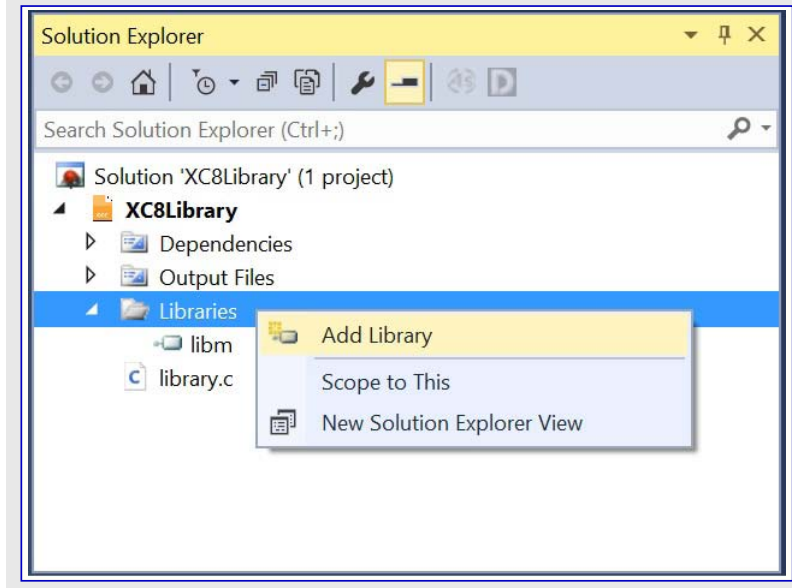
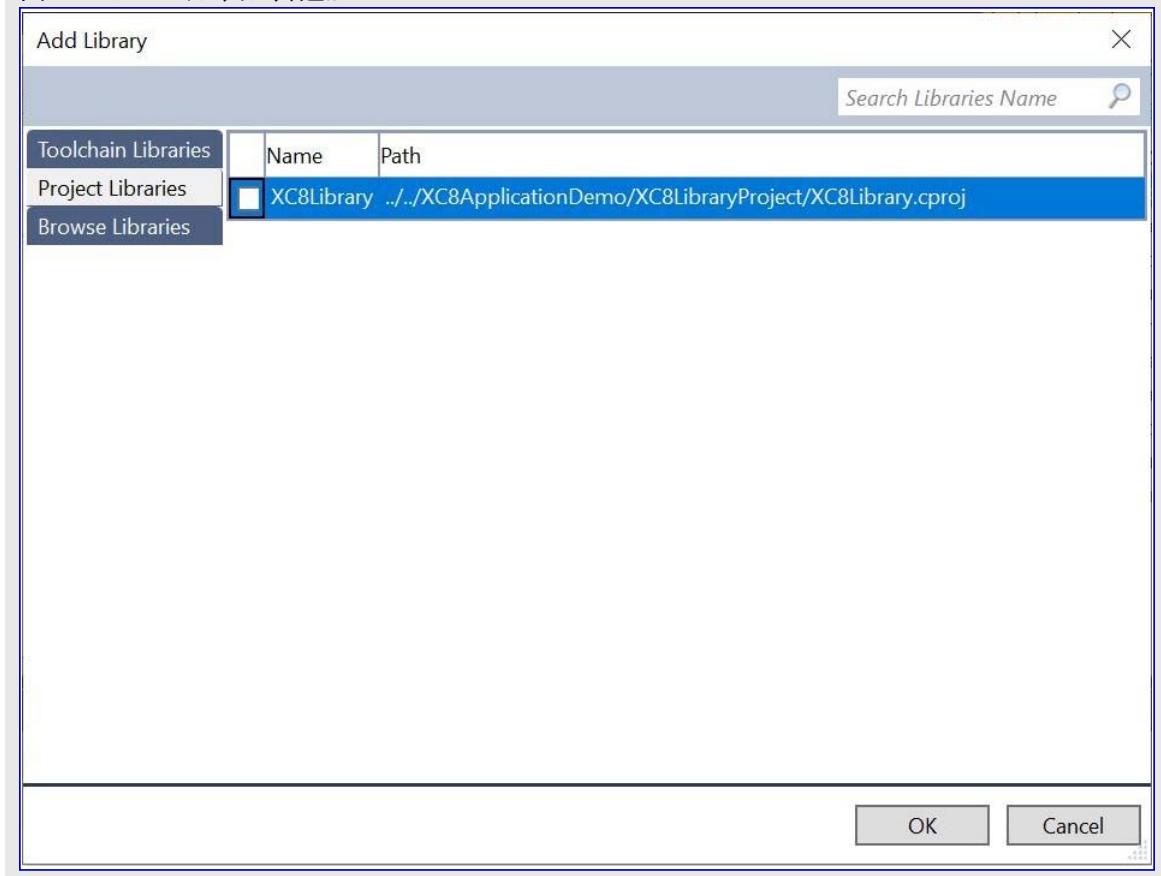
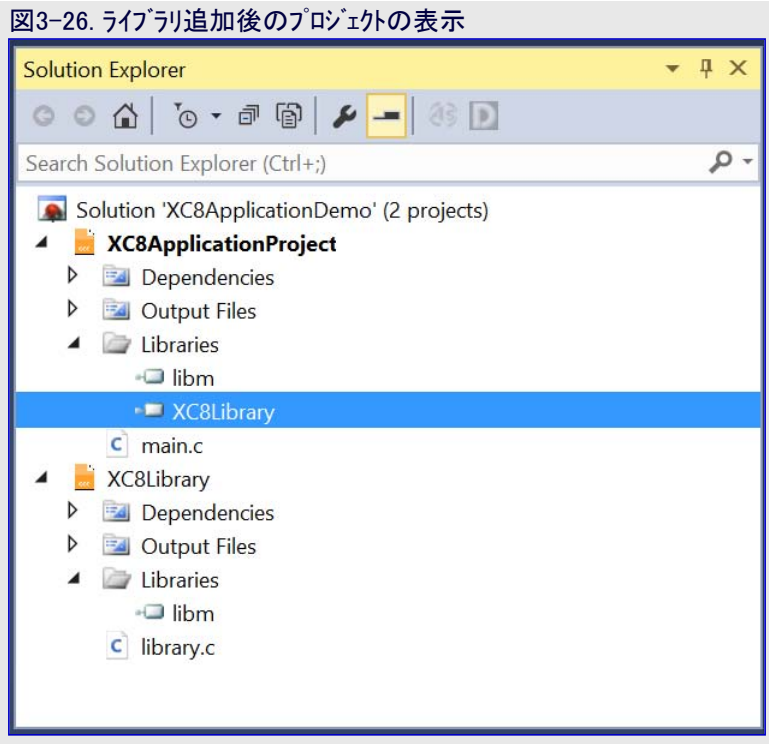
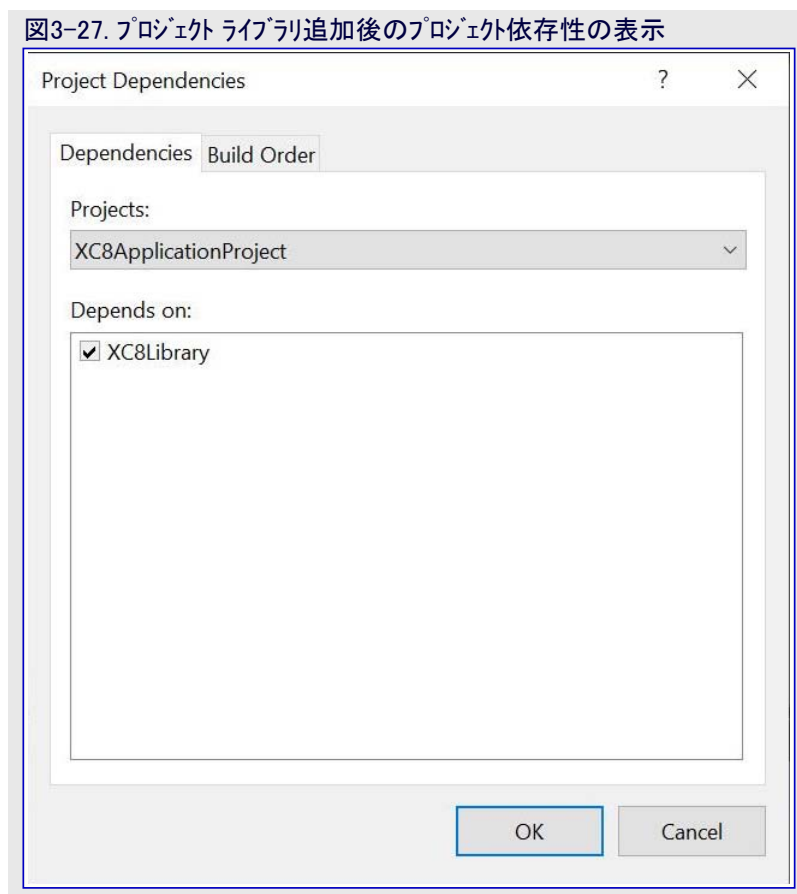


図3-25. プロジェクト ライブラリ追加





また、Project(プロジェクト)⇒Build Dependencies(構築依存性)⇒Project Dependencies(プロジェクト依存性)からライブラリがプロジェクト依存性として追加されたのを見てください。



3.3.2.2. ツールチェーン ライブラリの追加方法

1. “Add Library(ライブラリ追加)”ウィザードを呼び出すためにプロジェクトまたは‘Libraries(ライブラリ)’節点で右クリックしてください。
2. Toolchain Libraries(ツールチェーン ライブラリ)タブを選んでください。
3. 利用可能なツールチェーン ライブラリの一覧がここで一覧にされます。
4. 追加したいライブラリを選んでください。
5. 完了するためにOKをクリックしてください。

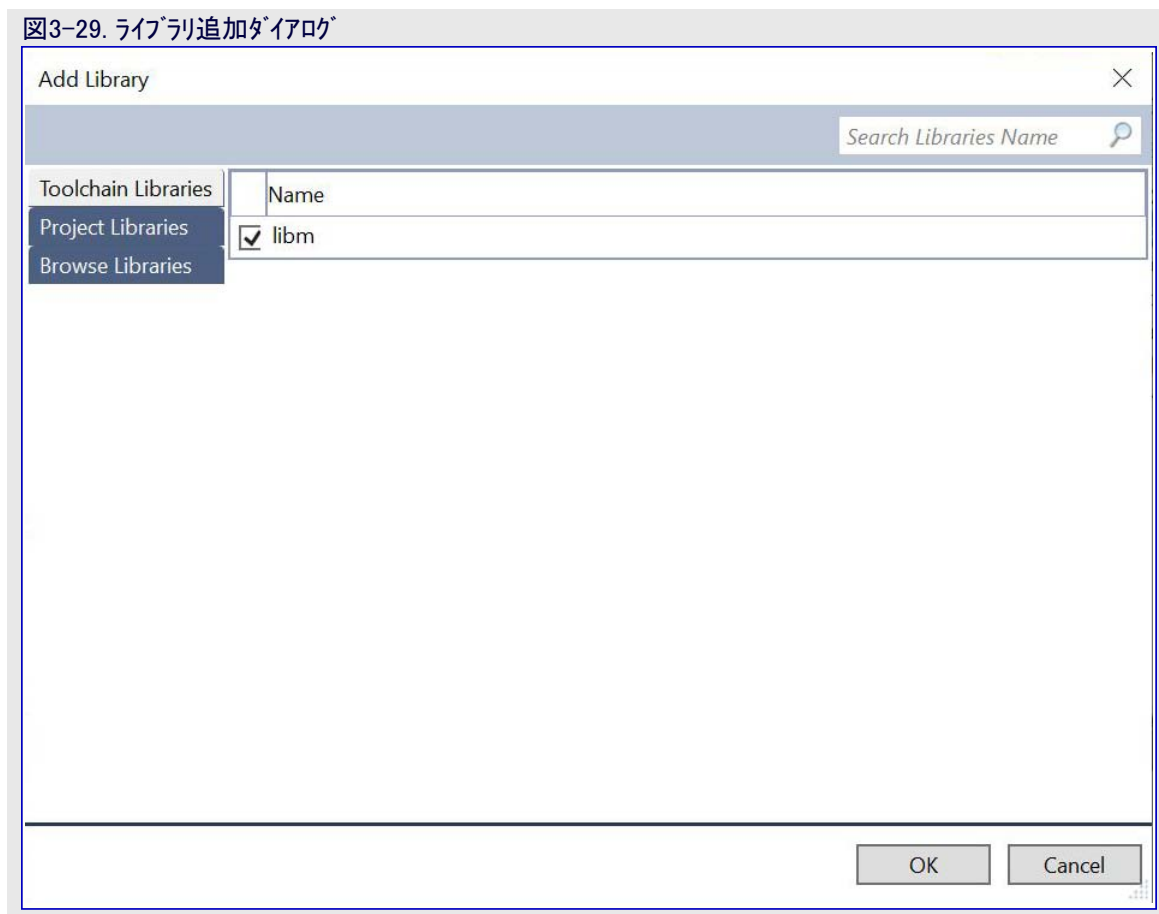
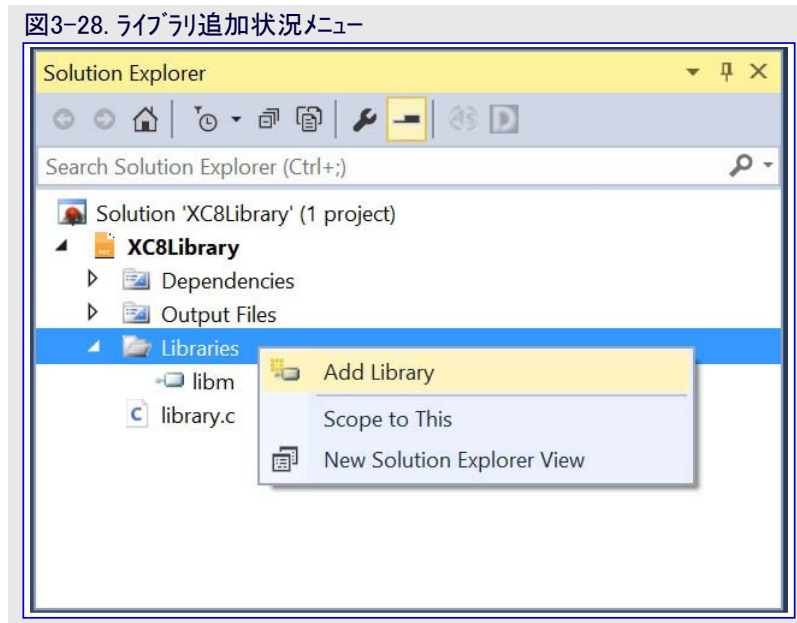
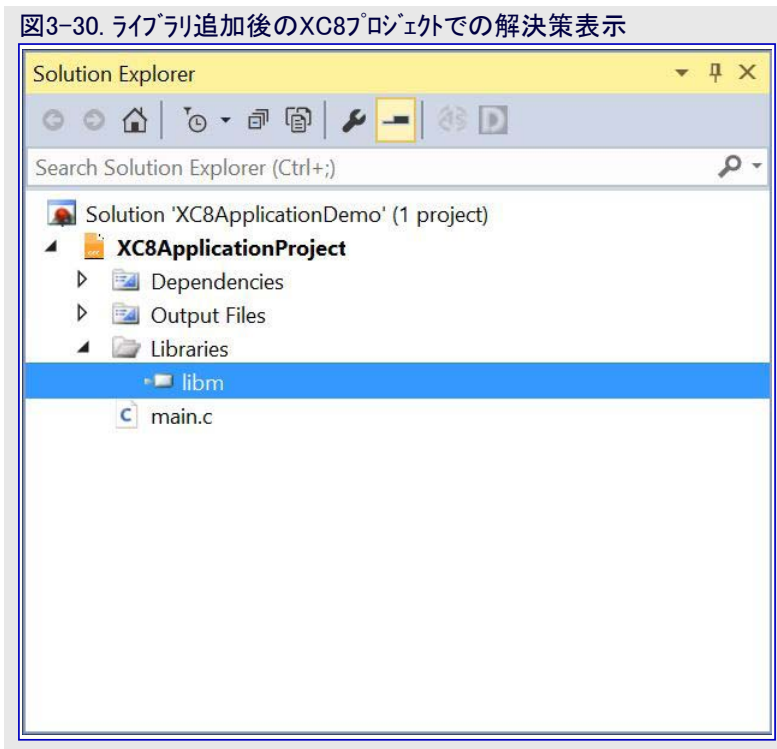
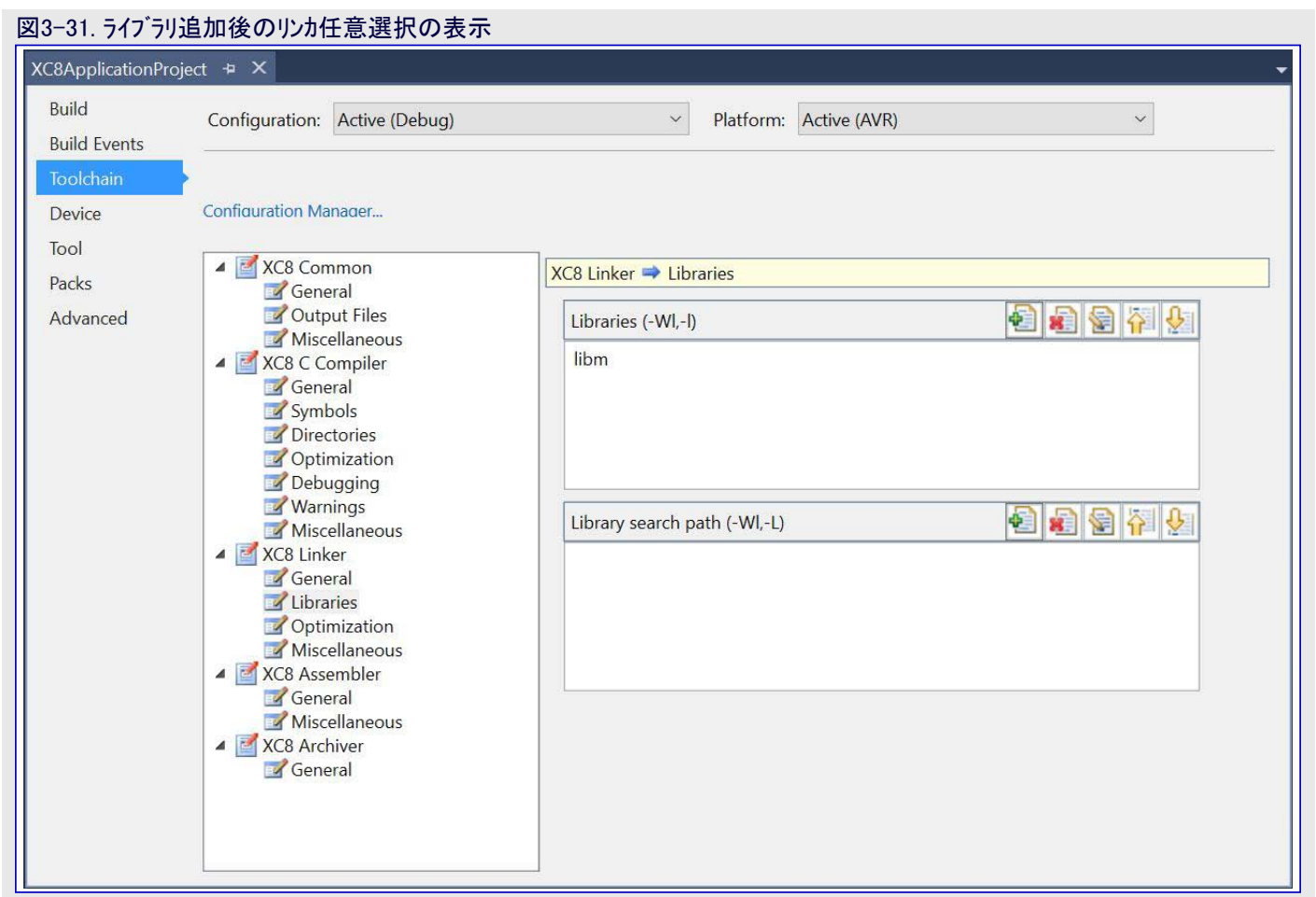


図3-30. ライブラリ追加後のXC8プロジェクトでの解決策表示



また、Project(プロジェクト)⇒Properties(特性)⇒Toolchain(ツールチェーン)タブからツールチェーン リンカ設定で追加された新しいライブラリを見ることができます。

図3-31. ライブラリ追加後のリンカ任意選択の表示



3.3.2.3. ライブラリの閲覧と追加の方法

1. “Add Library(ライブラリ追加)”ウィザードを呼び出すためにプロジェクトまたは’Libraries(ライブラリ)’節点で右クリックしてください。
2. ‘Browse Libraries(ライブラリ閲覧)’タブを選んでください。
3. ライブラリを閲覧して追加するために操作盤の’Browse(閲覧)’ 鈕をクリックしてください。
4. ライブラリ追加後、ライブラリ名、パスがここで一覧にされます。
5. 完了するためにOKをクリックしてください。

図3-32. ライブラリ追加状況メニュー

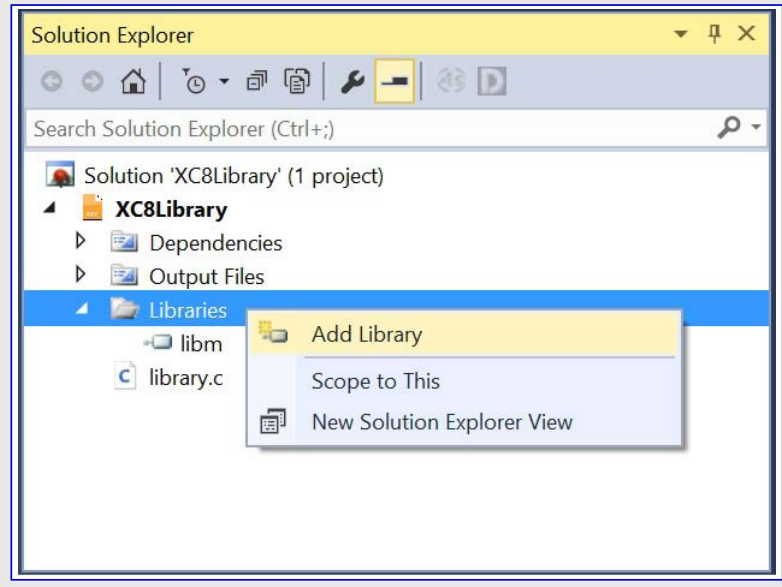
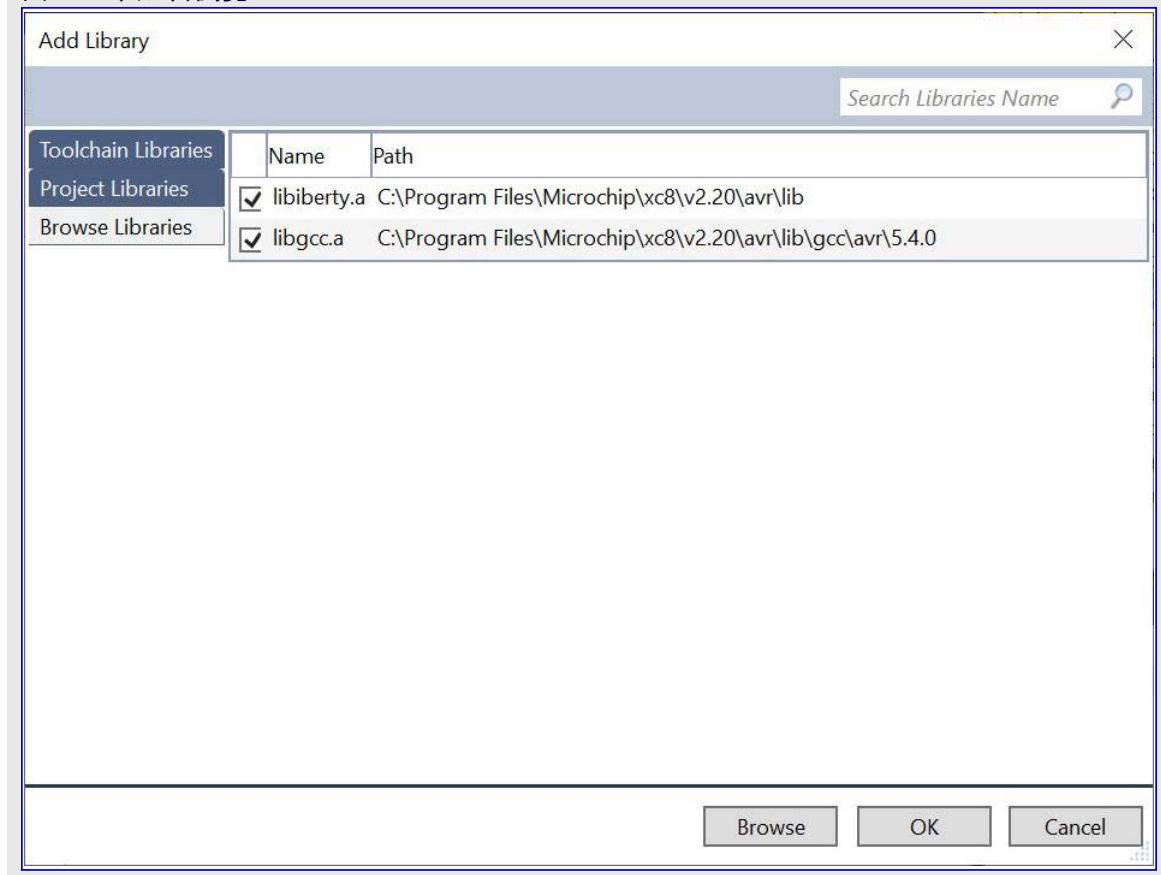


図3-33. ライブラリ閲覧



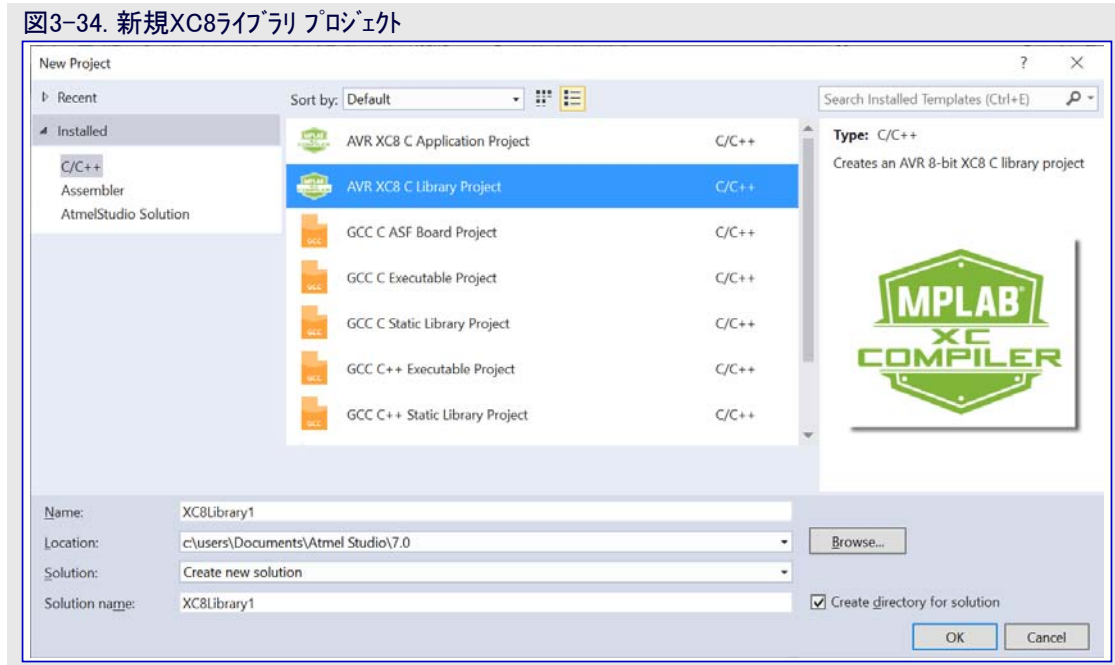
3.3.3. 新規XC8ライブラリプロジェクト開始

3.3.3.1. ライブラリの作成理由

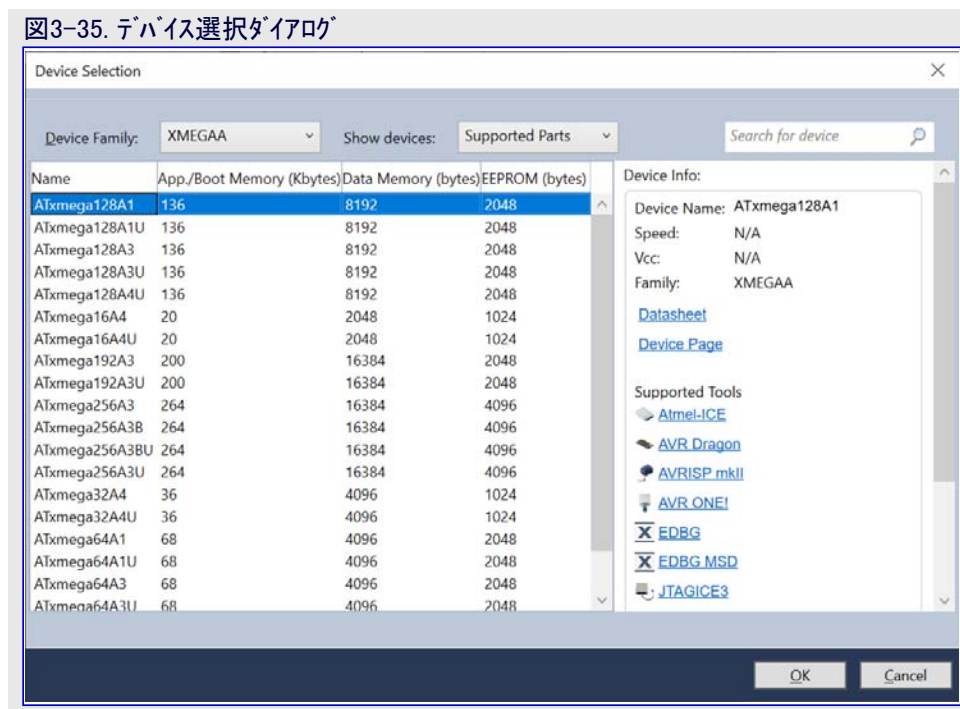
XC8ライブラリ(LIB)の作成はコードを再利用する良い方法です。全てのプログラムで同じルーチン/関数を再作成するよりも、それらを一度書き、機能を必要とするXC8応用プロジェクトからそれらを参照することができます。

3.3.3.2. 新規ライブラリプロジェクトの作成

1. Project(プロジェクト)メニューからNew Project(新規プロジェクト)を選ぶことによって新しいプロジェクトを作成してください。これはProject Wizard(プロジェクト ウィザード)を開きます。



2. 雛形としてC/C++(C/C++)⇒XC8 C Library Project(XC8 Cライブラリプロジェクト)を選び、その後にName(プロジェクト名)を指定し、Location(場所)を選び、Solution name(解決策名)を提供してください。myfunc()関数を持つソースファイルがプロジェクトに追加されます。進めるためにOKを押してください。
3. デバイス選択ダイアログが現れます。あなたのプロジェクトに対して適切な目的対象デバイスを選んでください。



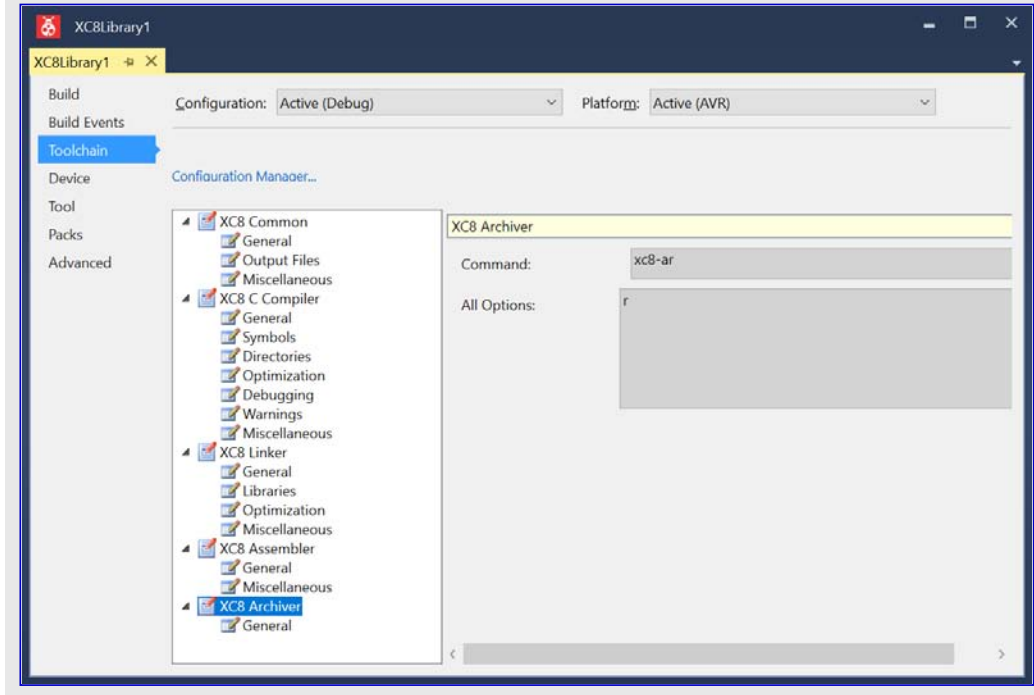
4. XC8ライブラリプロジェクトを作成するためにOKをクリックしてください。
5. プロジェクトを構築するためにプロジェクト節点で右クリックして'Build(構築)'を選んでください。

3.3.3.3. ライブラリプロジェクト任意選択 (XC8 Archiver)

XC8 archiver(纏め部)(xc8-ar)ははライブラリとしても知られる単一纏めファイル内にオブジェクトファイルの集合を結合します。

1. プロジェクト節点で右クリックして'Properties(特性)'を選んでください。
2. 'Toolchain(ツールチェーン)'タブをクリックし、その後にライブラリ任意選択を構成設定するためにXC8 Archiver(XC8纏め部)⇒General(全般)⇒Archiver Flags(纏め部フラグ)へ行ってください。
3. 構成設定した任意選択でプロジェクトを構築するためにプロジェクト節点で右クリックして'Build(構築)'を選んでください。

図3-36. XC8纏め部任意選択

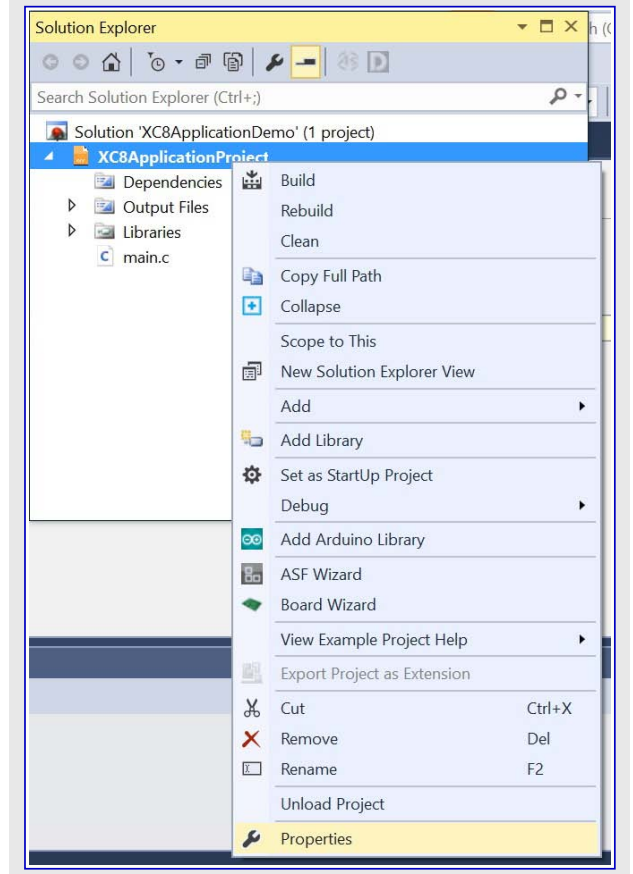


3.3.4. XC8プロジェクト任意選択と構成設定

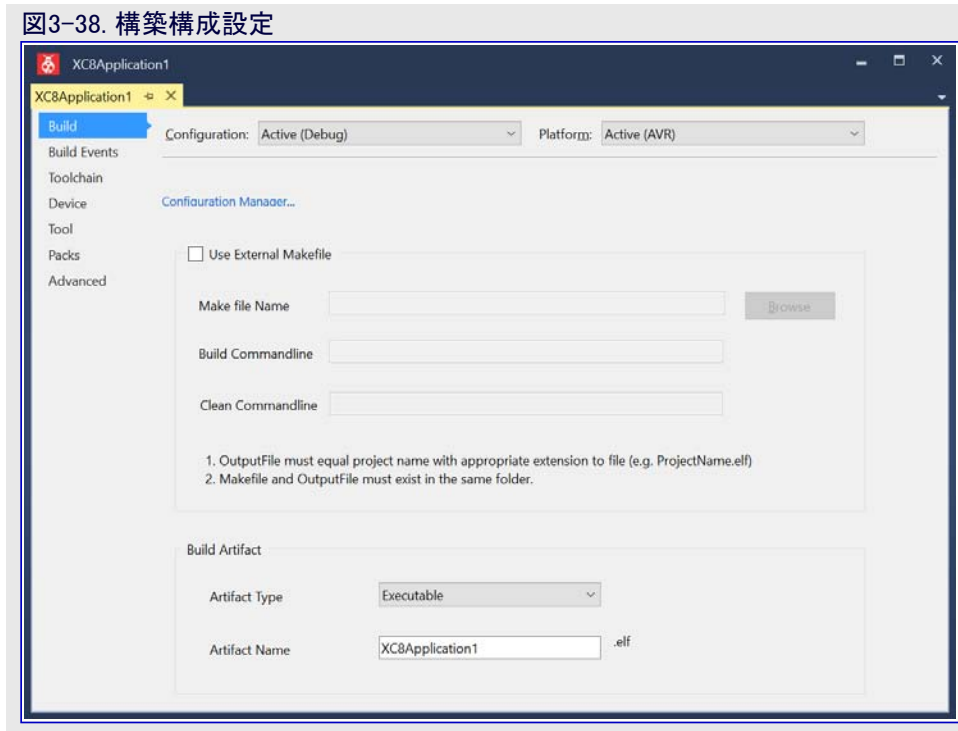
プロジェクト特性を開始するために解決策エクスプローラからプロジェクトを右クリックしてProperties(特性)を選んでください。

注: 変更を行う時に必ずプロジェクトファイルでの変更を更新するのに、File(ファイル)メニューまたはツールバーから'Save All(全て保存) (Ctrl+Shift+S)'を使ってください。

図3-37. プロジェクト特性状況メニュー



3.3.4.1. 構築任意選択



Build(構築)タブで、あなたのプロジェクトに対して外部メークファイルの使用を望むかどうかを構成設定することができます。その場合、単にUse External Makefile(外部メークファイル使用)チェック枠をチェックしてメークファイルの正しいパスを選ぶためにBrowse(検索)してください。

Build Commandline(構築コマンド行)はプロジェクトに対して構築が呼び出される時の外部メークファイルのために提供されます。既定構築目的の対象は”all(全て)”です。

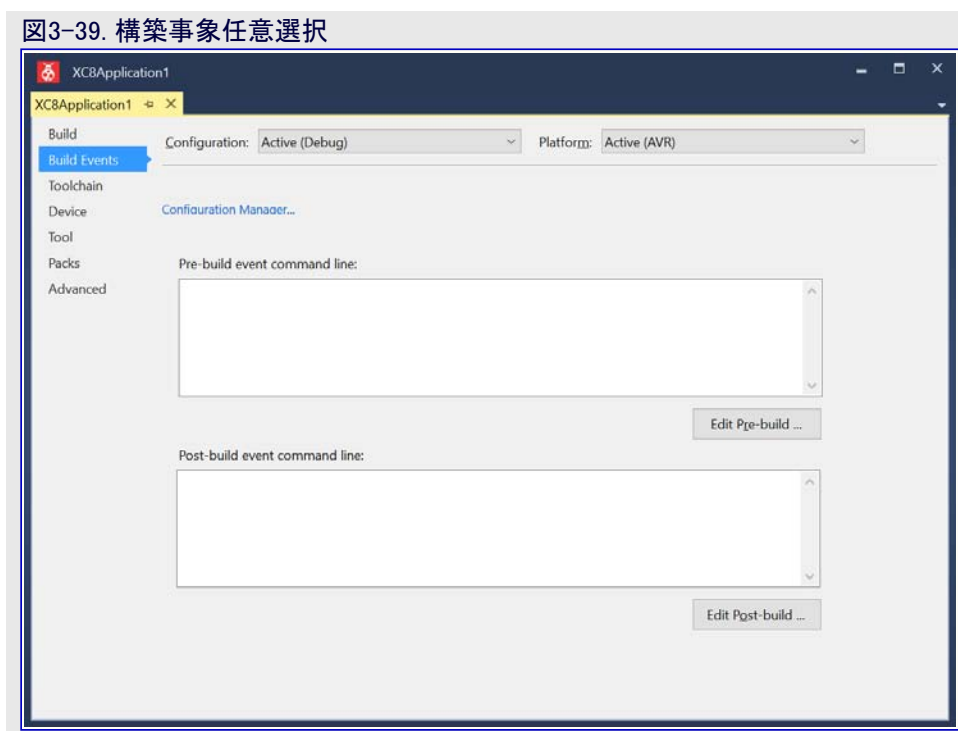
Clean Commandline(解消コマンド行)はプロジェクトに対して解消が呼び出される時の外部メークファイルのために提供されます。既定書き出しの対象は”clean(解消)”です。

外部メークファイル構成設定の他にも、構築に対して応用の形式を指定することもできます。この任意選択はArtifact Type(成果物の種類)コンボ枠を用いて選ぶことができるExecutable(実行可能)またはStatic Library(静的ライブラリ)です。

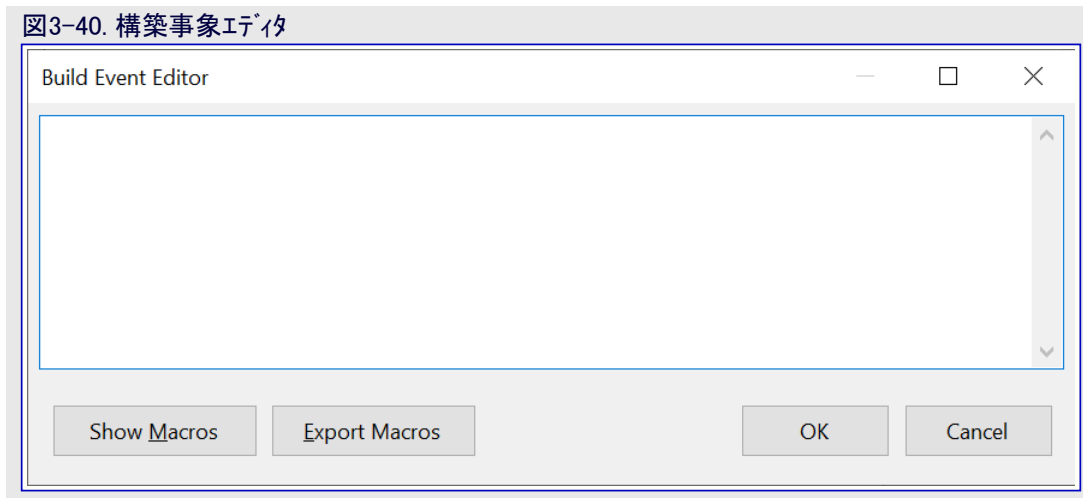
注: 独自のメークファイルは以下のようなこれらの条件を満たさなければなりません。

1. 目的対象名はプロジェクト名と同じでなければなりません。
2. メークファイルと目的対象は(NTFSリンクでも参照することができる)同じフォルダに存在しなければなりません。

3.3.4.2. 構築事象



Build Events(構築事象)タブはPre-build(構築前)とPost-build(構築後)のスク립トによって開始される各構成設定に対して計画された事象の一覧を含みます。これらの事象はEdit Pre-build...(構築前編集)またはEdit Post-build...(構築後編集)の釦のどちらかをクリックすることによって追加、削除、または編集することができます。これらの釦をクリックすることで、以下のダイアログであなたの命令を手動で追加すべきです。現公開版の時点で、他の利用可能な応用とのリンクとしてそれら内で宣言された環境変数と値を使うことが可能です。その機能を使うにはShow Macros(マクロ表示)釦を押してください。



マクロ

コメント行編集枠で挿入するのにマクロ/環境変数の一覧を表示するために編集枠を広げてください。

マクロ表

利用可能なマクロ/環境変数とその値を一覧してください。コメント行編集枠内へ挿入するのに一度に1つのマクロだけを選ぶことができます。MSBuildはプロジェクトファイルとMSBuildハインリについての情報を格納する予約されたプロパティの組を提供します。これらのプロパティは編集枠で一覧にされるかもしれません。

Microchip Studioに特有な説明については下のマクロ/環境変数をご覧ください。

表3-6. Microchip Studio構築マクロ表

マクロ	説明
\$(AVRSTUDIO_EXE_PATH)	(ドライブとパスで定義される)Microchip Studioのインストール ディレクトリ
\$(SolutionDir)	(ドライブとパスで定義される)解決策のディレクトリ
\$(SolutionPath)	(ドライブ、パス、基本名、ファイル拡張子で定義される)解決策の絶対パス名
\$(SolutionFileName)	解決策の名前
\$(SolutionName)	解決策の基本名
\$(SolutionExt)	解決策のファイル拡張子。ファイル拡張子の前の'.'を含みます。
\$(Configuration)	現在のプロジェクト構成設定の名前。例えば、"Debug"
\$(Platform)	現在の目的対象基盤の名前。例えば、"AVR"
\$(DevEnvDir)	(ドライブとパスで定義される)Microchip Studioのインストール ディレクトリ
\$(ProjectVersion)	プロジェクトの版番号
\$(ProjectGuid)	プロジェクトの固有識別子
\$(avrdevice)	現在選ばれているデバイスの名前
\$(avrdeviceseries)	選ばれているデバイスの系統。Microchip Studioによって内部的に使用
\$(OutputType)	現在のプロジェクトが実行可能型または静的ライブラリ型かを定義
\$(Language)	現在のプロジェクトの言語。例えば、C、CPP、またはアセンブラ
\$(OutputFileName)	(基本ファイル名として定義される)構築用基本出力ファイルの名前
\$(OutputFileExtension)	構築用基本出力ファイルのファイル拡張子。ファイル拡張子の前の'.'を含みます。
\$(OutputDirectory)	出力ファイル ディレクトリの絶対パス
\$(AssemblyName)	構築用基本出力のアセンブリ名
\$(Name)	プロジェクトの基本名
\$(RootNamespace)	プロジェクトの基本名
\$(ToolchainName)	ツールチェーンの名前
\$(ToolchainFlavour)	ツールチェーン コンパイラの名前

3.3.4.3. コンパイラとツールチェーンの任意選択

図3-41. コンパイラとツールチェーンの任意選択

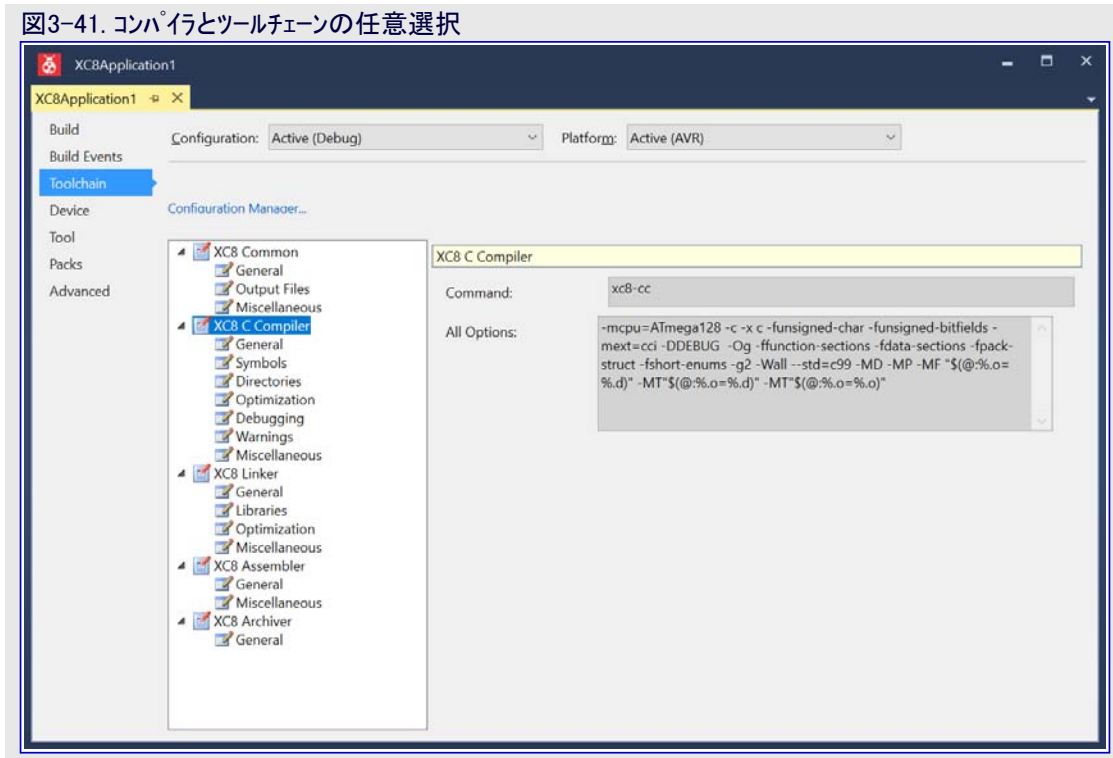


表3-7. XC8 コンパイラ任意選択

任意選択	説明
全般任意選択	
-mcall-prologues	プロローグとエピローグの機能にサブルーチンを使用
-mno-interrupts	割り込み禁止なしでスタックポインタを変更
-funsigned-char	既定char型は符号なし
-funsigned-bitfield	既定ビット領域は符号なし
シンボル任意選択	
ソース内のシンボルを定義(-D)または定義解除(-U)することができます。新しいシンボル宣言は下のインターフェース鉤を用いて追加、変更、再整理をすることができます。	
<ul style="list-style-type: none"> • 新しいシンボル追加。これと以下の全アイコンはMicrochip Studioインターフェースの他の部分で同じ意味で再利用されます。 • シンボル削除 • シンボル編集 • 解析順に於いてシンボルを上に移動 • 解析順に於いてシンボルを下に移動 	
インクルード デイレクトリ	
インクルードされるヘッダと定義指令の	全てを含み、シンボルと同じインターフェースを用いて編集することができます。
最適化任意選択	
最適化レベル(引き落としメニュー) -O0, -O1, -O2, -O3, -Os, -Og	最適化なし、速度最適化(レベル1~3)、大きさ最適化、良好なデバッグ体験用最適化
-ffunction-sections	ゴミ回収用関数準備。関数が決して使われないなら、メモリがかき集められます。
-fpack-struct	構造体メンバを共に一括化
-fshort-enums	列挙型によって必要とされるのと同数のバイトだけを割り当て
デバッグ任意選択	
デバッグレベル(引き落としメニュー) none, -g1, -g2, -g3	ソースコードで残されるまたは挿入されるコードとヘッダを追跡してデバッグするレベルを指定

次頁へ続く

表3-7 (続き). XC8 コンパイル任意選択

任意選択	説明
警告メッセージ出力任意選択	
-Wall	全ての警告を表示
-Werror	疑わしい構成に対する警告の代わりに異常を生成
-pedantic	厳密なANSI Cによって要求された警告、禁止拡張を使う全てのプログラムを却下
-w	全ての警告メッセージを抑制
その他の任意選択	
他のフラグ(書式領域)	他のプロジェクト特有フラグを入力
-v	詳細

3.3.4.3.1. XC8リンク任意選択

表3-8. XC8リンク任意選択

任意選択	説明
-nodefaultlibs	標準Cライブラリをリンクしない。
-Wl, -Map	マップ ファイルを生成
-Wl, -u, vfprintf	vfprintfライブラリを使用
ライブラリ任意選択	
ライブラリ -Wl, -l (書式領域)	 の鉤を用いて、ここでライブラリ名の追加、優先付け、編集ができます。
ライブラリ検索パス -Wl, -L (書式領域)	上と同じインターフェースでリンクが動的にリンクされるライブラリを探す所のパスの追加、優先付け、編集ができます。
最適化任意選択	
-Wl, -gc-sections	未使用領域コミ回収
-mrelax	分岐緩和
その他の任意選択	
他のリンクフラグ (書式領域)	他のプロジェクト特有フラグを入力

3.3.4.3.2. XC8アセンブラ任意選択

表3-9. XC8アセンブラ任意選択

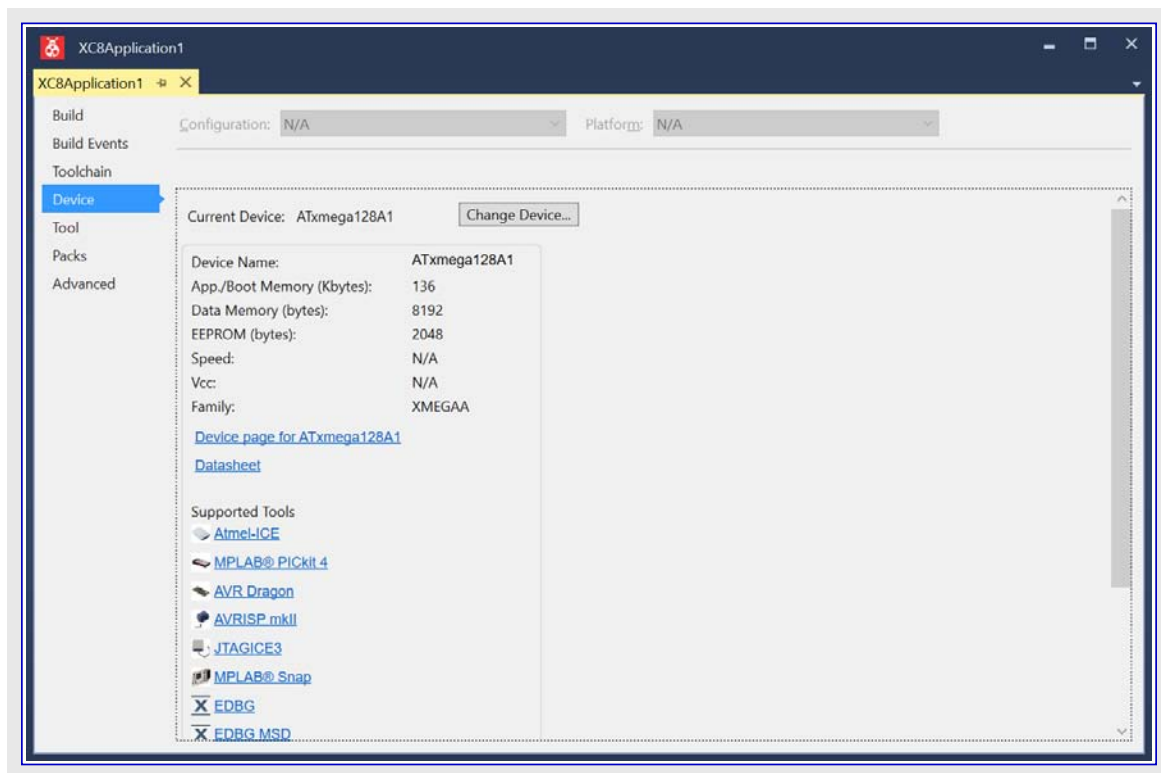
任意選択	説明
最適化任意選択	
-Wa, 任意選択	アセンブラに縛られる任意選択
-x assembler-with-cpp	-x assembler-with-cpp言語任意選択はアセンブリ言語ソース ファイルがアセンブルされる前に前処理されることを保証します。
-c	-c任意選択はアセンブラ実行後にコンパイルを停止するのに使われます。
-v	アセンブラ出力で版番号を公表
デバッグ任意選択	
-Wa, -g	生成するデバッグ情報の形式
その他の任意選択	
他のアセンブラフラグ	アセンブラ段階で内包/追加することを使用者が望むかもしれない他のプロジェクト特有アセンブラフラグを入力

3.3.4.4. デバイス任意選択

このタブは現在のプロジェクトに対するデバイスを選んで変更することを許し、デバイス選択部と同じです。図3-22をご覧ください。

 **助言:** 編集集中に素早くこのタブを得るにはDevice and Debugger(デバイスとデバッガ)ツールバーでDevice鉤をクリックしてください。





3.3.4.5. ツール任意選択

このタブは現在のプロジェクトに対してデバッグ基盤を選んで変更することを許します。

図3-42. ツール任意選択



1. 引き落とし一覧からツール/デバッグを選んでください。現在の選択 Simulator (シミュレータ) が示されます。
2. 引き落とし一覧からインターフェースを選んでください。更なる特性は選んだツールとインターフェースに依存します。

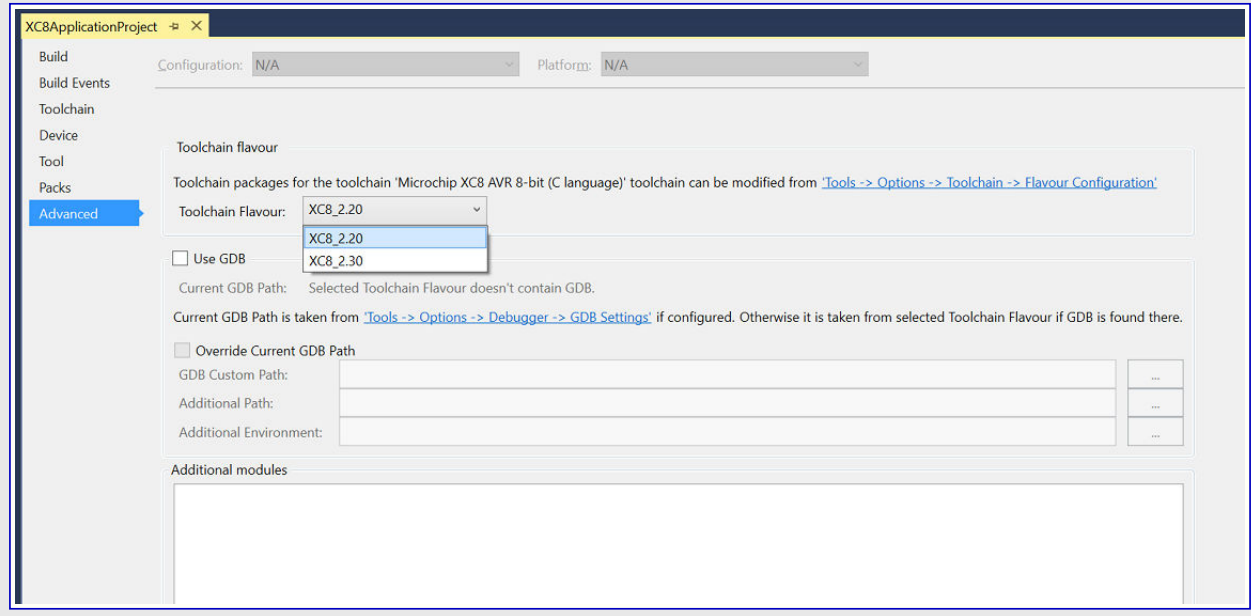
3.3.4.6. 高度な任意選択

Toolchain Flavour(ツールチェーン味付け)はXC8プロジェクトを構築するのに使われるツールチェーンのパスを指定します。利用可能なツールチェーン味付けは引き落とし選択枠で一覧にされます。



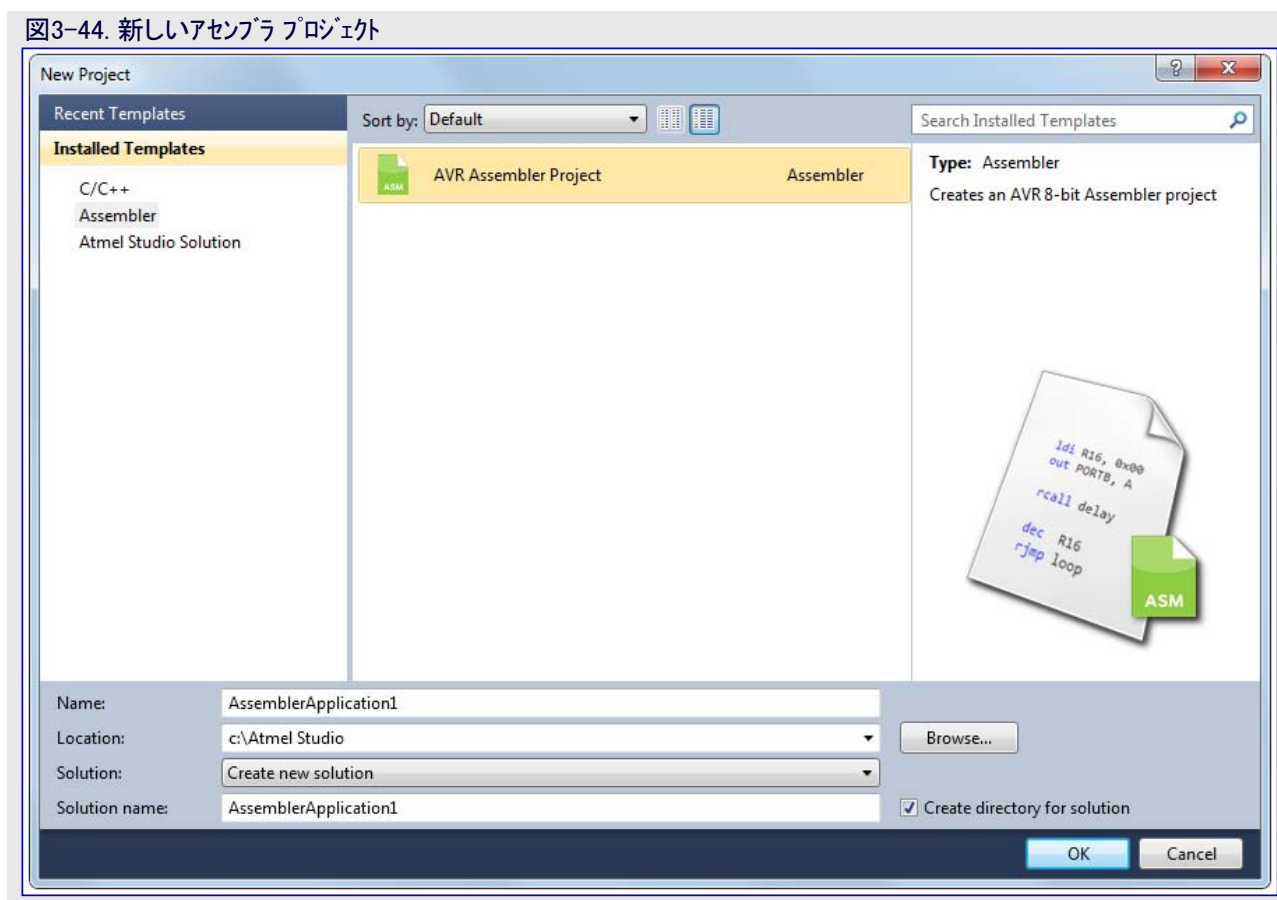
助言: ツールチェーン味付けの追加方法を学ぶには「[10.3.11. ツールチェーン](#)」を参照してください。

図3-43. 高度な任意選択



3.4. アセンブラ プロジェクト

3.4.1. 新規アセンブラ プロジェクト作成



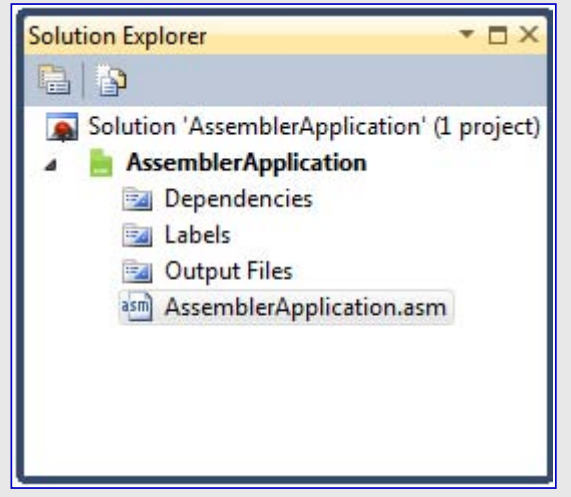
OK押下後、マイクロ コントローラを選んでください。

下で与えられる簡単なLED追っかけコードを値用してアセンブラ構築とコード デバッグを試してみることができます。あなたのハードウェアにポートを(この場合はEに)変更することによってどのAVR 8ビット マイクロ コントローラにも合うでしょう。

```
start: NOP
        LDI    R16, 0xFF
        STS    PORTE_DIR, R16
        LDI    R17, 0x80
output: STS    PORTE_OUT, R17
        ROL    R17
        LDI    R16, 0x00
delay:  LDI    R18, 0x00
delay1: INC    R18
        BRNE  delay1
;
        INC    R16
        BRNE  delay
;
        BREAK
        RJMP  output
```

新しいプロジェクトが作成されるか、または古いプロジェクトが読み込まれると、プロジェクト表示は全てのプロジェクトファイルと共に表示されます。Solution explorer(解決策エクスプローラ)ウィンドウで状況メニューを用いてプロジェクト一覧に対してファイルを追加、作成、削除をすることができます。

図3-45. アセンブラプロジェクトの表示



全てのソースファイルは一覧の最後で一覧にされます。エディタで開くにはどれかのファイル上でダブルクリックしてください。

全ての独自インクルードファイルは、プロジェクトで新しいフォルダを作成しない限り、プロジェクト名項目下のディレクトリで一覧にされます。

Dependencies(依存性) : 全てのインクルードファイルがここで一覧にされます。エディタで開くにはどれかのファイル上でダブルクリックしてください。

Labels(ラベル) : アセンブリプログラム内の全てのラベルがここで一覧にされます。ソース内のその位置を示すにはその項目上でダブルクリックしてください。印が正しい行を指示します。

Output Files(出力ファイル) : 全ての出力ファイルがこの項目下に表示されます。

図3-46. 構築完了後のアセンブラプロジェクトの表示

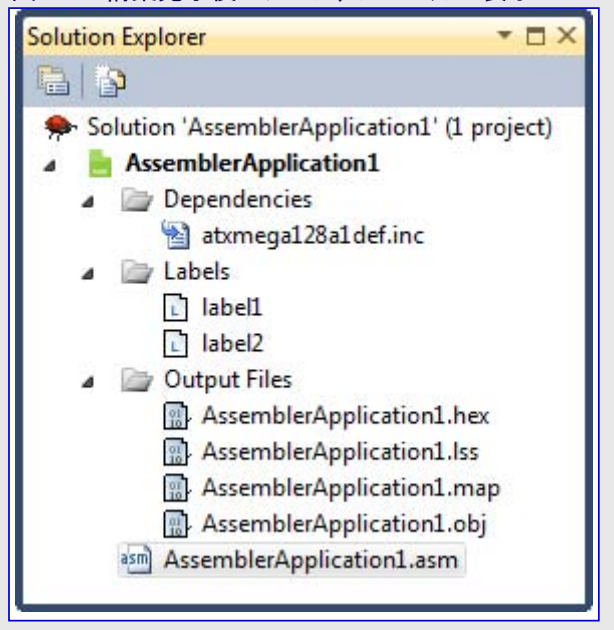


図3-47. ファイル状況メニュー

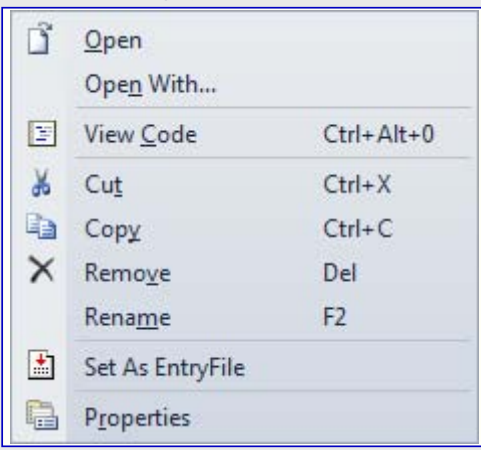


表3-10. ファイル状況メニュー

メニュー文	ショートカット	説明
Open	右クリック⇒O	選択したファイルを開く
Open With...	右クリック⇒n	選択したファイルを別のエディタかツールで開く
Cut	Ctrl X	現在の区分からファイルを切り取り
Copy	Ctrl C	現在の区分からファイルを複製
Remove	Del	プロジェクトから選択したファイルを削除
Rename	F2	選択したファイルを改名
Set As EntryFile		選択したファイルを入力ファイルとして設定
Properties	Alt Enter	現在のファイルのプロパティ

全てのインターフェース表示部は既定によって結合されます。ウィンドウを望む位置周辺にドラッグする、またはMicrochip Studioの迅速結合メニューでウィンドウをドラッグ&ドロップすることによって表示部を結合と切り離しをすることができます。迅速結合メニューはインターフェース表示部またはウィンドウのドラッグを開始する毎に現れます。

3.4.1.1. プロジェクト状況メニュー

様々な構築命令はメニューとツールバーから利用可能です。プロジェクト用の状況メニューもあります。

図3-48. プロジェクト状況メニュー

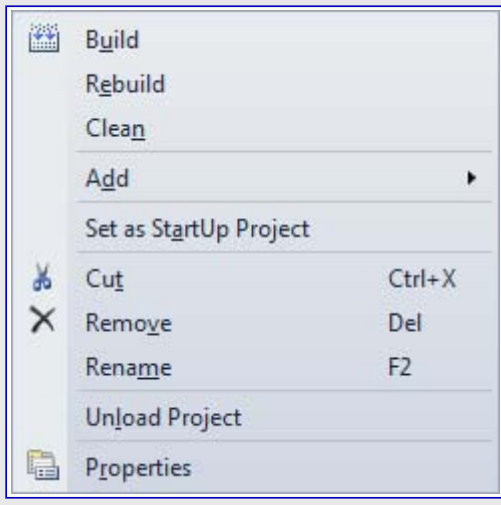


表3-11. プロジェクト状況メニュー

メニュー文	ショートカット	説明
Build	右クリック⇒u	選択したプロジェクトを構築
Rebuild	右クリック⇒e	プロジェクトを解消して構築
Clean	右クリック⇒n	成果物を解消して消去
Add	Ctrl Shift A(新規) Shift Alt A(既存)	プロジェクトに新規ファイルまたは既存ファイルを追加
Set as StartUp Project	右クリック⇒a	始動で現在のプロジェクトを自動的に開くように構成設定
Cut	Ctrl X	別の解決策に対する補助プロジェクトとして貼り付けるためにプロジェクト切り取り
Remove	Del	カーソル下のプロジェクトまたは補助プロジェクトを削除
Rename	F2	現在のプロジェクトを改名
Unload Project	右クリック⇒l	IDEから有効なプロジェクトを取り除く
Properties	Alt Enter	プロジェクトプロパティ

3.4.2. アセンブラ任意選択

メニューのProject(プロジェクト)⇒あなたのプロジェクト名 Properties(プロパティ)で任意選択ウィンドウを開いてください。このメニュー項目はアセンブラプロジェクトが開いている時にだけ利用可能です。Project properties(プロジェクトプロパティ)ウィンドウが開いた後、アセンブラ任意選択を構成設定するために6つのタブを見るでしょう。Toolchain(ツールチェーン)をクリックしてください。

図3-49. アセンブラ構成設定ダイアログ、コマンド行表示

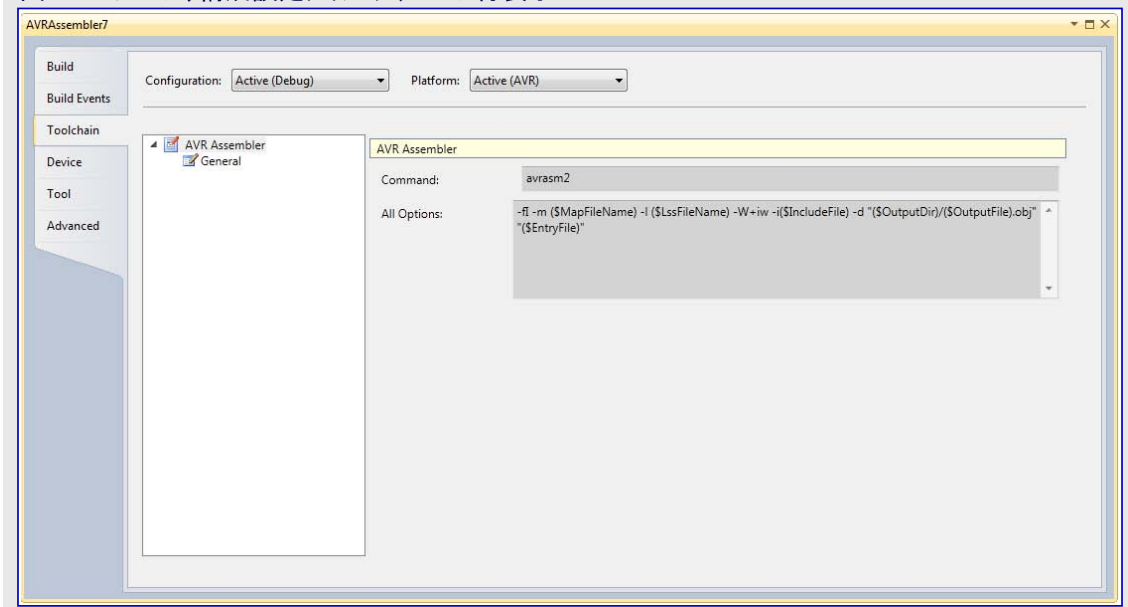
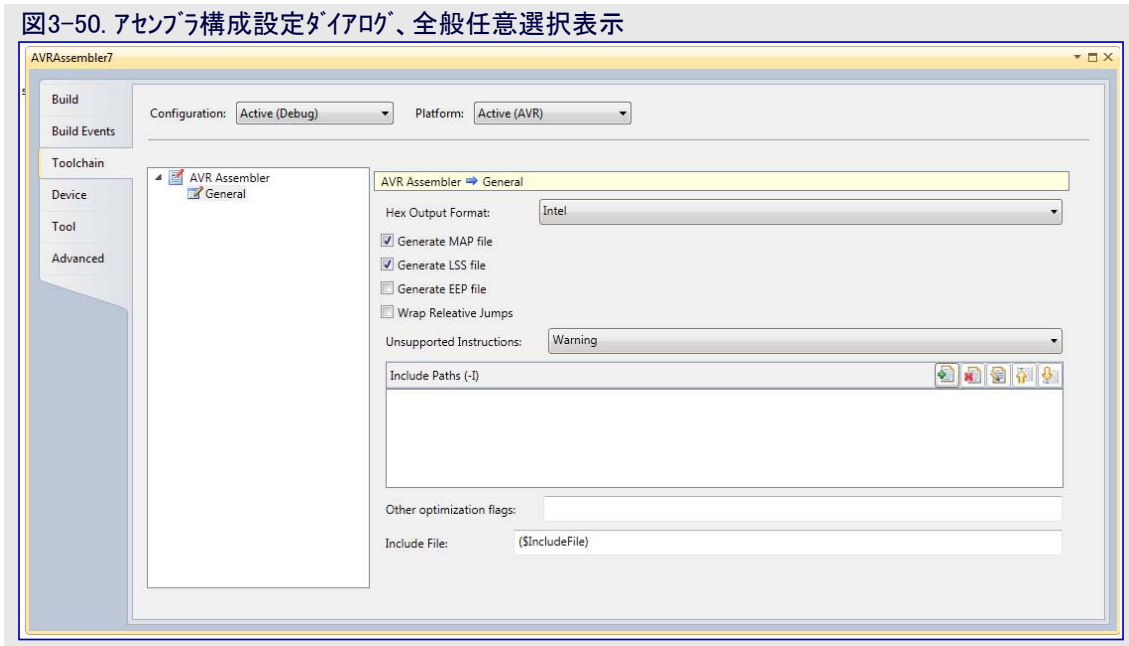


図3-50. アセンブラ構成設定ダイアログ、全般任意選択表示



3.4.2.1. 様々な設定の説明

Configuration(構成設定)メニューはプロジェクトプロパティへの変更によって影響を及ぼされるプロジェクト完成の過程を選ぶことを許します。既定によって**Debug**(デバッグ)が初期過程と初期活性構成設定です。以下の任意選択が利用可能です。

1. **Debug**(デバッグ)
2. **Release**(公開)
3. **All configurations**(全構成設定)

Platform(基盤)メニューは試作に利用可能な目的対象適合基盤を示します。

Hex Output Format(HEX出力形式)：出力形式として以下のファイル形式を選ぶことができます。

1. **Intel Hex**
2. **Generic Hex**
3. **Motorola Hex**(Sレコード)

Wrap Relative Jumps(相対分岐丸め)：AVRの**RJMP/RCALL**命令は±2K語に対応する12ビットPC相対変位を許します。4K語(8Kバイト)またはそれ未満のフラッシュプログラムメモリを持つデバイスに対して、この丸め任意選択はアセンブラの変位計算をアドレス指定可能なプログラムメモリ範囲で丸めるようにさせ、これらの命令を用いてプログラムメモリ全体にアドレス指定されることを許します。

4K語よりも多いプログラムメモリを持つデバイスに対しては、予期せぬ結果を避けるためこの任意選択をOFFにしてください。これがONのままの場合、アセンブラは丸めが効果を発する時に警告を生成します。

! **注意**： **RJMP/RCALL**丸めは4K語を超えるデバイスに対して規則違反です。 - 丸め任意選択をOFFにして**JMP/CALL**を使ってください。(訳補:これは丸めが発生する**RJMP/RCALL**をしないようにとの話です。)

この診断はアセンブラの初期版との互換性を保持するために異常ではなく警告として与えられますが、使用者によって異常として扱われるべきです。**JMP/CALL**の2語命令は22ビットの絶対アドレスを取り、代わりに使われるべきです。

Unsupported Instructions(未支援命令)：既定によって、この任意選択はアセンブラが実デバイスに対して未支援命令を見つけた時に警告を与えるように設定されます。任意で、異常を出力することができます。

Include Paths (-I)(インクルードパス)：第三者の単位部や独自のインクルードパスを使う時に追加のインクルードパスをここで設定することができます。

Other optimization flags(他の最適化フラグ)：これは最適化をあなた固有の必要性に逃えるように設定することができます。より多くの情報についてはアセンブラのヘルプ(**Help**(ヘルプ)⇒**View Help**(ヘルプ表示)⇒**AVR Assembler Help**(AVRアセンブラヘルプ))をご覧ください。

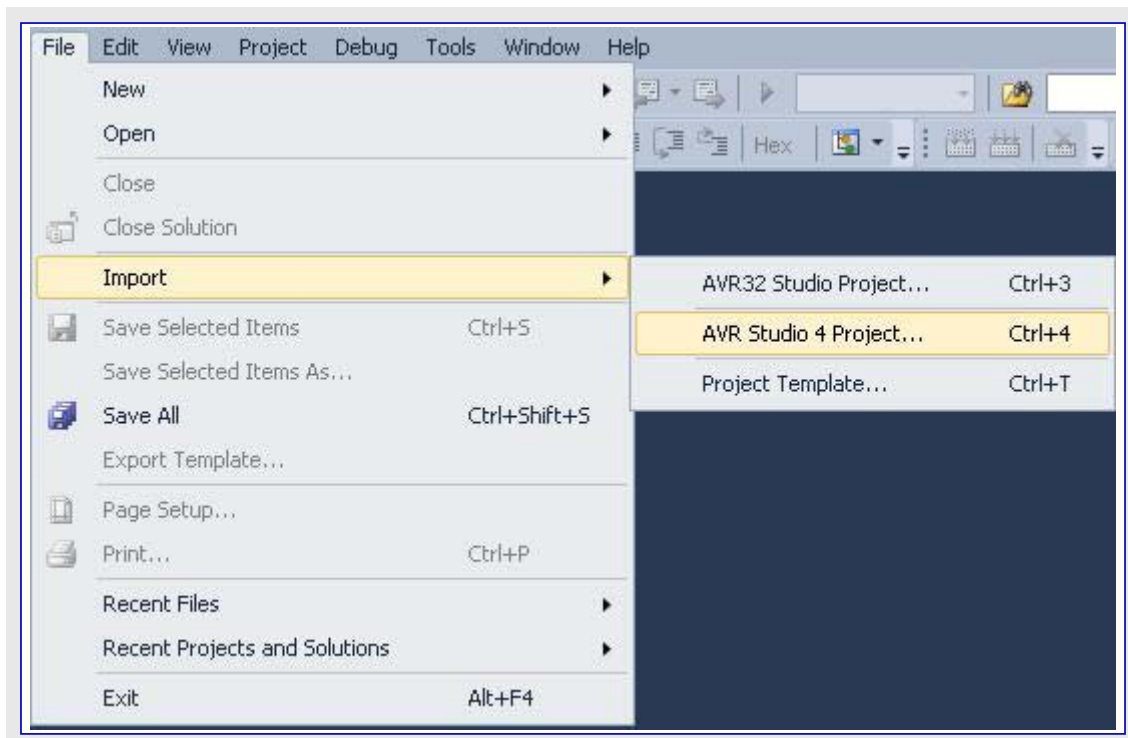
3.5. プロジェクトのインポート

3.5.1. 序説

Microchip Studioは以前の既存プロジェクトソースからプロジェクトのインポートを許します。本項は既存プロジェクトのインポート方法を詳述します。

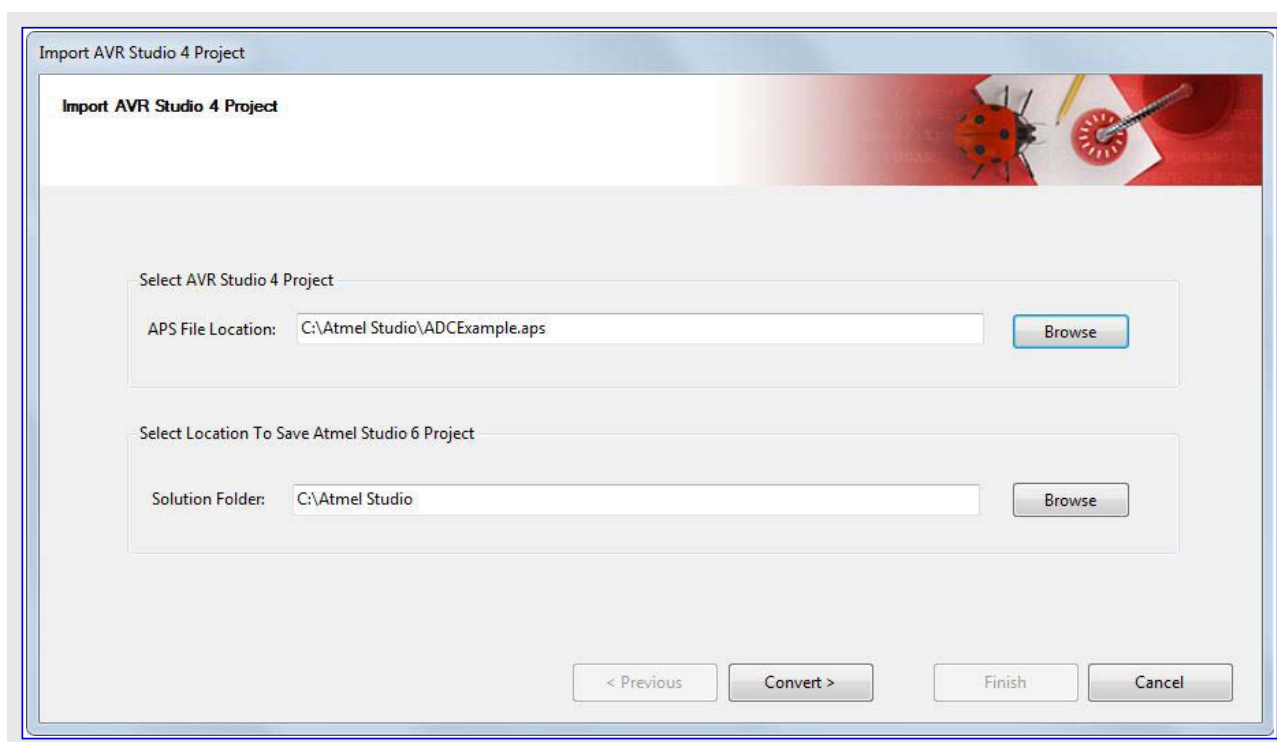
3.5.2. AVR® Studio 4プロジェクトのインポート

メニューでFile(ファイル)⇒Import(インポート)⇒AVR® Studio 4 Project(プロジェクト)をクリックするか、Ctrl+4を押下してください。



Import AVR Studio 4 Project(AVR Studio 4プロジェクトのインポート)ダイアログが現れます。

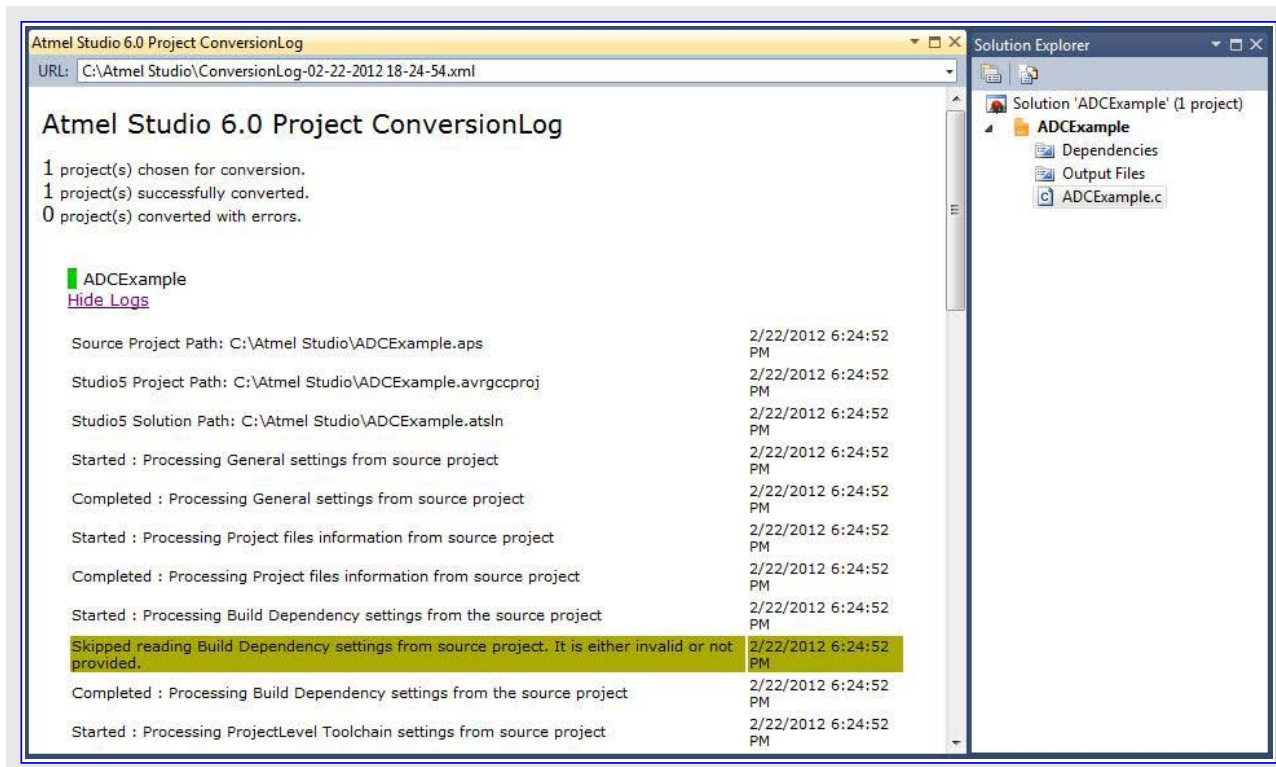
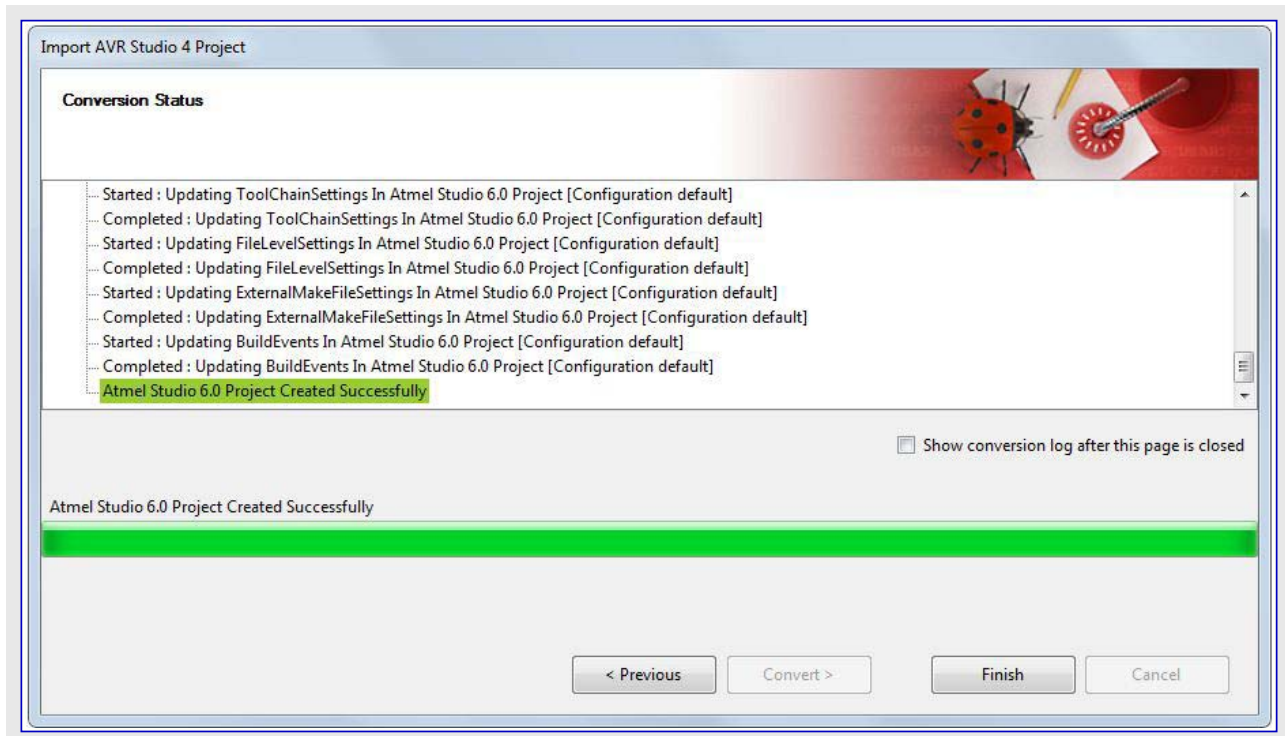
あなたのプロジェクトの名前を入力するか、またはAPS File Location(APSファイル位置)欄のBrowse(検索)鈕をクリックすることによってプロジェクトの場所を検索してください。



Microchip Studioは変換を進めて進捗の更新を与えます。警告と異常は要約ウィンドウで示されます。

完全な変換記録を表示するには>Show conversion log after this page is closed(このページが閉じられた後に変換記録を表示)をチェックしてください。

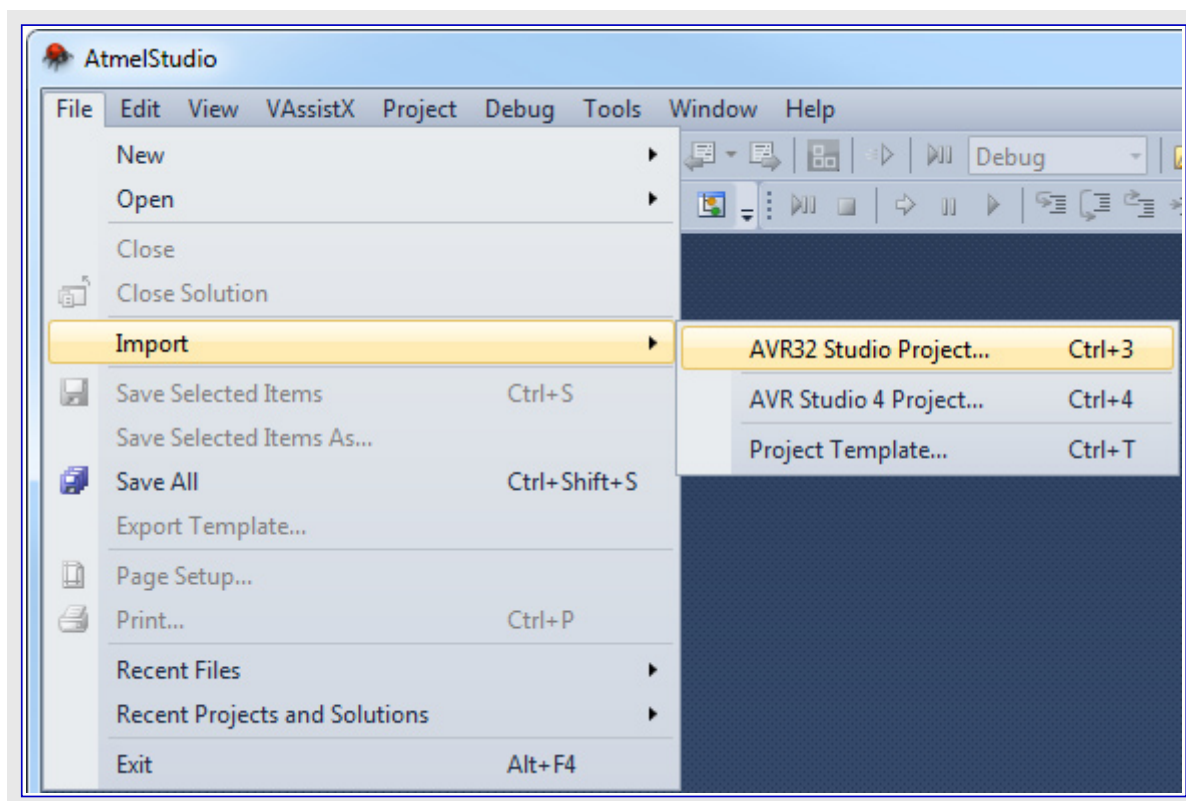
新たに変換されたプロジェクトへアクセスするにはFinish(終了)をクリックしてください。



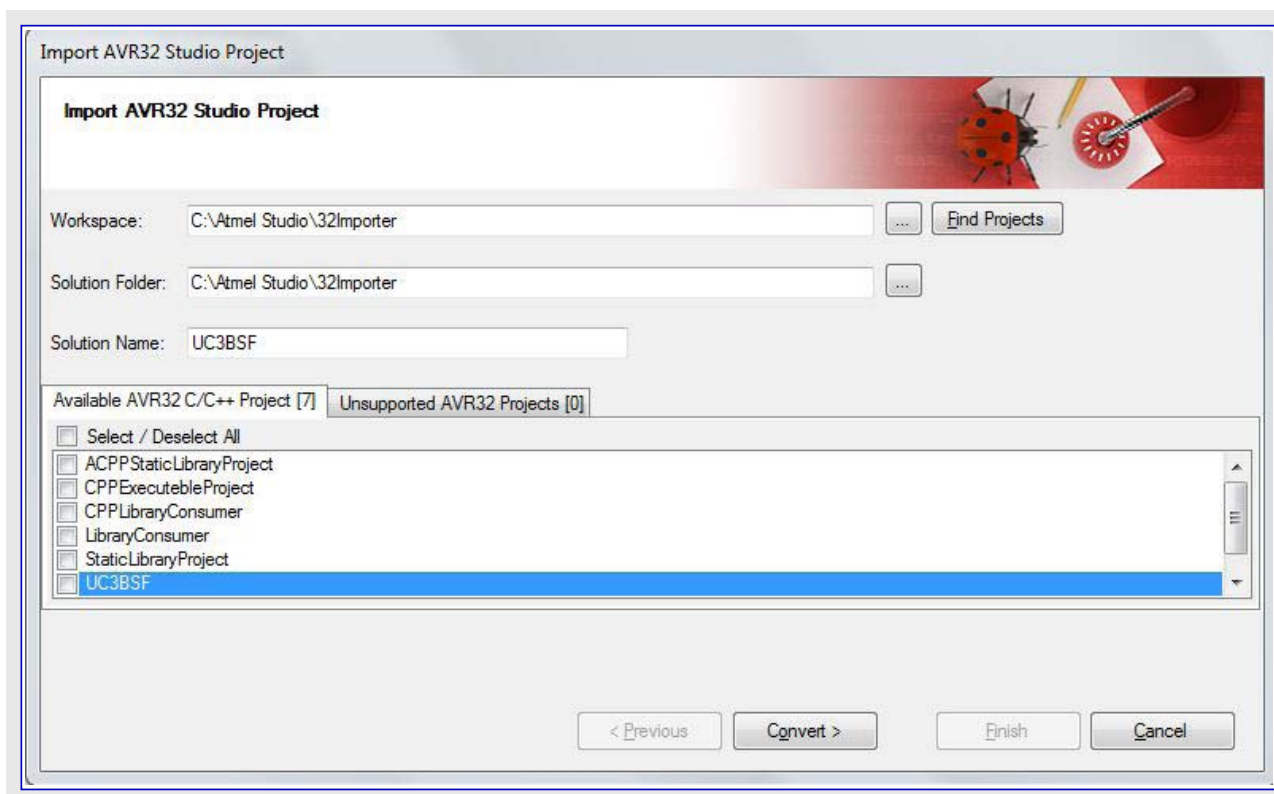
注: 現在、Solution(解決策)フォルダがAPSファイル位置と同じ場合、変換はプロジェクトファイルと解決策ファイルだけを追加します。他のファイルは全く変更されません。

3.5.3. AVR® 32 Studioプロジェクトのインポート

メニューでFile(ファイル)⇒Import(インポート)⇒AVR® Studio 32 Project(プロジェクト)をクリックするか、Ctrl+3を押下してください。

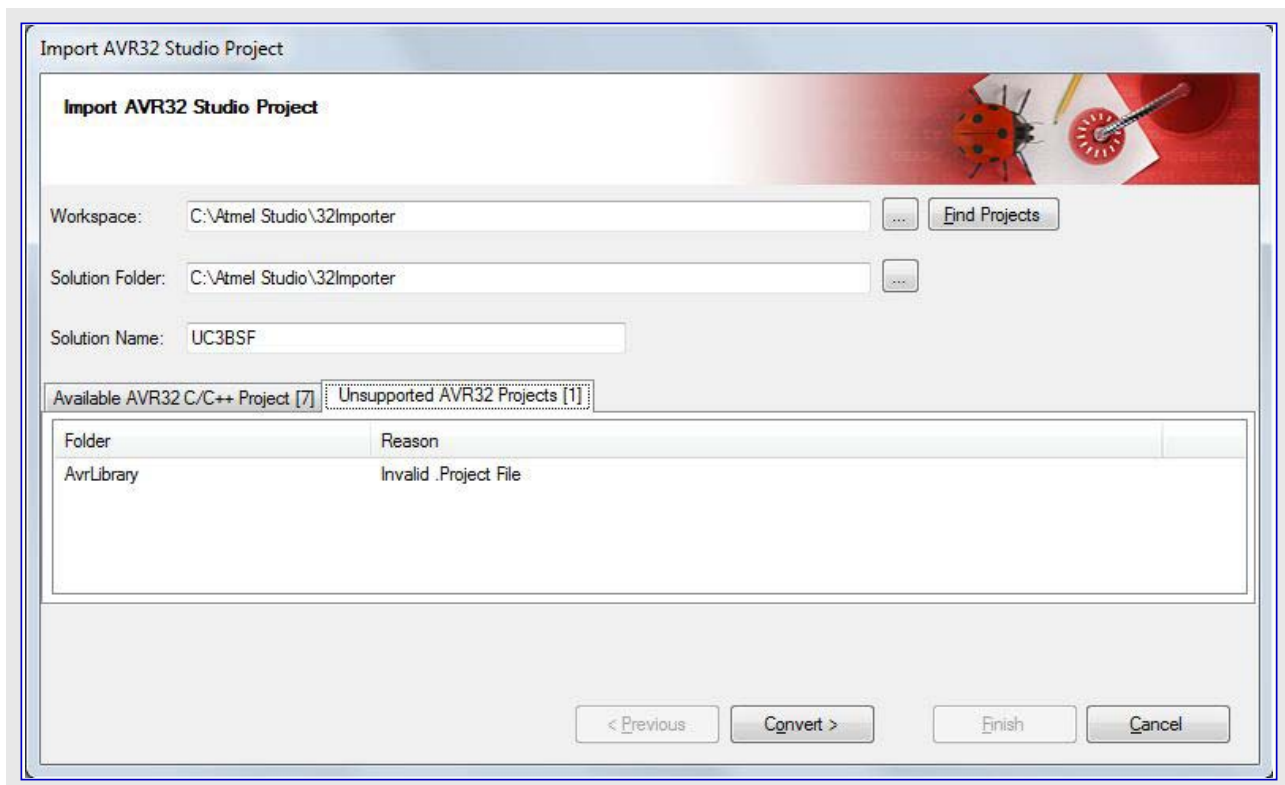


“Import AVR Studio 32 Project(AVR Studio 32プロジェクトのインポート)”ダイアログが現れます。



あなたの作業空間の名前を入力するか、またはWorkspace(作業空間)欄の...(検索釘)をクリックすることによって作業空間の場所を検索してください。全てのプロジェクトファイルを見つけるにはFind Projects(プロジェクト検索)をクリックして、作業空間で利用可能な他のフォルダを入れてください。

Available AVR32 C/C++ Project(利用可能なAVR32 C/C++プロジェクト)タブはインポートすることができる全てのAVR32 C/C++プロジェクトが入れられ、これは利用可能なプロジェクトの総数も表示します。

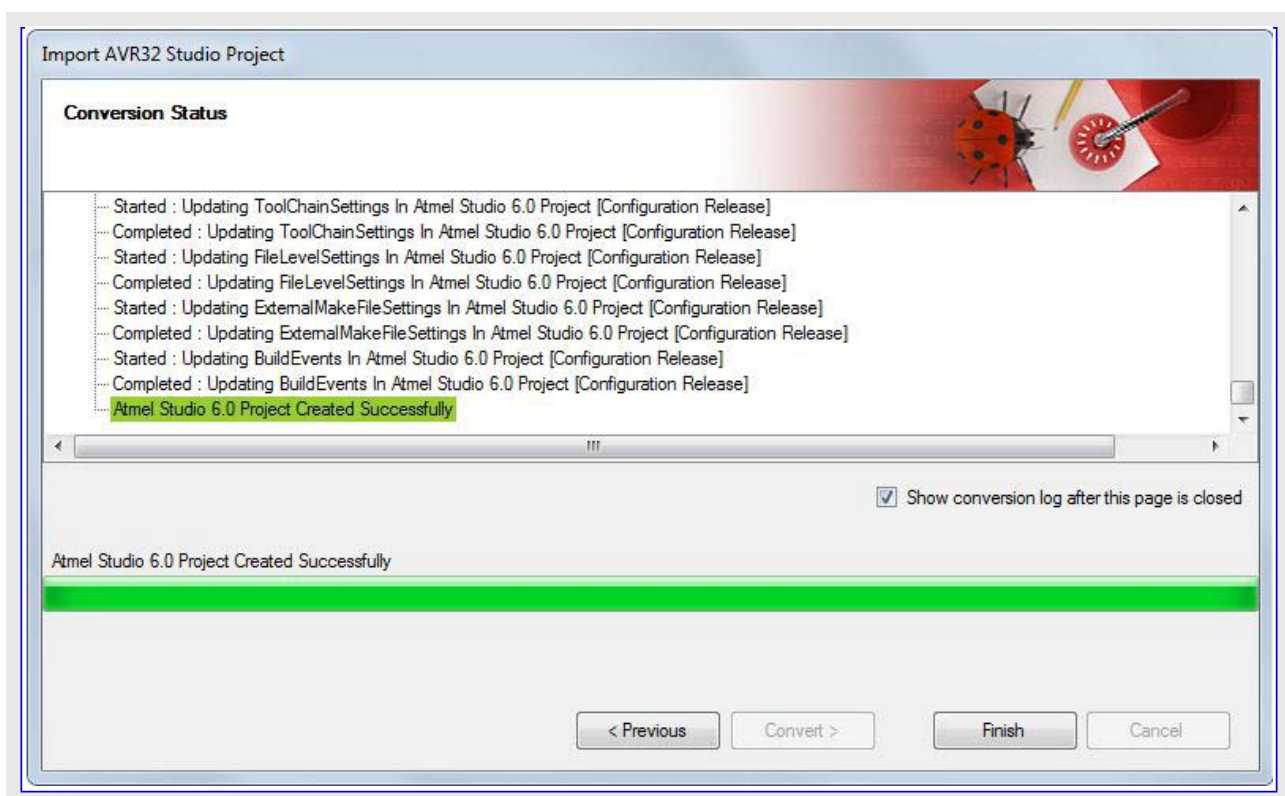


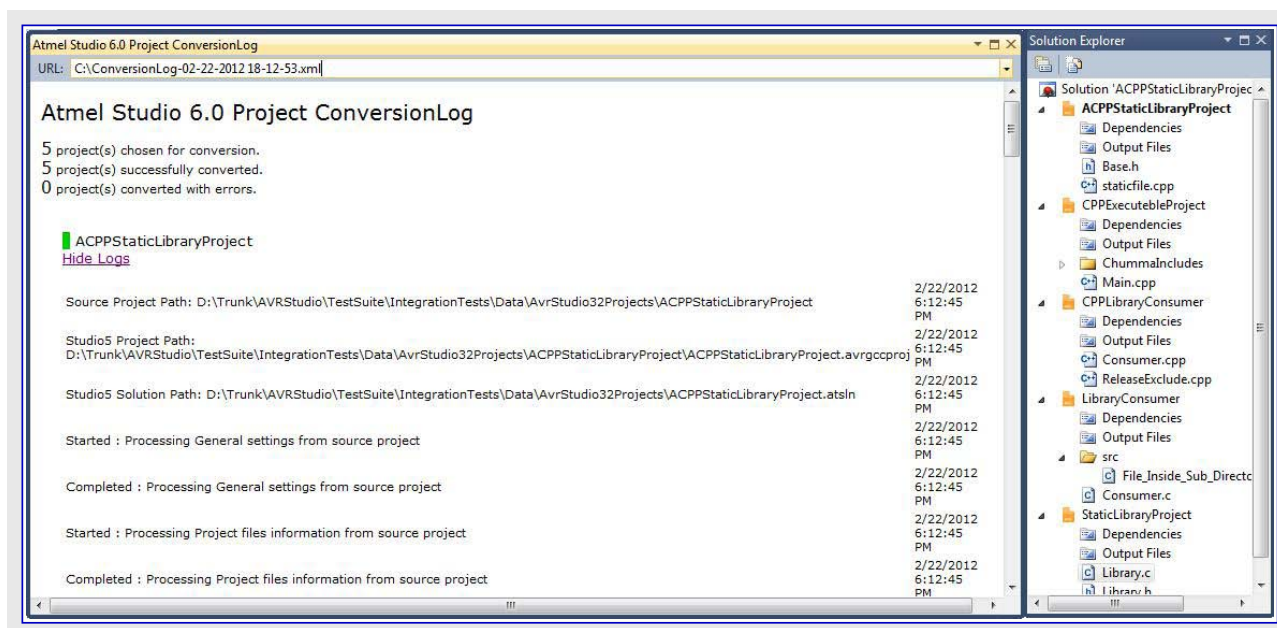
Unsupported AVR32 Projects(非支援AVR32プロジェクト)タブはインポートすることができない全ての無効AVR32プロジェクトが入れられ、これは理由と共に変換不能なプロジェクトの総数も表示します。

Microchip Studioは変換を進め、進捗の更新を与えます。警告と異常は要約ウィンドウで示されます。

完全な変換記録を表示するには**Show conversion log after this page is closed**(このページが閉じられた後に変換記録を表示)をチェックしてください。

新たに変換されたプロジェクトへアクセスするには**Finish**(終了)をクリックしてください。





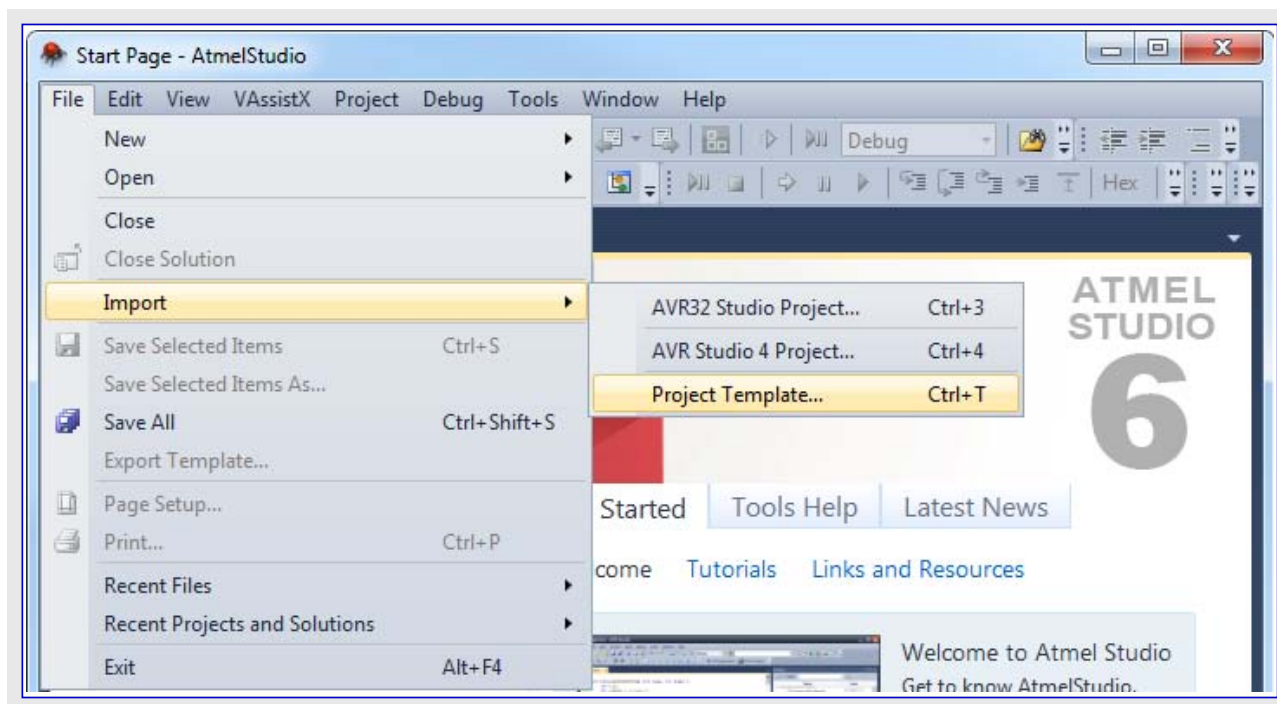
注: ・ 現在版のAVR32インポート部はAVR32 C/C++プロジェクトを支援します。

- ・ AP7デバイスシステムは現在Microchip Studioによって支援されません。
- ・ 現在、Solution(解決策)フォルダが作業空間フォルダと同じ場合、変換はプロジェクトファイルと解決策ファイルだけを追加します。他のファイルは全く変更されません。
- ・ 前/後構築設定はインポートされません。
- ・ 自動生成一覧(*.lss)ファイルはインポートされません。

3.5.4. プロジェクト雛形のインポート

File(ファイル)⇒Import(インポート)⇒Project Template...(プロジェクト雛形)またはCtrl+Tによっていくつかの予め定義されたプロジェクトをMicrochip Studioにインポートすることができます。

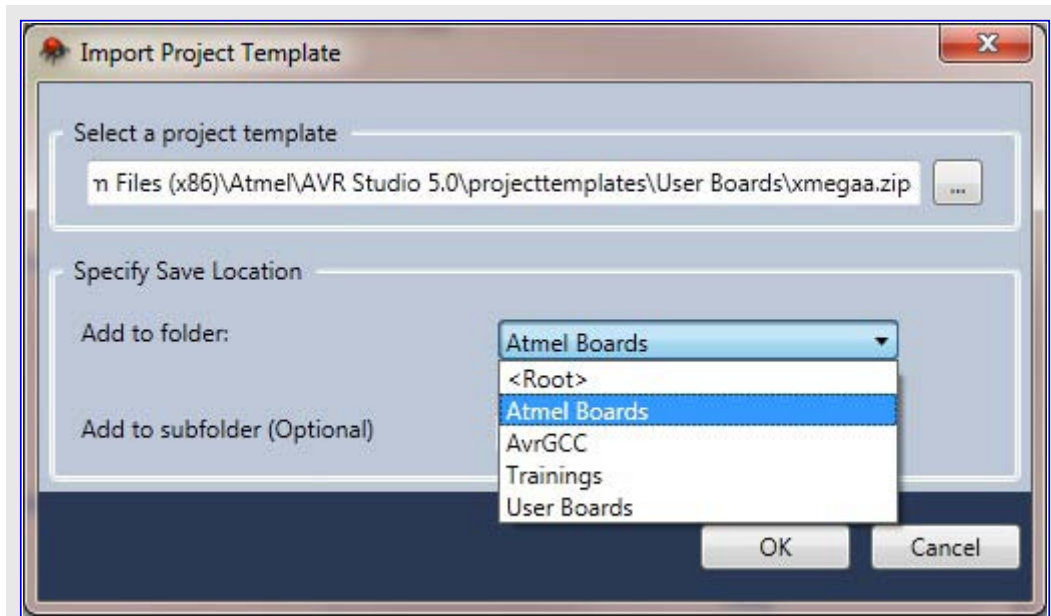
これらの雛形は新しいプロジェクトの作成を始める、または現在のプロジェクトを拡張するための開始点を提供します。プロジェクト雛形は標準アセンブリ参照を含む特定のプロジェクト形式に対して必要とされる基本ファイルを提供し、既定プロジェクトプロパティとコンパイラ任意選択を設定します。



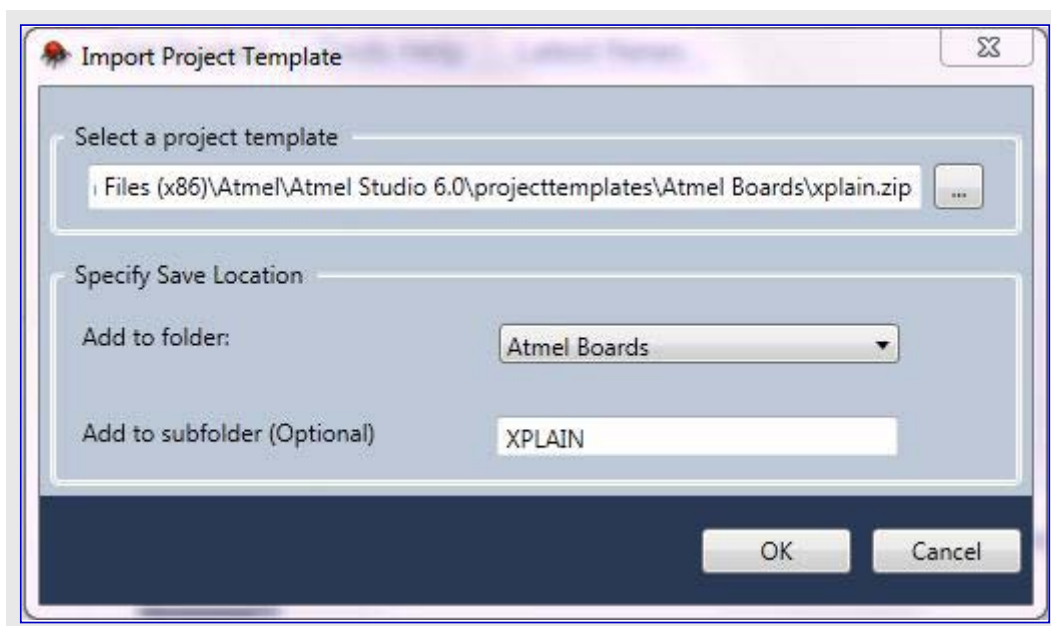
“Import Project Template(プロジェクト雛形インポート)”ウィンドウは以下を指定します。

- あなたのプロジェクト雛形の場所。
- 保存場所。コンボ枠はNew Project(新規プロジェクト)⇒Installed Template(インストール済雛形)で利用可能なインストール済雛形を示します。

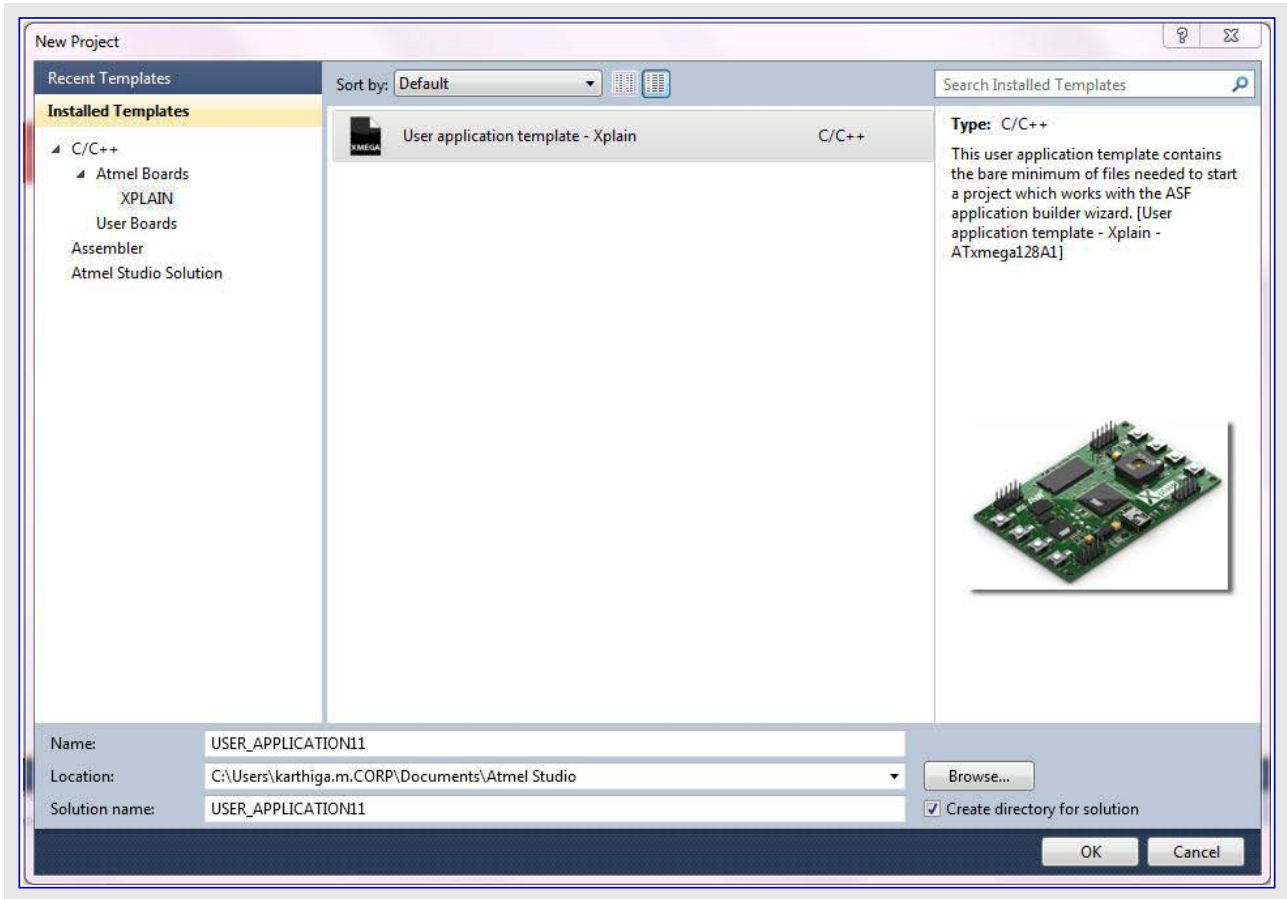
あなたのプロジェクトを追加したいどれかの下の何れかの雛形を選択してください。”Add to folder(フォルダへ追加)”で<root>を選ぶことによってあなたの雛形をルートに追加することもできます。



- 指定した”Add to subfolder (Optional)(副フォルダ’に追加(任意選択))”下のフォルダ名を指定することによって、あなたのプロジェクト雛形を追加したい独立したフォルダを作成することができます。



結果として生じたプロジェクト雛形は既存のインストール済み雛形に追加され、File(ファイル)⇒New(新規)⇒Project...(プロジェクト)またはCtrl+Shift+Nからアクセスすることができます。



注: プロジェクト雛形インポート部は同じ版に対して作成された雛形で動きます。

3.6. Microchip Studioでのオブジェクト ファイル デバッグ

3.6.1. 序説

デバッグ作業はMicrochip Studioによって支援されるオブジェクト ファイルを読み込む必要があります。デバッグ ファイルはデバッグのためのシンボル情報を含みます。

3.6.2. Microchip Studioが支援するデバッグ形式

表3-12. Microchip Studioによって支援されるオブジェクト ファイル形式

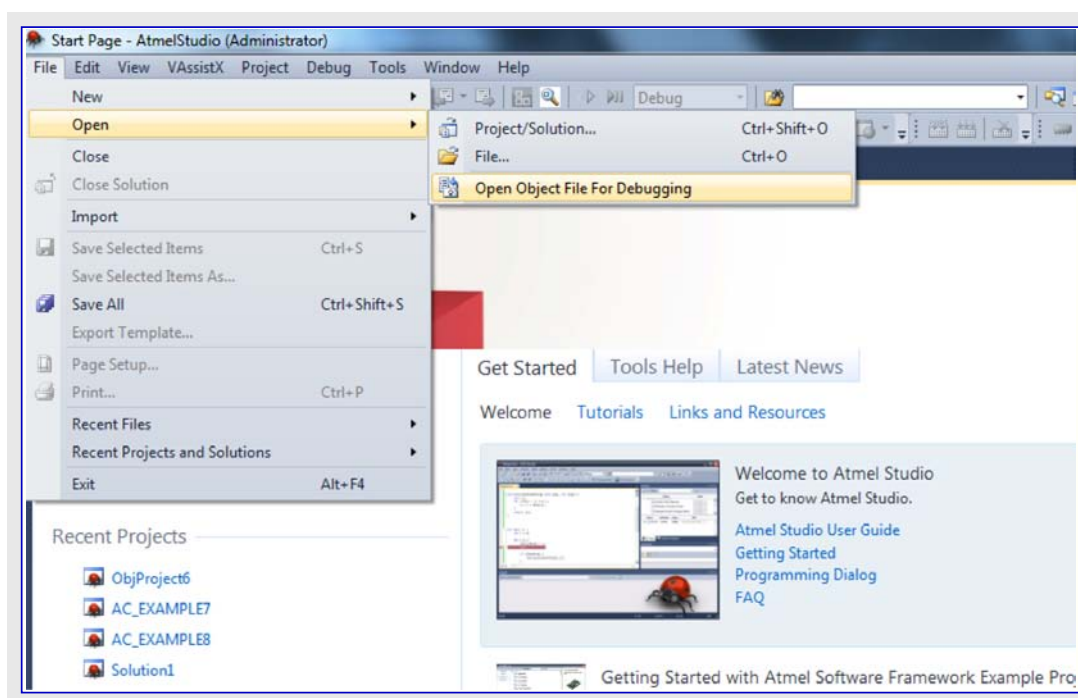
オブジェクト ファイル形式	拡張子	説明
UBROF	.d90	UBROFはIAR専有形式です。デバッグ出力ファイルは全ての監視形式を支援するための完全なデバッグ情報一式とシンボルを含みます。UBROF8とそれ以前版が支援されます。これはIAR EW 2.29とそれ以前版の既定出力形式です。IAR EW 3.10とそれ以降版でUBROF8生成に強制する方法を下をご覧ください。
ELF/DWARF	.elf	ELF/DWARFデバッグ情報は公開標準です。このデバッグ形式は全ての監視形式を支援するための完全なデバッグ情報一式とシンボルを含みます。Microchip Studioによって読まれる形式版はDWARF2です。DWARF2出力用に構成設定されるAVR-GCC版はこの形式を生成することができます。
AVRCOFF	.cof	COFFはMicrochip Studioによって支援される拡張やツールを作成する第三者の供給者を意図された開放標準です。
AVRアセンブラ形式	.obj	AVRアセンブラ出力ファイル形式はソース段階実行用のソース情報を含みます。これは内部Microchip形式専用です。いくつかの監視情報を得るために自動的に.mapファイルが解析されます。

デバッグに先立って、上の形式の1つのようなデバッグ ファイルを生成するようにコンパイラ/アセンブラを構成設定することを確実にしてください。第三者のコンパイラ版はELF/DWARFオブジェクト ファイル形式を出力すべきです。

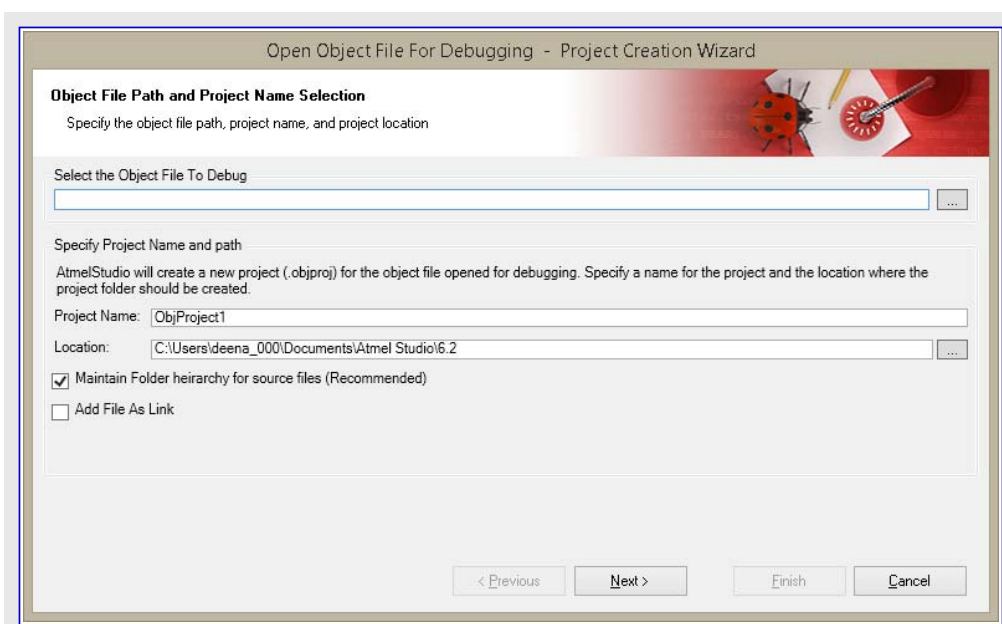
3.6.3. デバッグ用にオブジェクト ファイルを開く

オブジェクト プロジェクトを作成するための手順

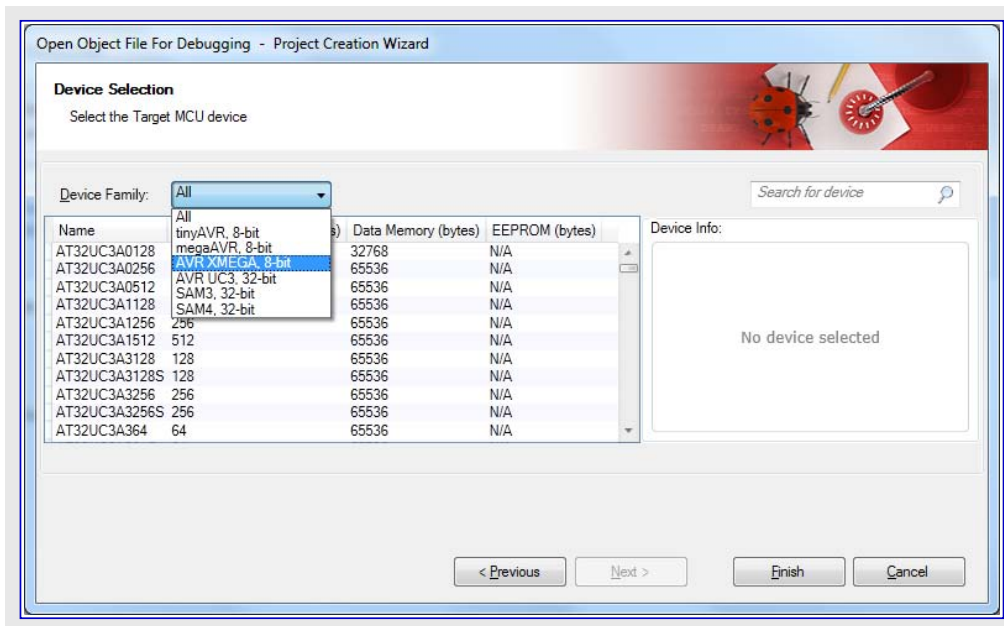
- File(ファイル)メニューで、Open(開く)をクリックし、そしてOpen Object File For Debugging(デバッグ用のオブジェクト ファイルを開く)をクリックしてください。
- Open Object File For Debugging(デバッグ用のオブジェクト ファイルを開く)ウィザードが現れます。



- Select the object file to debug(デバッグのためのオブジェクト ファイル選択)で、デバッグするためのオブジェクト ファイルを選んでください。Microchip Studioはそのオブジェクト形式を支援しなければなりません。
- Project Name(プロジェクト名)で、プロジェクトの名前を入力してください。Microchip Studioは名前を示唆し、望むなら上書きすることができます。
- Location(場所)で、保存場所を選んでください。Microchip Studioは場所を示唆し、望むなら上書きすることができます。
- Maintain Folder hierarchy for source files(ソース ファイル用フォルダ階層を維持)任意選択は既定によって、ソースプロジェクトのそれ、即ちオブジェクト ファイルを作成するのに使うプロジェクトとしてSolution Explorer(解決策エクスプローラ)内で同じフォルダ構造を作成するが選ばれます。さもなければ、全てのファイルがプロジェクト ファイルのルート フォルダに追加され、即ち使用者はSolution Explorer(解決策エクスプローラ)でどのフォルダも見えません。
- Add File As Link(リンクとしてファイルを追加)任意選択は既定によって、オブジェクト プロジェクトはプロジェクト ディレクトリ内の局所複製を除き、その元の場所からファイルを参照するが選ばれます。この任意選択が選ばれない場合、Microchip Studioはファイルをオブジェクト プロジェクト ディレクトリ内に複製します。

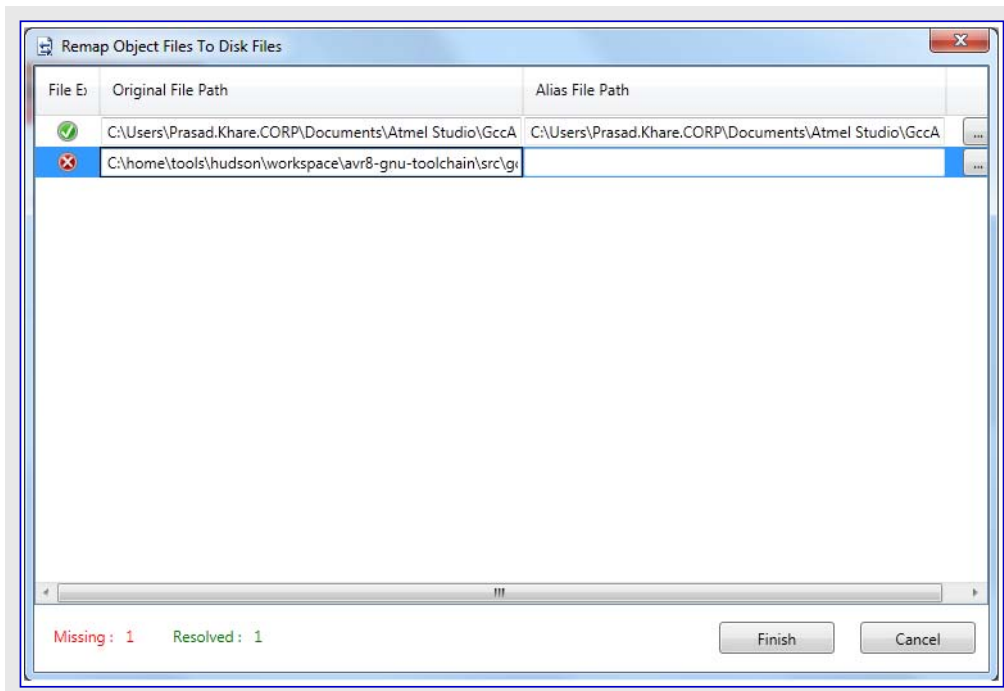


- **Next**(次へ)をクリックしてください。 **Device Selection**(デバイス選択)ダイアログが現れます。
 - 適切な目的対象デバイスを選んでください。目的対象デバイスは元々オブジェクトファイルを作成するために選ばれたものと同じであるべきです。

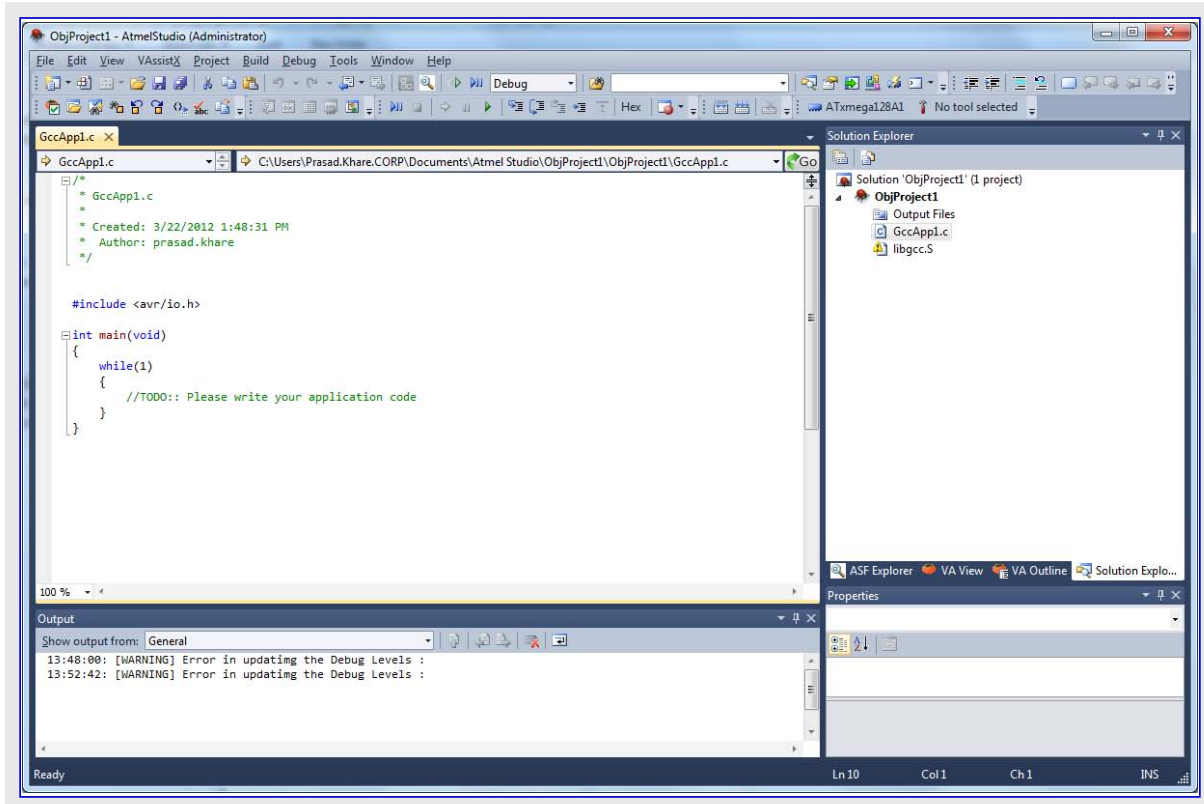


- **Finish**(終了)をクリックしてください。オブジェクトプロジェクトファイル再配置部が現れます(下の複写画面をご覧ください)。このダイアログはプロジェクトファイルの位置を再配置を許します。

使用者が何れかの元ファイルに対して親フォルダを解決する場合、それに続くディレクトリ内の他の全てのファイルが再帰的に再配置されます。故に、単に1つだけを再配置することによってファイル数分を再配置するので使用者に対して有用です。



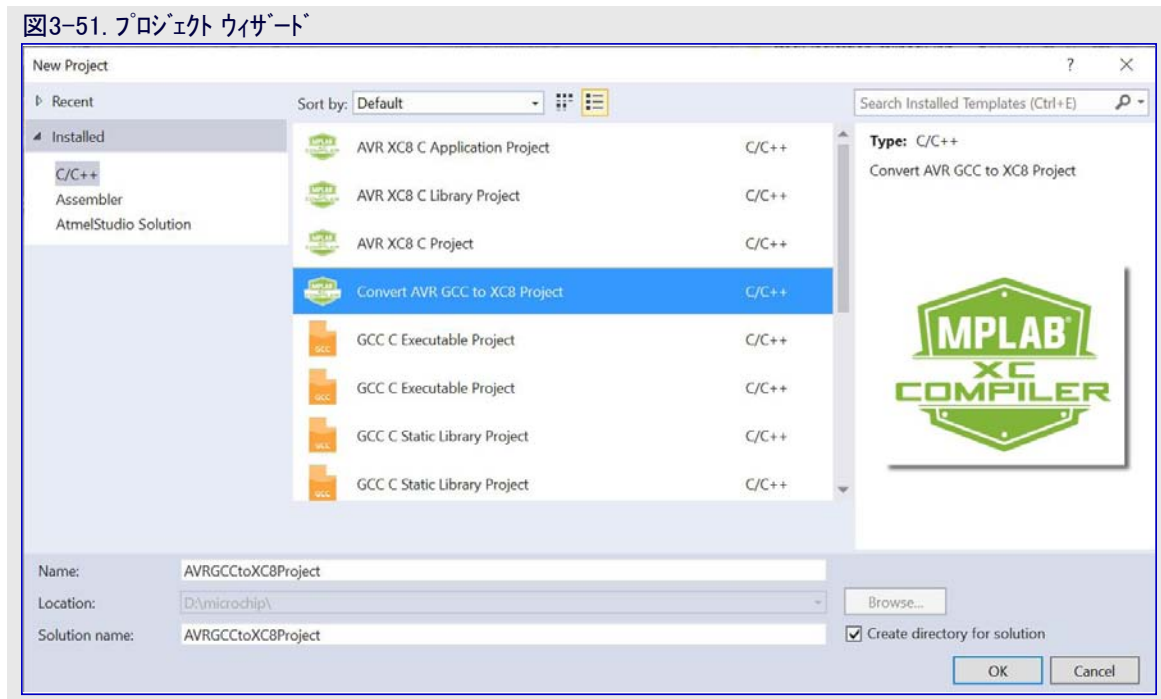
- 次にオブジェクトファイルが作成されます。再配置されないファイルのプロパティが警告の合図と共に”libgcc.S”のようにSolution explorer (解決策エクスプローラ)で示されます。このプロジェクトをデバッグするにはF5を押してください。



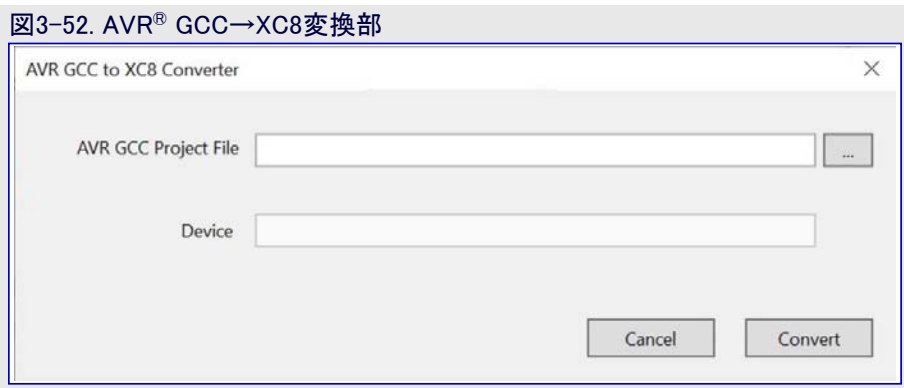
3.7. AVR® GCCプロジェクトのXC8プロジェクトへの変換

既存のAVR GCCプロジェクトをXC8プロジェクトに変換するには以下のこれらの手順に従ってください。

1. Microchip StudioメニューからFile(ファイル)⇒New(新規)⇒Project...(プロジェクト)を選ぶことによって”Project Wizard(プロジェクト ウィザード)”を開いてください。



2. **Convert AVR GCC to XC8 Project**(AVR GCCをXC8プロジェクトに変換)を選び、その後に進めるために**OK**を押してください。
3. **AVR GCC to XC8 Converter**(AVR GCC→XC8変換部)ダイアログが現れます。AVR GCC Cプロジェクト ファイル(*.cproj)を検索して選んでください。



4. プロジェクトで使われるデバイスが**Device**(デバイス)枠で自動的に設定されます。デバイスが変換作業の流れによって支援されない場合、適切な反応が与えられます。



助言: プロジェクトでAVRデバイス用デバイス系統一括(ファミリー パック)がインストールされていない場合、その後にMicrochip Studioメニューの**Tools**(ツール)⇒**Device Pack Manager**(デバイス一括管理部)からそれをインストールしてください。

5. AVR GCCプロジェクトをXC8プロジェクトに変換するために**Convert**(変換)をクリックしてください。
6. プロジェクトを構築するためにプロジェクト節点で右クリックして**Build**(構築)を選んでください。

4. デバッグ

4.1. 序説

Microchip Studioは組み込みシミュレータ、または様々なツール(「6.3. 利用可能なツール表示」をご覧ください)、例えばAVR ONE!、JTAGICE3、Atmel ICE(別途購入)に対して目的対象とすることができます。

4.1.1. デバッグ環境から独立したデバッグ基盤

どのデバッグ基盤が動いているかと無関係に、Microchip Studio環境は同じように見えます。デバッグ基盤切り替え時、全ての環境任意選択が新しい基盤に対して維持されます。いくつかの基盤は独自機能を持ち、新しい機能/ウィンドウが現れるでしょう。

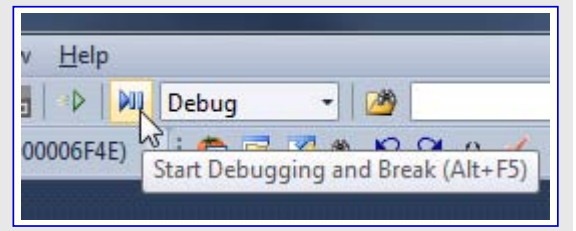
4.1.2. 基盤間の違い

全てのデバッグ基盤がデバッグ環境で同じに見えるとは言え、それらの間には小さな違いがあります。実時間エミュレータは大きなプロジェクトに対してシミュレータよりもかなり速くなります。エミュレータはシステムが実際のハードウェア環境に接続されている間にデバッグすることを許し、一方シミュレータは印加するための予め定義された起因だけを許します。シミュレータでは、全てのレジスタが表示のために潜在的に常に利用可能で、そしてそれはエミュレータの場合にないかもしれません。

4.2. デバッグ作業開始

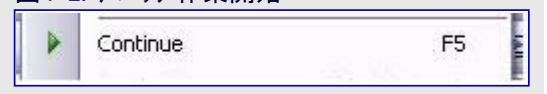
デバッグ作業を開始して停止するには、**Alt+F5**を押すか、またはメニューから**Debug**(デバッグ)⇒**Start Debugging and Break**(デバッグと中断の開始)を選んでください。代わりに、右で図解されるようにツールバーの鈕を押してください。

図4-1. デバッグ作業開始



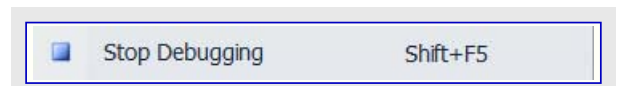
デバッグ作業を開始して実行を続けるには、**F5**を押すか、継続シンボルを持つツールバーの鈕を押すか、または右で図解されるようにメニューから**Debug**(デバッグ)⇒**Continue**(継続)を選んでください。

図4-2. デバッグ作業開始



4.3. デバッグ作業終了

デバッグ作業を終了するには**Stop Debugging**(デバッグ中止)鈕またはキーボードショートカットの**Shift+F5**を使ってください。



4.4. 目的対象への取り付け

目的対象を取り付けるにはDebug(デバッグ)メニューでAttach to Target(目的対象へ取り付け)任意選択、またはデバッグ ツールバーで取り付け アイコンを使ってください。これはMicrochip Studioに新しい応用のアップロードやリセットを起こすことなく選んだ目的対象でのデバッグ作業を開始させます。一旦デバッグ作業が確立されると、目的対象のコードは停止され、目的対象の現在の実行位置がプロジェクト内のコードに置かれ、これは目的対象の状態が維持され、通常のデバッグ技法で調べることが可能で、プログラムが現在位置で停止することを意味します。成功裏に取り付け後に完全な走行制御とシンボリック デバッグが利用可能でしょう。

- 注:**
- プロジェクト外のコードはこの配置の正当性を確認するための何れの可能性も除き、走行する目的対象の内容に配置されます。これはプロジェクトが目的対象でないコードを含む場合、変数と関数がプロジェクト外の目的対象での違うコード位置を持つかもしれないため、状態と走行制御が真実を反映しないかもしれないことを意味します。
 - 目的対象をリセットすることなくデバッグ作業を起動する能力は基本構造依存です。全ての基本構造がこの機能を支援する訳ではありません。



注意: 目的対象への物理的なデバッグ探針接続は、殆どのデバッグ探針がデバイスのリセット線へ電氣的な接続が必要なので、目的対象をリセットにさせます。これを避けるために標準の電氣的予防処置を取ってください。

4.5. デバッグなしでの開始

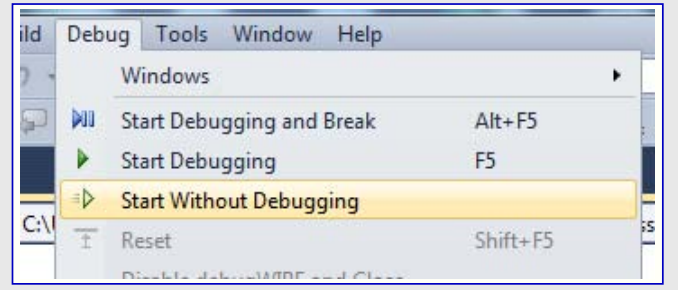
4.5.1. 1クリックプログラミング – プログラミングと走行

Start Without Debugging(デバッグなしで開始)命令はプログラミング ダイアログへの1クリック切り替えです。メニューからDebug(デバッグ)⇒Start Without Debugging(デバッグなしで開始)を選ぶ、またはツールバーで 鈕を押すことによって実行してください。

これは(何か変更が行われる場合に)解決策が構築され、デバッグ作業を開始することなく目的対象デバイスを書きます。

Start Without Debuggingはプロジェクト任意選択で指定したツールとインターフェースの設定を使います。これは全てでプロジェクトに関連されない独立型プログラミング ダイアログ使用時に行われるものと違います。

図4-3. デバッグなしで開始



- 注:** デバッグだけでなく、書き込み器とスタータ キットはStart Without Debugging命令で使うこともできます。

Start Without Debugging命令はEEPROM、ヒューズ、施錠ビット、使用者識票(XMEGAのみ)部分も、これらがオブジェクト ファイルに存在する場合に書き込みます。GCCコンパイラはこのような部分を含むELFオブジェクト ファイルを生成することができます。より多くの情報については「3.2.7.7. 他のメモリ形式でのELFファイル作成」をご覧ください。

- 注:** 使用者識票はStart Without Debuggingによって消去されません。ELFファイルから書かれる使用者識票はデバイス内の内容とANDされます。識票をファイル内の物との置換を望む場合、手動でuser signature erase(使用者識票消去)を実行しなければなりません。

4.5.2. キーボード ショートカット

既定によってこの機能に対するキーボード ショートカットはありませんが、その機能を沢山使う場合にその追加を望むかもしれません。それを追加するには、Tools(ツール)⇒Options(任意選択)メニューをクリックしてEnvironment(環境)⇒Keyboard(キーボード)へ行ってください。Show commands containing(含む命令の表示)入力領域でstartwithoutdebuggingを入力し、一覧でDebug.StartWithoutDebuggingを選んでください。その語にPress shortcut keys(ショートカット キー押下)入力領域を選択し、望むキーの組み合わせを押してください。例えばCtrl+F5を押すことができます。このショートカットが既にBreakAll(全て中断)に割り当てられていることに注意してください。それを上書きすることを選ぶなら、新しいキーボード ショートカットを割り当てるためにAssign(割り当て)鈕を押してください。または、未使用のキー組み合わせを選ぶことができます。

図4-4. キーボード ショートカット追加

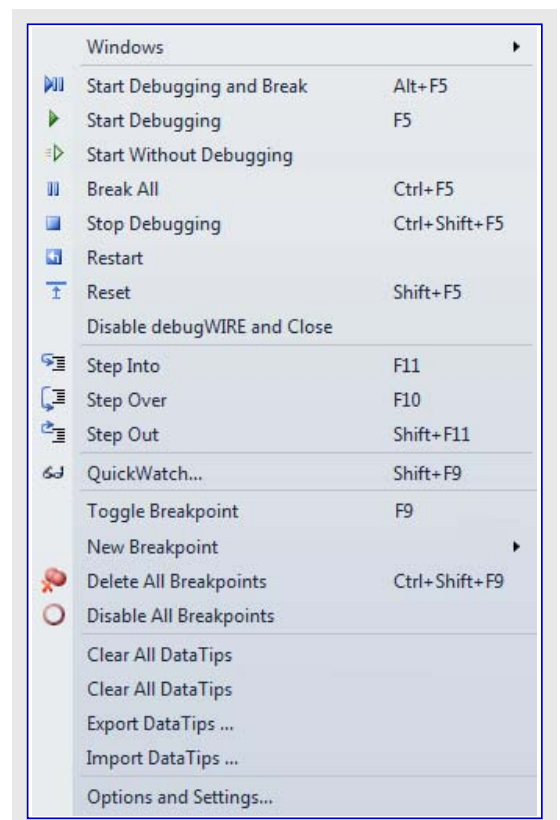


4.6. デバッグ制御

デバッグが制御するのに様々な命令が利用可能です。それらはDebug(デバッグ)メニューと様々なツールバーの両方から利用可能です。Microchip Studio統合開発環境(IDE: Integrated Development Environment)は設計動作とデバッグ動作の2つの主な動作形態を持ちます。設計動作はソースコード プロジェクトを編集している時で、一方デバッグ動作はプロジェクトをデバッグする時です。IDEは動作に適応してメニューとツールバーを変えます。

注: いくつかのデバッグ命令はデバッグ動作、デバッグ動作内のいくつかで利用可能です。

- 設計動作に於いて、利用可能なデバッグ命令はデバッグ作業を開始するそれら、例えば、Start Debugging and Break(デバッグ開始と中断)、Start Debugging(デバッグ開始)、Start without Debugging(デバッグなしで開始)です。
- デバッグ動作に於いて、BreakAll(全て中断)、Step Out(外側まで実行)、Reset(リセット)のような命令を見つけましょう。



- Start Debugging and Break** (デバッグ開始と中断) : デバッグを開始し、プログラムの最初の文で実行を中断します。
- Start Debugging** (デバッグ開始) : デバッグを開始してプログラムを走行します。デバッグ動作での停止では実行を再開します。
- Start Without Debugging** (デバッグなしで開始) : デバッグを開始しないでプロジェクトを書き込みます。詳細については「[4.5. デバッグなしでの開始](#)」をご覧ください。
- Break All** (全て中断) : デバッグを停止します。
- Stop Debugging** (デバッグ中止) : デバッグ作業を停止して終了し、設計動作へ戻ります。
- Restart** (再始動) : デバッグを再始動してプログラムを再設定します。
- Reset** (リセット) : プログラムを最初の文にリセットします。
- Disable debugWire and Close** (デバッグWIREを禁止して閉じる) : デバッグWIREインターフェースを使ってデバイスをデバッグする時に利用可能。この命令はデバッグWIREを禁止(ISPインターフェースを許可)してデバッグ作業を終えます。
- Step Into** (1命令実行) : 1命令を実行。逆アセンブリレベルの時は1つのアセンブリレベル命令が実行され、さもなければ1つのソースレベル命令が実行されます。
- Step Over** (外側1命令実行) : **Step Info**と同様に、**Step Over**は1命令を実行します。けれども、その命令が関数/サブルーチンの呼び出しを含む場合、その関数/サブルーチンもまた実行されます。**Step Over**の間に使用者中断点に出会うと、実行が停止されます。
- Step Out** (外側まで実行) : 現在の関数が完了されるまで実行を継続。**Step Out**の間に使用者中断点に出会うと、実行が停止されず。プログラムが最上位の時に、**Step Out**命令が発行された場合、プログラムは中断点に達する、または使用者によって停止されるまで実行を続けます。
- Quick Watch** (一瞬監視) : カーソル下の変数または式の一瞬監視を追加。詳細については「[4.9.4. 一瞬監視と監視](#)」をご覧ください。
- Toggle Breakpoint** (中断点ON/OFF) : カーソルが置かれた場所の命令に対して中断点(ブレークポイント)の状態を交互切り替え。この機能がソースウインドウまたは逆アセンブリウインドウが活性ウインドウの時にだけ利用可能なことに注意してください。
- New Breakpoint** (新中断点作成) : カーソルの位置に新しい中断点を作成。より多くの情報については「[4.7. 中断点](#)」をご覧ください。
- Disable All Breakpoint** (全中断点禁止) : この機能は禁止されている中断点を含めて設定された全プログラム中断点を解除します。
- Clear All Data Tips** (全データ情報解除) : 記された全データ情報を解除、「[4.10. データ情報](#)」をご覧ください。
- Export Data Tips** (データ情報を出力) : 記された全データ情報をVisual Studio Shell形式で保存。
- Import Data Tips** (データ情報を入力) : Visual Studio Shellファイルからデータ情報を読み込み。
- Options and Settings** (任意選択と設定) : デバッグ任意選択と設定、「[10.3.5. デバッグ](#)」をご覧ください。

4.7. 中断点 (ブレークポイント)

4.7.1. 中断点での一般情報

中断点は指定条件が起こる、例えば与えられた命令が今にも実行されようとしている時にプログラムの実行を一時的に停止することをデバッグに伝えます。





中断点は必要な場所や時に実行を一時停止することを許す強力な道具を提供します。行単位または命令単位でコードを通して段階実行するよりも、むしろ中断点に的中するまで走行することを許し、その後にデバッグを開始することができ、これはデバッグ処理を高速化します。

4.7.1.1. 中断点絵記号

ソースウインドウと逆アセンブリウインドウは左余白で絵記号と呼ばれるシンボルを表示することによって中断点位置を示します。下表はこれらの絵記号を記述します。

中断点絵記号上にマウスを留めると、中断点情報がより多くの情報と共に現れます。この情報は異常と警告の中断点に対して特に有用です。

表4-1. 中断点絵記号

絵記号	説明
	標準中断点。一様の絵記号は中断点が許可されることを示します。中空の絵記号は禁止されることを示します。
	高度な中断点。活性/禁止。+符号は中断点がそれに付随された最低1つの(条件、的中回数、選別のような)高度な機能を持つことを示します。
	中断点異常。×は中断点が異常状態のために設定することができないことを示します。
	中断点警告。感嘆符は中断点が一時的な条件のために設定することができないことを示します。通常、これは中断点または追跡点の位置のコードが読み込まれていないことを意味します。これは処理に伴ってその処理用のシンボルが読み込まれていない場合にも見ることができます。コードまたはシンボルが読み込まれると、中断点が許可され、絵記号が変わります。

4.7.2. 中断点での操作

4.7.2.1. 中断点設定

1. ソースウィンドウで中断点を設定しない場所の実行可能コードの行をクリックしてください。右クリックメニューでBreakpoint(中断点)をクリックしてその後にInsert Breakpoint(中断点挿入)をクリックしてください。

--または、--

ソースウィンドウで中断点を設定しない場所の実行可能コードの行をクリックしてください。Debug(デバッグ)メニューでToggle Breakpoint(中断点交互ON/OFF)をクリックしてください。

4.7.2.2. アドレス中断点設定

1. Debug(デバッグ)メニューでWindows(ウィンドウ)を示し、その後に逆アセンブリウィンドウが未だ見えていなければDisassembly(逆アセンブリ)をクリックしてください。この任意選択が見えるためにはデバッグ作業であることが必要です。
2. 逆アセンブリウィンドウでコードの行をクリックし、その後にDebug(デバッグ)メニューでToggle Breakpoint(中断点交互ON/OFF)をクリックしてください。

--または、--

コードの行を右クリックし、その後にInsert Breakpoint(中断点挿入)を選んでください。

4.7.2.3. 中断点位置編集

1. 中断点ウィンドウで中断点を右クリックし、その後に右クリックメニューでLocation(位置)をクリックしてください。

--または、--

ソース、逆アセンブリ、呼び出しスタックウィンドウで中断点絵記号を含む行を右クリックし、その後に右クリックメニューでBreakpoints(中断点)からLocation(位置)をクリックしてください。

注: ソースウィンドウで中断点が設定された正確な文字を右クリックしなければなりません。これは中断点がソースコードの行内の特定文字上に設定された場合に必要です。

4.7.2.4. 中断点が何回的中かを把握する的中回数

実行は既定によって中断点に的中する毎に中断します。以下のように選ぶことができます。

- 常に中断 (既定)
- 的中回数が指定された値と等しい時に中断
- 的中回数が指定された値の倍数と等しい時に中断
- 的中回数が指定された値と等しいかより大きな時に中断

中断点に的中する回数を把握するけれども、決して実行を中断したくない場合、中断点が決して的中しないように的中回数を非常に大きな値に設定することができます。

指定した的中回数はデバッグ作業の間だけ保持されます。デバッグ作業終了時、的中回数は0にリセットされます。

4.7.2.5. 的中回数指定

1. 中断点ウィンドウで中断点を右クリックし、その後に右クリックメニューでHit Count(的中回数)をクリックしてください。

--または、--

ソース、逆アセンブリ、呼び出しスタックウィンドウで中断点絵記号を含む行を右クリックし、その後に右クリックメニュー上のBreakpoints(中断点)補助メニューからHit Count(的中回数)をクリックしてください。

2. Hit Count(的中回数)ダイアログ枠で中断点に的中した時の一覧から望む動作を選んでください。Break always(常に中断)以外のどれかを選ぶ場合、一覧の傍にテキスト枠が現れます。望む的中回数を設定するにはテキスト枠内に現れる整数を編集してください。

3. OKをクリックしてください。

4.7.2.6. 単一中断点の許可/禁止

ソース、逆アセンブリ、呼び出しスタックウィンドウで中断点絵記号を含む行を右クリックし、中断点を示し、その後にEnable Breakpoints(中断点許可)またはDisable Breakpoints(中断点禁止)をクリックしてください。

--または、--

Breakpoints(中断点)ウィンドウで中断点傍のチェック枠を選択または解除してください。

全中断点の許可/禁止

Debug(デバッグ)メニューからEnable All Breakpoints(全中断点許可)をクリックしてください。

4.7.2.7. 中断点削除

Breakpoints(中断点)ウィンドウで中断点を右クリックし、その後に右クリックメニューでDelete(削除)をクリックしてください。

--または、--

ソースウィンドウまたはDisassembly(逆アセンブリ)ウィンドウで中断点絵記号をクリックしてください。

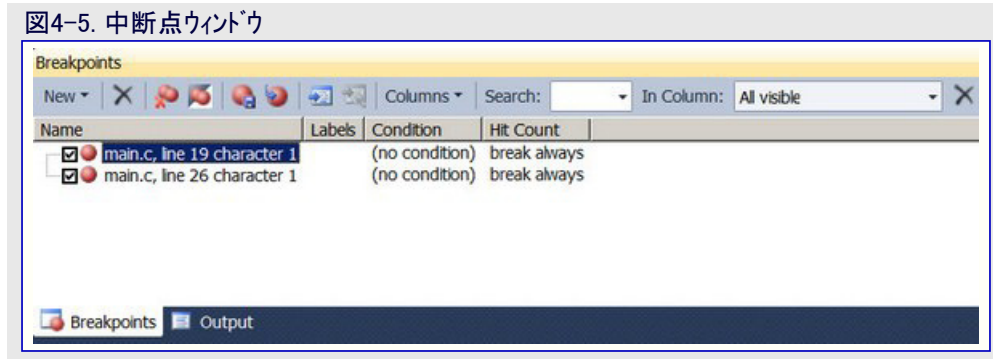
4.7.2.8. 全中断点削除

Debug(デバッグ)メニューからDelete All Breakpoints(全中断点削除)をクリックしてください。

確認指示

全中断点を削除すると、任意設定に依存して活動の確認を求める指示が現れるかもしれません。

4.7.3. 中断点ウインドウ



Debug(デバッグ)メニューからBreakpoints(中断点)ウインドウを開くことができます。

4.7.3.1. 中断点ウインドウを開く

Debug(デバッグ)メニューでWindows(ウインドウ)を示し、その後にBreakpoints(中断点)をクリックしてください。

4.7.3.2. 中断点の位置へ行く

Breakpoints(中断点)ウインドウで中断点をダブルクリックしてください。

--または、--

Breakpoints(中断点)ウインドウで中断点を右クリックしてGo To Source Code(ソースコードへ行く)またはGo To Disassembly(逆アセンブリへ行く)を選んでください。

--または、--

中断点をクリックし、その後にGo To Source Code(ソースコードへ行く)またはGo To Disassembly tool(逆アセンブリツールへ行く)をクリックしてください。

4.7.3.3. 追加の列を表示

Breakpoints(中断点)ウインドウの上部のツールバーでColumns(縦列)ツールをクリックし、その後に表示したい列名を選んでください。

4.7.3.4. 現在の検索基準に合う全中断点をエクスポート

Breakpoints(中断点)ウインドウのツールバーでExport all breakpoints matching current search criteria(現検索基準に合う全中断点をエクスポート)アイコンをクリックしてください。

1. Save As(名前を付けて保存)ダイアログ枠が現れます。
2. Save As(名前を付けて保存)ダイアログ枠でFile name(ファイル名)枠に名前を入力してください。
3. これは出力される中断点を含むXMLファイルの名前です。このダイアログ枠の上部で示されるフォルダパスに注意してください。違う場所にXMLファイルを保存するにはその枠で示されるフォルダパスを変更するか、または新しい場所を検索するためにBrowse Folders(フォルダ検索)をクリックしてください。
4. Save(保存)をクリックしてください。

4.7.3.5. 選んだ中断点をエクスポート

1. Breakpoints(中断点)ウインドウでエクスポートしたい中断点を選択してください。複数の中断点を選択するにはCtrlキーを押さえて続けて追加の中断点をクリックしてください。
2. 中断点一覧で右クリックしてExport selected(選択物をエクスポート)を選んでください。
3. Save As(名前を付けて保存)ダイアログ枠が現れます。
4. Save As(名前を付けて保存)ダイアログ枠でFile name(ファイル名)枠に名前を入力してください。これは出力される中断点を含むXMLファイルの名前です。
フォルダパスがダイアログ枠の上部で示されます。違う場所にXMLファイルを保存するにはその枠で示されるフォルダパスを変更するか、または新しい場所を検索するためにBrowse Folders(フォルダ検索)をクリックしてください。
5. Save(保存)をクリックしてください。

4.7.3.6. 中断点をインポート

1. Breakpoints(中断点)ウィンドウのツールバーでImport breakpoints from a file(ファイルから中断点を導入)アイコンをクリックしてください。
2. Open(開く)ダイアログ枠が現れます。
3. Open(開く)ダイアログ枠でファイルが置かれたディレクトリを検索し、その後にファイル名を入力するか、またはファイル一覧からファイルを選んでください。
4. OKをクリックしてください。

4.7.3.7. 指定した文字列に一致する中断点を見る

Search(検索)枠で以下の活動の1つを実行してください。

1. Search(検索)枠で検索文字列を入力してEnterを押してください。
--または、--
2. Search(検索)引き落とし一覧をクリックして以前の検索文字列一覧から文字列を選んでください。

指定した列に検索を制限するにはColumns(縦列)引き落とし一覧でクリックし、その後に検索したい列の名前をクリックしてください。

4.7.3.8. 検索後に全中断点を見る

Search(検索)枠で検索文字列を選んで削除し、その後にEnterを押してください。

4.7.3.9. 中断点ラベル

Microchip Studioでは中断点の経緯を保つのを助けるためにラベルを使うことができます。ラベルは中断点または中断点群に付随することができる名前です。ラベルに対して名前を作成するか、または既存ラベルの一覧から選ぶことができます。各中断点に対して複数のラベルを取り付けることができます。

ラベルは何らかの方法で関連される中断点の群を作りたい時に有用です。中断点にラベル付けした後、指定したラベルを持つ全ての中断点を見つけるために中断点ウィンドウでの検索機能に使うことができます。

検索機能はBreakpoints(中断点)ウィンドウで見ることができる情報の全ての列を検索します。検索に容易なラベルを作るため、別の列で現れる文字列と食い違ってもいいラベル名の使用を避けてください。例えば、"Program.c"と名付けたソースファイルを持つ場合、"Program.c"はそのファイルで設定したどの中断点の名前でも現れます。ラベル名として"Prog"を使う場合、Program.cで設定した全ての中断点は"Prog"と名付けた中断点を検索する時に現れます。

4.7.3.10. 中断点にラベル付け

1. Breakpoints(中断点)ウィンドウで1つまたは複数の中断点を選択してください。
2. 複数の中断点を選択するには中断点を選択する時にCtrlキーを使ってください。
3. 選択した中断点を右クリックし、その後にEdit labels(ラベル編集)をクリックしてください。
4. Edit breakpoint labels(中断点ラベル編集)ダイアログ枠が現れます。
5. Choose among existing labels(既存ラベルの中で選択)枠で1つまたは複数のラベルを選んでください。
--または、--
Type a new label(新しいラベルを入力)枠で新しいラベル名を入力し、その後にAdd(追加)をクリックしてください。

4.7.3.11. 指定したラベルを持つ中断点を検索

1. Breakpoints(中断点)ウィンドウのツールバーでIn Column(縦列内)枠をクリックして引き落とし一覧からLabels(ラベル)を選んでください。
2. Search(検索)枠で検索したいラベルの名前を入力してEnterを押してください。

4.7.3.12. 中断点からラベルを削除

1. Breakpoints(中断点)ウィンドウで1つまたは複数の中断点を選択してください。
2. 選択した中断点を右クリックし、その後にEdit labels(ラベル編集)をクリックしてください。
3. Edit breakpoint labels(中断点ラベル編集)ダイアログ枠が現れます。
4. Choose among existing labels(既存ラベルの中で選択)枠で選択した中断点から削除したいラベルに対するチェック枠を解除してください。

4.7.3.13. ラベルによって中断点一覧を並び替え

1. Breakpoints(中断点)ウィンドウで中断点一覧を右クリックしてください。
2. Sort by(によって並び替え)を指示し、その後にLabel(ラベル)をクリックしてください。

(任意選択) 並び順を変更するには再び中断点一覧を右クリックし、Sort by(によって並び替え)を指示し、その後にSort Ascending(昇順整列)またはSort Descending(降順整列)をクリックしてください。

4.8. データ中断点

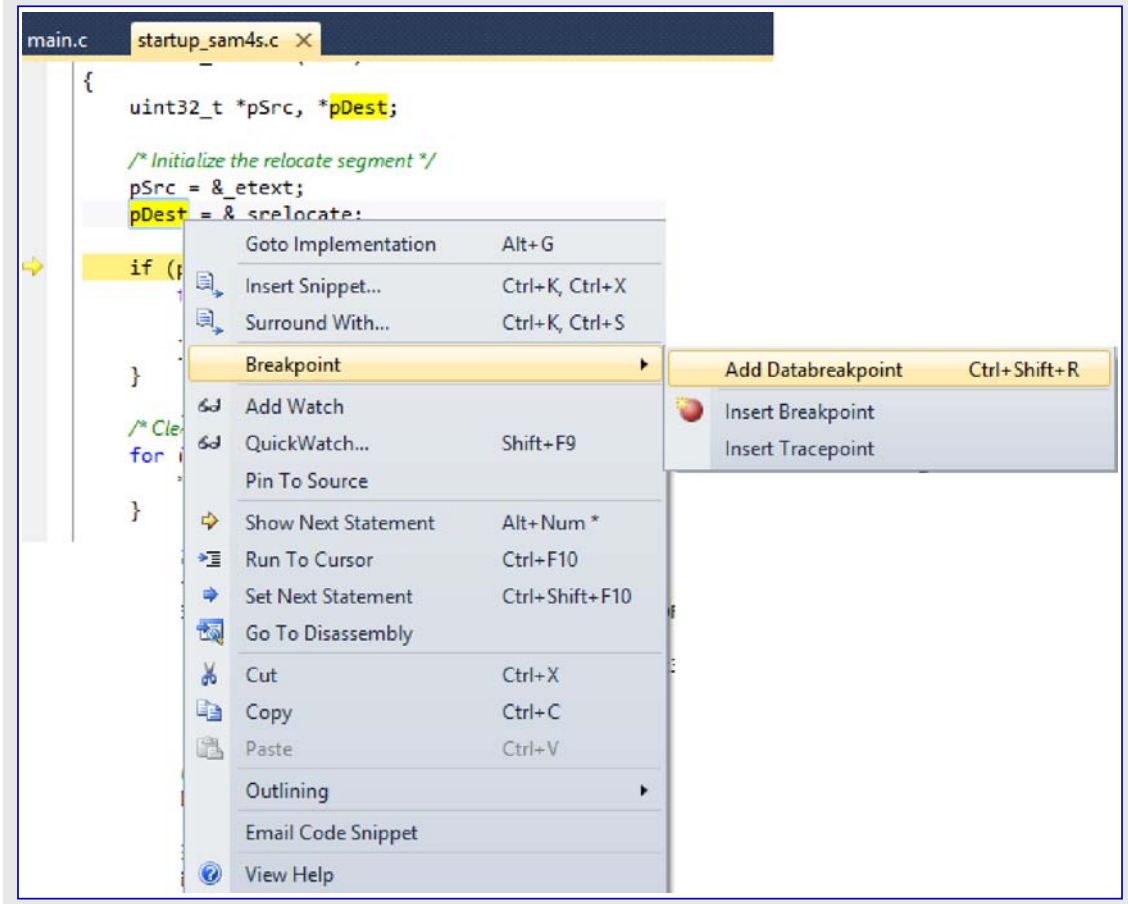
データ中断点は指定したメモリ位置に格納された値が(読み/書き)アクセスされる時に実行を中断することを許します。一般的に、データ中断点ハードウェア単位部はCPUとデータ キャッシュ間でデータ アドレスとデータ値の線を聴取し、アドレスと/または値が格納した比較値と一致する場合にCPUを停止することができます。プログラム中断点と異なり、データ中断点は中断点を引き起こした取得/格納命令が完了された後の次の命令で停止します。

4.8.1. データ中断点追加

コード エディタ状況メニューを用いるデータ中断点追加

コード エディタからデータ中断点を追加することができます。コード エディタで変数を選んで状況メニューからBreakpoint(中断点)⇒Add Databreakpoint(データ中断点追加)を選んでください。これは与えられた変数アドレスに対してデータ中断点を追加します。既定アクセス動作は書き込みで既定遮蔽はなしです。ショートカット キーのCtrl+Shift+Rを用いて同じ命令を呼び出すことができます。

図4-6. 状況メニューからのデータ中断点追加



デバッグ メニューからのデータ中断点追加

Debug(デバッグ)⇒New Breakpoint(新しい中断点)⇒New Data Breakpoint(新しいデータ中断点)から新しいデータ中断点を追加することができます、これはData Breakpoint Configuration(データ中断点構成設定)ウィンドウを開きます。

データ中断点ツール ウィンドウからのデータ中断点追加

データ中断点ツール ウィンドウのNew(新規)鈕を用いて新しいデータ中断点を追加することができます、これはData Breakpoint Configuration(データ中断点構成設定)ウィンドウを開きます。

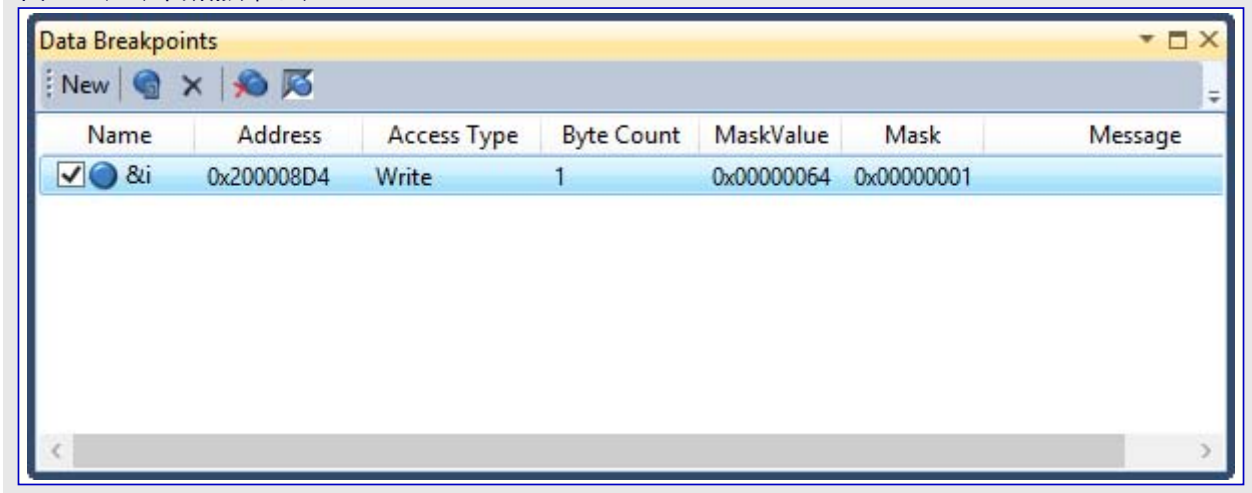
注: デバッグ動作でだけデータ中断点を追加または変更することができます。

4.8.2. データ中断点ウィンドウ





4.8.2.1. データ中断点ツール ウィンドウ

データ中断点ウィンドウはデータ中断点を追加して追加したデータ中断点を一覧にするための任意選択を提供します。Debug(デバッグ)⇒Windows(ウィンドウ)⇒Data Breakpoints(データ中断点)を選ぶことによってこのウィンドウを許可してください。

図4-7. データ中断点ウィンドウ



データ中断点ツールバーは以下の要素を持ちます。







- **New** – 新しいデータ中断点を追加するためにデータ中断点構成設定ウィンドウを提供します。
-  – 選択したデータ中断点を編集するためにデータ中断点構成設定ウィンドウを提供します。
-  – 選択したデータ中断点を削除します。Deleteキーを用いて同じ命令を呼び出すことができます。
-  – 全てのデータ中断点を削除します。
-  – 全てのデータ中断点を許可または禁止します。

データ中断点ウィンドウは中断点構成設定に関連する様々な列を表示します。列のいくつかは中断点構成設定に基づいて動的に隠されます。例えば、中断点が構成設定された遮蔽を全く持たない場合、列に関連する遮蔽は表示されません。

Name(名前)列は以下の様な3つの部分を持ちます。

- チェック枠 – 中断点を許可または禁止するのにチェック枠を使ってください。
- アイコン – 中断点の現在状況を表すための絵記号。下表はこれらの絵記号を記述します。中断点絵記号上にマウスを留める場合、より多くの情報と共に中断点情報が現れます。この情報は異常または警告の中断点に対して特に有用です。
- テキスト – 中断点に対して構成設定された位置式を表示します。

表4-2. 中断点アイコン

アイコン	説明
	標準中断点。一様の絵記号は中断点が許可されることを示します。中空の絵記号は禁止されることを示します。
	高度な中断点。活性/禁止。+符号は中断点がそれに付随された的中回数を持つことを示します。
	追跡点。活性/禁止。この点への的中は指定した活動を実行しますが、プログラム実行を中断しません。
	高度な追跡点。活性/禁止。+符号は追跡点がそれに付随された的中回数を持つことを示します。
	中断点または追跡点の異常。×は中断点または追跡点が異常状態のために設定することができないことを示します。異常メッセージでより多くの詳細についてはメッセージ列を調べてください。
	中断点または追跡点が設定されますが警告付きです。警告メッセージでより多くの詳細についてはメッセージ列を調べてください。

4.8.2.2. megaAVR[®]用データ中断点構成設定ウィンドウ

このウィンドウはATmegaデバイス用のデータ中断点に関連する構成設定任意選択を提供します。アドレス遮蔽は任意選択です。

Location (位置)

RAM内の特定アドレスを直接(例えば:0x8004)、またはRAM内のアドレスの値を求める式(例えば:&x)を入力してください。入力した式が監視するデータのアドレスを示すことを確実にしてください。

注: 局所変数上のデータ中断点はスタックメモリの再利用のため誤った的に帰着し得ます。デバッグ目的のためにstaticとしてそれを宣言することを推奨します。

Access Mode (アクセス動作)

特定のアクセス動作で中断するように中断点を構成設定してください。3つのアクセス動作形式が支援されます。

- **Read**(読み込み) – プログラムは指定した位置での読み込みで中断します。
- **Write**(書き込み:既定) – プログラムは指定した位置での書き込みで中断します。
- **Read/Write**(読み書き) – プログラムは指定した位置での読み込みまたは書き込みで中断します。

Address Mask (アドレス遮蔽)

megaのデータ中断点でのアドレス遮蔽は任意選択です。1つのアドレスよりも多く、または特定のアクセスでのアドレスの範囲で中断するのにアドレス遮蔽を使ってください。

Mask 1つよりも多いアドレスまたはアドレスの範囲を定義するためにLocation(位置)アドレスを遮蔽するための遮蔽値。遮蔽で1の値を持つビットは意味を持つビットで、0はいつでもよいビットです。

一般的に与えられたアドレス(A)と遮蔽(M)について、以下の時にアドレス(B)が成功裏に一致します。

$(A) \& (M) = (B) \& (M)$ 、ここでAはLocation(位置)で入力された式に対して解決されたアドレス、MはMask(遮蔽)で入力された遮蔽値、BはRAM内の何れかのアドレスです。

Masked Addr これはプログラムが中断し得るアドレス一致の範囲を示す読み込み専用領域です。記されたアドレスは簡潔性のために2進形式で示されます。'X'はいつでもよいビットを表す一方で、残りのビットが一致を意図されます。

例えば、0b00000010XX000XXはXビットを除いてこの文字列のような全ビットを持つアドレスでアクセス動作のとおり中断し得ます。この場合、下位側のビット0,1,5,6はこれらのビットがいつでもよい(X)ため何でも有り得ます。

注: ATmegaデバイスはデータ遮蔽を支援しません。

4.8.2.3. XMEGA[®]用データ中断点構成設定ウィンドウ

このウィンドウはXMEGAデバイス用のデータ中断点に関連する構成設定任意選択を提供します。

Location (位置)

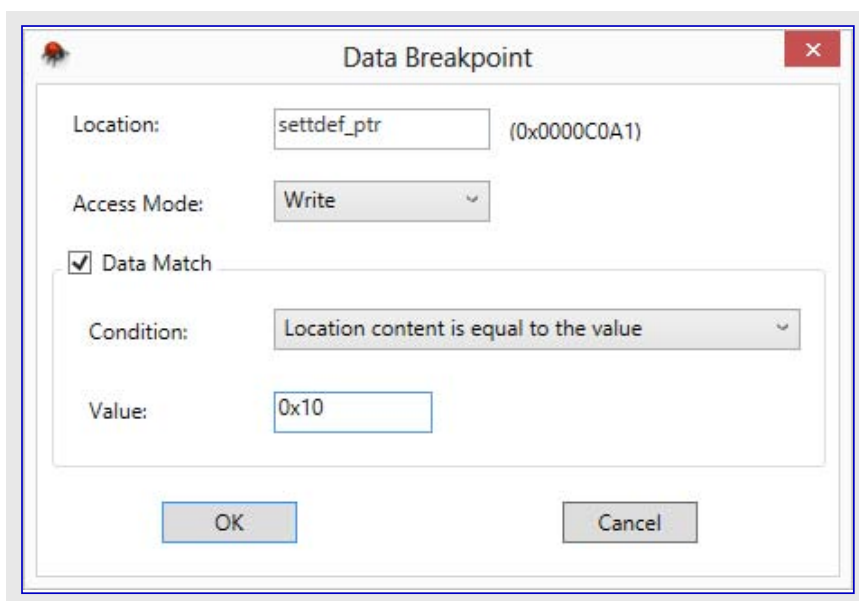
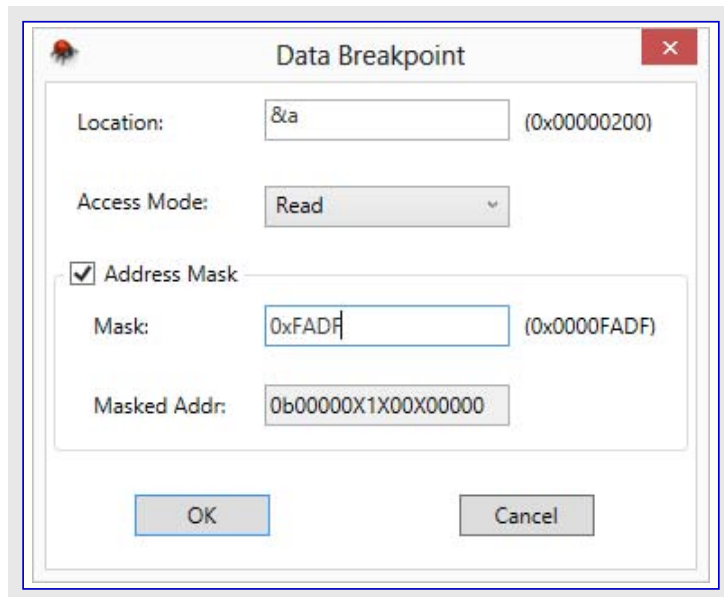
RAM内の特定アドレスを直接(例えば:0x8004)、またはRAM内のアドレスの値を求める式(例えば:&x)を入力してください。入力した式が監視するデータのアドレスを示すことを確実にしてください。

注: 局所変数上のデータ中断点はスタックメモリの再利用のため誤った的に帰着し得ます。デバッグ目的のためにstaticとしてそれを宣言することを提案します。

Access Mode (アクセス動作)

特定のアクセス動作で中断するように中断点を構成設定してください。3つのアクセス動作形式が支援されます。

- **Read**(読み込み) – プログラムは指定した位置での読み込みで中断します。
- **Write**(書き込み:既定) – プログラムは指定した位置での書き込みで中断します。
- **Read/Write**(読み書き) – プログラムは指定した位置での読み込みまたは書き込みで中断します。



Data Match (データ一致)

以下の条件の1つに基づいて指定した位置でデータを比較するにはデータ中断点を構成設定するためにデータ一致任意選択を使ってください。

- Location content is equal to the value (位置内容が値に等しい)
- Location content is greater than the value (位置内容が値よりも大きい)
- Location content is less than or equal to the value (位置内容が値と等しいか、またはより小さい)
- Location content is within the range (位置内容が範囲内)
- Location content is outside the range (位置内容が範囲外)
- Bits of the location is equal to the value (位置のビットが値に等しい)

注: ビット遮蔽: '1'を持つビットは意味を持ち、'0'を持つビットはいつでもよい、8ビット値です。

一般的に与えられた値(V)とビット遮蔽(M)について、位置領域(VL)の値が次の条件を満足する時に中断事象が起動されます。

$$(A) \& (M) = (VL) \& (M)$$

例えば、値=xA0とビット遮蔽=0xF0の使用は位置領域が0xA0~0xAF間の値のどれかを持つ時に中断事象を起動します。

注: Value(条件値)領域は1バイトでなければなりません。

4.8.2.4. UC3用データ中断点構成設定ウィンドウ

このウィンドウはUC3デバイス用のデータ中断点に関連する構成設定任意選択を提供します。

Location (位置)

RAM内の特定アドレスを直接(例えば:0x8004)、またはRAM内のアドレスの値を求める式(例えば:&x)を入力してください。入力した式が監視するデータのアドレスを示すことを確実にしてください。

注: 局所変数上のデータ中断点はスタックメモリの再利用のため誤った中に帰着し得ます。デバッグ目的のためにstaticとしてそれを宣言することを提案します。

Access Mode (アクセス動作)

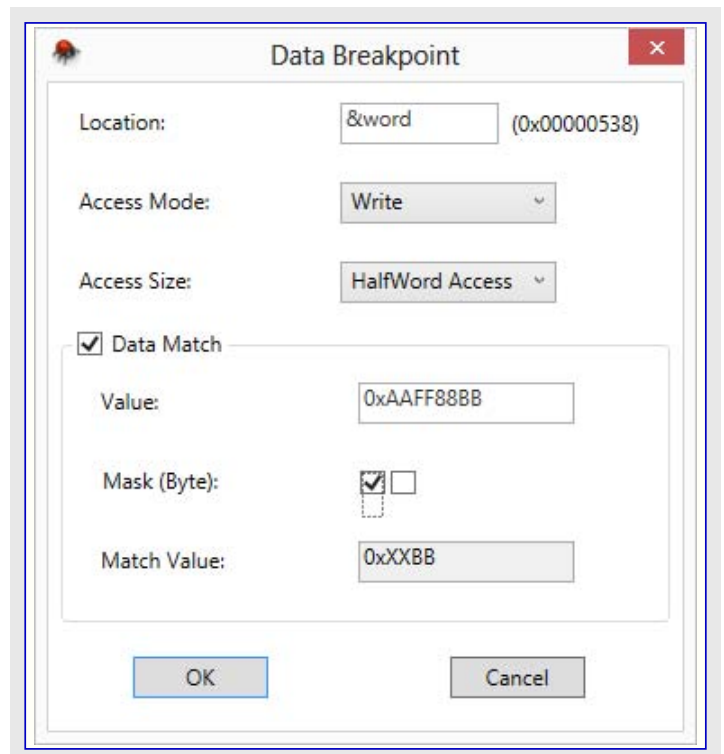
特定のアクセス動作で中断するように中断点を構成設定してください。3つのアクセス動作形式が支援されます。

- Read(読み込み) - プログラムは指定した位置での読み込みで中断します。
- Write(書き込み:既定) - プログラムは指定した位置での書き込みで中断します。
- Read/Write(読み書き) - プログラムは指定した位置での読み込みまたは書き込みで中断します。

Access Size (アクセス幅)

特定のアクセス幅で中断するように中断点を構成設定してください。4つのアクセス幅形式が支援されます。

- Any Access (全アクセス:既定) - プログラムは指定された位置での全アクセスで中断します。
- Byte Access (バイトアクセス) - プログラムは指定された位置でのバイトアクセスで中断します。
- HalfWord Access (半語アクセス) - プログラムは指定された位置での半語アクセスで中断します。
- Word Access (語アクセス) - プログラムは指定された位置での語アクセスで中断します。



例4-1. アクセス幅設定例

コード

```

1: int word = 0;
2: short *halfWord = (short*)&word;
3:
4: int main(void)
5: {
6:     word = 0xAABBCCFF;
7:     *halfWord = 0xDDEE;
8: }

```

上のコードと右の構成設定に対して、プログラムは半語(HalfWord)アクセス後の8行目で中断します。

構成設定

Data Match (データ一致)

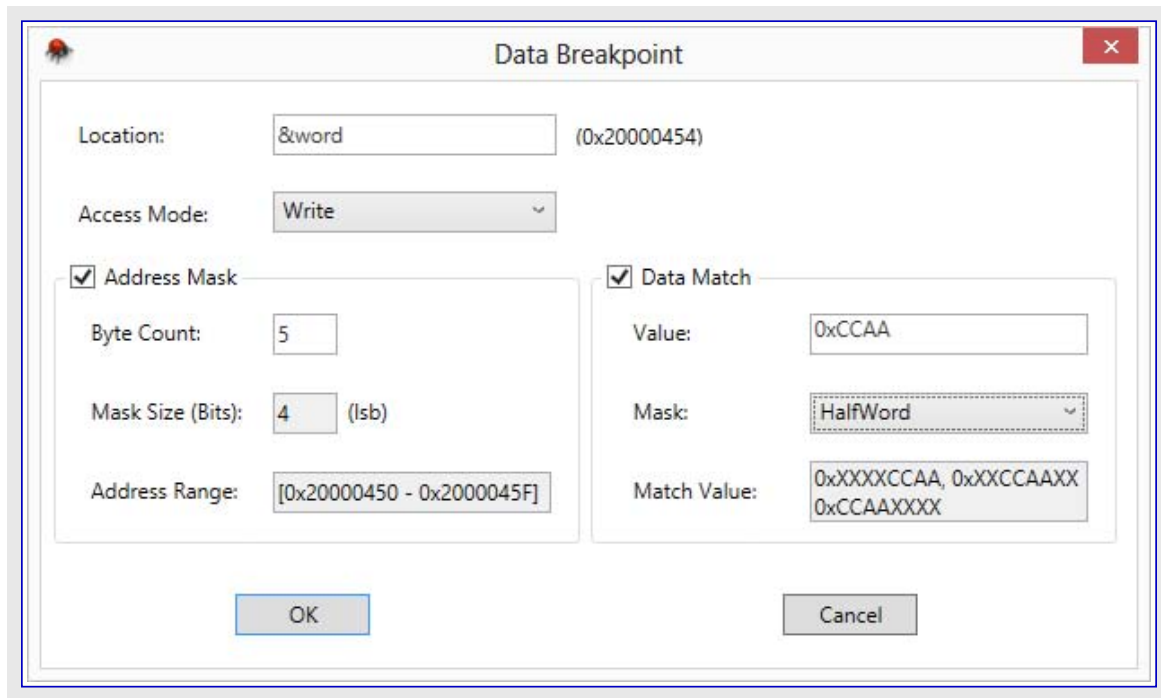
32ビット値を持つ指定した位置でデータを比較するようにデータ中断点を構成設定するにはData Match(データ一致)任意選択を使ってください。中断点事象は一致成功によって起動されます。

Value Location(位置)アドレスでデータと比較するための32ビット(4バイト)値。この値は10進数または16進数(例えば:100または0x64)にすることができます。Access Size(アクセス幅)に基づいて、各々のバイトがデータ比較に使われます。例えば、Access Size(アクセス幅)として”HalfWord(半語)”を選び、Value(値)として0xAABBCCDDを入力した場合、それはデータ比較に最後の2バイト(0xCCDD)が使われます。更に、Mask(遮蔽)を指定することによってValue(値)を精細化することができます。

Mask (Byte) 各チェック枠はValue(値)領域内の各々のバイトの意義を制御します。Value(値)領域内の特定バイトを遮蔽するには適切なチェック枠を選んでください。表示されるチェック枠数はAccess Size(アクセス幅)に基づいて決められます。”Any Access(全アクセス)”または”Word Access(語アクセス)”に対しては(バイト毎に)4つのチェック枠が、”HalfWord Access(半語アクセス)”に対しては2つのチェック枠が、”Byte Access(バイトアクセス)”に対しては1つのチェック枠が表示されます。

Match Value Access Size(アクセス幅)、Value(値)、Mask (Byte)(遮蔽(バイト))の領域に基づいて遮蔽された値を表示する読み出し専用領域です。遮蔽されたバイトはそのバイトがデータ比較で意味を持たないことを意味する’XX’として表されます。

4.8.2.5. SAM用データ中断点構成設定ウィンドウ



SAMデバイス用構成設定任意選択を持つデータ中断点ウィンドウ

Location (位置)

RAM内の特定アドレスを直接(例えば:0x8004)、またはRAM内のアドレスの値を求める式(例えば:&x)を入力してください。入力した式が監視するデータのアドレスを示すことを確実にしてください。

注: 局所変数上のデータ中断点はスタックメモリの再利用のため誤った中に帰着し得ます。デバッグ目的のためにstaticとしてそれを宣言することを提案します。

Access Mode (アクセス動作)

特定のアクセス動作で中断するように中断点を構成設定してください。3つのアクセス動作形式が支援されます。

- **Read**(読み込み) – プログラムは指定した位置での読み込みで中断します。
- **Write**(書き込み:既定) – プログラムは指定した位置での書き込みで中断します。
- **Read/Write**(読み書き) – プログラムは指定した位置での読み込みまたは書き込みで中断します。

Address Mask (アドレス遮蔽)

アクセスを監視するためのアドレスの範囲を定義するのにこの設定を使ってください。

Byte Count **Location**(位置)アドレスで始まる監視のためのアドレス位置数を入力してください。実際に監視されるアドレスの範囲は意図する範囲よりも広くなり得ます。例えば、**Location**(位置)が0x23FAで**Byte Count**(バイト数)が5の場合、実際に監視されるアドレスの範囲は0x23F8~0x23FFです。意図する範囲の0x23FA~0x23FA+5を網羅するために、実際の範囲は**Location**(位置)アドレスで遮蔽されなければならない下位側ビット数を計算することによって算定されます。この場合、遮蔽されるべきビット数は3です。結果として実際の範囲0x23F8~0x23FFは意図する範囲の0x23FA~0x23FFよりも広くなります。

Mask Size これは**Location**(位置)アドレスで遮蔽される下位側ビット数を表示する読み込み専用領域です。**Mask Size**(遮蔽幅)は**Byte Count**(バイト数)と**Location**(位置)に基づいて計算されます。

Address Range これは実際にアクセスを監視されるアドレスの範囲を表示する読み込み専用領域です。範囲は最小と最大の両アドレスを含む閉区間です。

Data Match (データ一致)

32ビット値を持つ指定した位置でデータを比較するようにデータ中断点を構成設定するには**Data Match**(データ一致)任意選択を使ってください。中断点事象は一致成功によって起動されます。

Value これは**Location**(位置)アドレスでデータと比較するための32ビット(4バイト)値です。この値は10進数または16進数(例えば:100または0x64)にすることができます。**Mask**(遮蔽)を指定することによって**Value**(値)を更に精細化することができます。

Mask データ比較に使うために**Value**(値)から適切なバイトを抽出するのに**Mask**(遮蔽)を使ってください。例えば、**Value**(値)が0xAABBCCDDで**Mask**(遮蔽)がHalfWordの場合、**Value**(値)から最後の2バイト(0xCCDD)が抽出されてデータ比較に使われます。これはデータ比較がここでのXがどうでもよい16進数(0~F)である以下の一致、0xFFFFCCDD、0xFFCCDDXX、0xCCDDXXXXに対して成功することを意味します。以下の3つの**Mask**(遮蔽)の形式が支援されます。

- **Byte** (バイト) – データ比較に**Value**(値)から抽出された最後のバイトを使います。
- **HalfWord** (半語) – データ比較に**Value**(値)から抽出された最後の2バイトを使います。
- **Word** (語:既定) – データ比較に**Value**(値)からの全4バイトを使います。

Match Value これは一致値を表示する読み込み専用領域です。

4.8.2.5.1. データ一致例

例4-2. バイト一致例

- **Location**(位置) = 0x200008D4
- **Value**(値) = 0x0000CCAA
- **Mask**(遮蔽) = Byte

プログラムは位置0x200008D4が以下の値のどれかを持つ時に中断します。

- 0XXXXXXXXAA
- 0XXXXAAXX
- 0XXAAAXXX
- 0AAAXXXXX

例4-3. 半語一致例

- **Location**(位置) = 0x200008D4
- **Value**(値) = 0x0000CCAA
- **Mask**(遮蔽) = HalfWord

プログラムは位置0x200008D4が以下の値のどれかを持つ時に中断します。

- 0XXXXCCAA
- 0XXCCAAXX
- 0CCAAXXXX

例4-4. 語一致例

- **Location**(位置) = 0x200008D4
- **Value**(値) = 0x0000CCAA
- **Mask**(遮蔽) = Word

プログラムは位置0x200008D4が以下の値を持つ時に中断します。

- 0XXXXCCAA

4.8.2.6. シミュレータ ツール用データ中断点構成設定ウィンドウ

右の構成設定ウィンドウはシミュレータ ツールが選ばれる時に何れかのデータの基本構造に対して表示されます。

Location (位置)

RAM内の特定アドレスを直接(例えば:0x8004)、またはRAM内のアドレスの値を求める式(例えば:&x)を入力してください。入力した式が監視するデータのアドレスを示すことを確実にしてください。

注: 局所変数上のデータ中断点はスタック メモリの再利用のため誤った中に帰着し得ます。デバッグ目的のためにstaticとしてそれを宣言することを提案します。

Access Mode (アクセス動作)

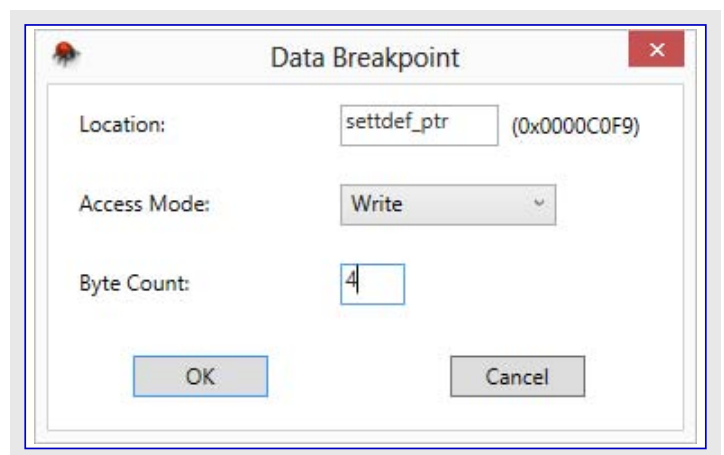
特定のアクセス動作で中断するように中断点を構成設定してください。3つのアクセス動作形式が支援されます。

- **Read**(読み込み) – プログラムは指定した位置での読み込みで中断します。
- **Write**(書き込み:既定) – プログラムは指定した位置での書き込みで中断します。
- **Read/Write**(読み書き) – プログラムは指定した位置での読み込みまたは書き込みで中断します。

Byte Count (バイト数)

Location(位置)アドレスで始まる監視にアドレス位置数を入力してください。

注: シミュレータは無制限数のデータ中断点を支援します。



4.8.2.7. データ中断点的中回数指定方法

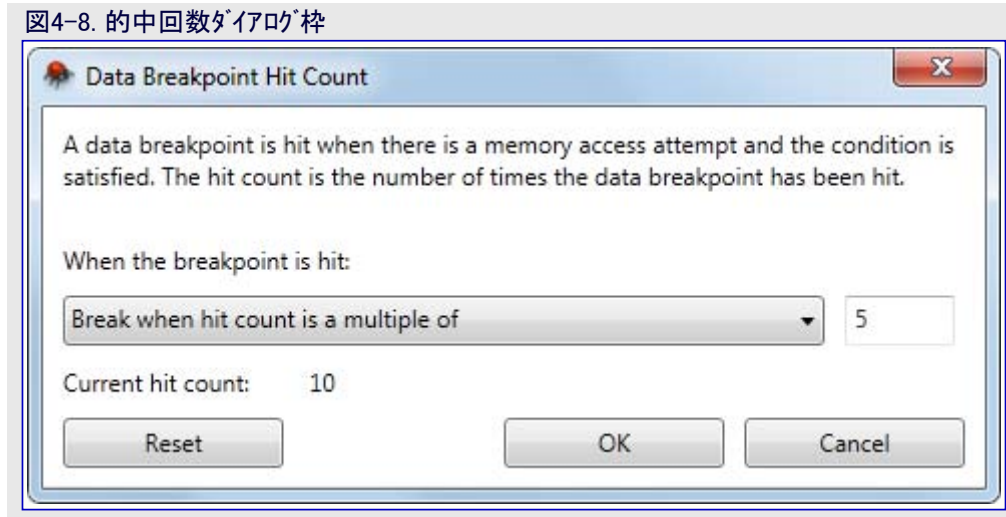
Hit Count (的中回数)

指定したメモリ位置に格納された値がアクセス(読み/書き)される回数。

的中回数を指定

的中回数属性を指定または編集するにはHit Count(的中回数)ダイアログ枠を開かなければなりません。

Data Breakpoints(データ中断点)ウィンドウで中断点行を選択し、その後に状況メニューでHit Count(的中回数)を選んでください。



的中回数属性を設定または編集するには以下の手段を使ってください。

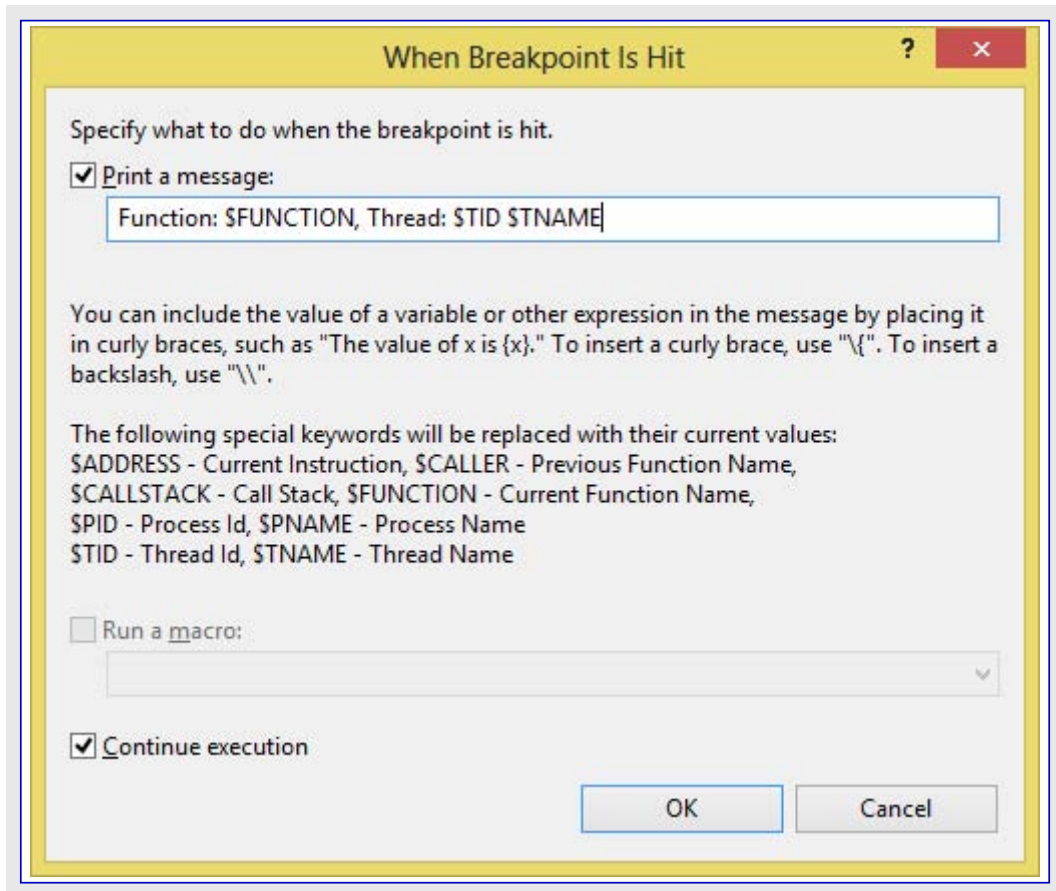
- **When the breakpoint is hit:**(中断点に当たった時に) : この設定はこれが当たった時に中断点がどう動くべきかを決めます。以下を選ぶことができます。
 - Break always (常に中断:既定)
 - Break when the hit count equals a specified value (的中回数が指定した値と等しい時に中断)
 - Break when the hit count equals a multiple of a specified value (的中回数が指定した値の倍数と等しい時に中断)
 - Break when the hit count is greater or equals to a specified value (的中回数が指定した値と等しいかより大きい時に中断)
- **Current hit count:**(現在の的中回数) : この値はデータ中断点が当たった回数を示します。変数に対するデータの読み/書きは多数の命令に変換され、様々なメモリアクセスに帰着します。故にデータ中断点は同じ変数に対して多数回的中、的中回数はそれに応じて更新されます。
- **Reset**(リセット) : この鈕はCurrent hit count:(現在の的中回数)で示される値を0にリセットします。

When the breakpoint is hit:(中断点に的中した時に)引き落とし一覧で既定以外のどれかの任意選択を選ぶ場合、その次に編集枠が現れます。的中回数値を設定するためにその編集枠で値を編集してください。例えば、Break when the hit count equals a specified value(的中回数が指定した値と等しい時に中断)を選んで5を入力してください。これは中断点が他の的中ではなく5回目の的中で実行を停止させます。

4.8.2.8. When Breakpoint is Hit(中断点的中時)ダイアログ枠

このダイアログ枠でデータ中断点が当たった時に出力ウィンドウでメッセージを出力することができます。

When Breakpoint is Hit(中断点的中時)ダイアログ枠を開くにはData Breakpoints(データ中断点)ウィンドウへ行き、中断点行を選択し、その後に状況メニューからWhen Hit(的中時)を選んでください。



メッセージ指定


- When Breakpoint is Hit(中断点的中時)ダイアログ枠で記述されるキーワードのどれかを使ってください。例えば、プロセスID: \$PID。
- `sum={a+b}`のように中括弧内にそれを配置することによってメッセージ内に式を指定してください。

中断点の動きを指定

中断点に当たった時に実行を中断するにはContinue execution(実行継続)チェック枠を解除してください。Continue execution(実行継続)がチェックされると、実行は停止されません。両方の場合でメッセージは出力されます。

4.8.3. データ中断点の一般的情報

- データ中断点はデバッグ動作でだけ追加/編集することができます。
- 局所変数は常に関数名で修飾され、これはプログラムがそこで停止している関数から変数を追加したい場合にもです。局所変数上のデータ中断点はスタックメモリを再利用するために誤った助言に終わり得ます。

 **助言:** 局所変数を静的(static)として宣言してLocation(位置)領域で静的変数のアドレスを提供してください。静的変数のアドレスはコンパイル/リンクの間に定められます。

- 全域変数は始動中に既定値で初期化されます。全域変数に対して有効なデータ中断点は始動中の逆アセンブリまたは初期化コードで的中します。
- 変数に対するデータの読み書きを実行する様々な命令が有り得ます。例えば'int'データ型は2つの個別バイト読み/書き命令を持つことができ、故にデータ中断点は同じ変数に対して2度的中します。
- データ中断点事象はバスアクセスが指定アドレスに対して起こる時に発生することができます。
- 支援されるデータ中断点の最大数 (これは指定するデバイス/システムに基づいて変わり得ます。データシートを参照してください。) [次表参照](#)

基本構造	支援される最大データ中断点
megaAVR	<ul style="list-style-type: none"> データ遮蔽なしの2つ (または) データ遮蔽なしの1つとデータ遮蔽付きの1つ
XMEGA	<ul style="list-style-type: none"> データ遮蔽なしの2つ (または) データ遮蔽なしの1つとデータ遮蔽付きの1つ (または) データ遮蔽付きの2つ
UC3	<ul style="list-style-type: none"> データ遮蔽なしの2つ (または) データ遮蔽なしの1つとデータ遮蔽付きの1つ (または) データ遮蔽付きの2つ
SAM	デバイス依存、データシート参照
tinyAVR	データ中断点を支援しません。

デバイスの殆どは上の制限に一致します。

注: megaAVRとSAMのデバイスはデータ遮蔽付きデータ中断点が設定される時に複数のハードウェア資源を使います。故に、データ遮蔽の使用は設定することができるデータ中断点数を減らし得ます。

4.8.4. データ中断点の使い方

4.8.4.1. データ中断点を用いるスタック溢れ検出

応用に対する最大スタック量を決めてスタックの適切な最終アドレスを計算してください。

最終アドレスでデータ中断点を適用するアドレス遮蔽によるアドレス範囲とRead/Write(読み/書き)としてのアクセス形式に設定してください。

注: 上の方法はヒープメモリが指定したスタック最終アドレスをアクセスしようとする時に誤った中断を引き起こすかもしれません。

4.9. 一瞬監視、監視、局所、自動のウィンドウ

Microchip Studioのデバッガはデバッグ中に様々な情報を表示するために、変数ウィンドウとして総称される様々なウィンドウを提供します。各変数ウィンドウはName(名前)、Value(値)、Type(型)の3つの列を持つ格子を持ちます。Name(名前)列はAutos(自動)とLocals(局所)のウィンドウで自動的に追加される変数の名前を示します。

Watch(監視)ウィンドウで、Name(名前)列はあなたの変数や式を追加することができる場所です。デバッガでの式の監視方法をご覧ください。

Value(値)とType(型)の列は対応する変数や式の結果の値とデータ型を表示します。

Value(値)列で変数の値を編集することができます。

Autos(自動)、Locals(局所)、Watch(監視)の変数ウィンドウはデバッグ作業中に或る変数の値を表示します。QuickWatch(一瞬監視)ダイアログ枠も変数を表示することができます。デバッガの中断動作の時に、それらの位置で現れる殆どの変数の値を編集するのにこの変数ウィンドウを使うことができます。

注: 浮動小数点値の編集は小数部分の10進⇒2進変換のため僅かな不正確さに帰着し得ます。表面上は害のない編集でさえ、浮動小数点変数で下位側のいくつかのビットへの変更に帰着し得ます。

Watch(監視)ウィンドウで式が評価される時に更新(Refresh)アイコンを見るかもしれず、これは異常または古い値を示します。

必要とされるなら、値に対して式を入力することができます。デバッガは式を評価してそれを結果値で置き換えます。デバッガはWatch(監視)ウィンドウで殆どの有効な言語表現を受け入れます。より多くの情報については「4.9.5. 表現形式」をご覧ください。

本来のコードでプログラミングする場合、変数名または変数名を含む式の属性を時々修飾しなければなりません。属性は変数が置かれる関数、ソースファイル、単位部を意味します。これを行わなければならない場合、属性演算子構文を使うことができます。

いくつかの式の評価は変数の値を変更することがあり、また、さもなければプログラムの状態に影響を及ぼします。例えば、以下の式の評価はvar1とvar2の値を変更します。

```
var1 = var2++
var1 = var2++
```

データを変更する式は副作用を持つと言われ、それらに気付かない場合に予期せぬ結果を生成することがあります。従って、それを実行する前に式の影響の理解を確実にしてください。

変数ウィンドウでの変数編集

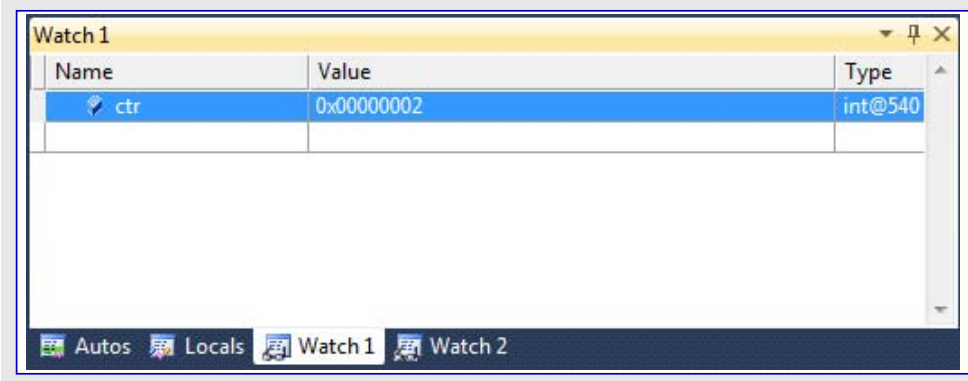
1. デバッガは中断動作でなければなりません。
2. 変数が配列またはオブジェクトの場合、樹状制御部がName(名前)列の傍に現れます。Name(名前)列で編集したい値の要素を見つけるために必要ならば変数を広げてください。
3. 変更する列のValue(値)をダブルクリックしてください。
4. 新しい値を入力してください。
5. Enterを押してください。

変数ウィンドウを表示

Debug(デバッグ)メニューでWindows(ウィンドウ)を選び、その後に表示したい変数ウィンドウの名前(Autos(自動)、Locals(局所)、Watch(監視)、Watch1(監視1)~Watch4(監視4))を選んでください。

設計動作ではこれらのメニュー項目にアクセスしたり、これらのウィンドウを表示することができません。これらのメニュー項目を表示するにはデバッグが走行しているかまたは中断動作でなければなりません。

4.9.1. 監視ウィンドウ



Watch(監視)ウィンドウとQuickWatch(一瞬監視)ダイアログ枠は変数名とデバッグ作業中に監視したい式を入力できる場所です。

QuickWatch(一瞬監視)ダイアログ枠は一度に1つの変数または式を調べることが許されます。これは1つの値やより大きなデータ構造体をちょっと見てみるのに有用です。Watch(監視)ウィンドウはデバッグ作業間に見たい少数の変数と式を格納することができます。Microchip StudioはWatch1(監視1)~Watch4(監視4)と番号付けされた複数の監視ウィンドウを持ちます。

変数名は入力する最も簡単な式です。本来のコードをデバッグしている場合、変数名だけでなくレジスタ名も使うことができます。けれども、デバッグはそれよりも更に複雑な式を受け入れることができます。例えば、3つの変数の平均値を見つけるのに以下の式を入力することができます。

```
(var1 + var2 + var3) / 3
```

デバッグはWatch(監視)ウィンドウで最も有効な言語表現を受け入れます。より多くの情報については「4.9.5. 表現形式」をご覧ください。本来のコードでプログラミングする場合、変数名または変数名を含む式の属性を時々修飾しなければなりません。属性は変数が置かれる関数、ソースファイル、単位部を意味します。これを行わなければならない場合、属性演算子構文を使うことができます。

プログラムに影響を及ぼす式

いくつかの式の評価は変数の値を変更することがあり、また、さもなければプログラムの状態に影響を及ぼします。例えば、以下の式の評価はvar1の値を変更します。

```
var1 = var2
```

データを変更する式は副作用を持つと言われます。Watch(監視)ウィンドウに副作用を持つ式を入力する場合、その副作用は式がWatch(監視)ウィンドウによって評価される時毎に起こるでしょう。副作用を持つ式に気付かない場合、これは予期せぬ結果を生成することがあります。副作用を持つことが分かる式はそれを最初に入力する時に1度だけ評価されます。続いて起こる評価は禁止されます。値の傍に現れる更新アイコンをクリックすることによって手動でこの動きを無効にすることができます。

予期せぬ副作用は度重なる関数評価の結果です。例えば、Watch(監視)ウィンドウに以下の関数呼び出しを入力することができます。

```
PrintFunc1(var1)
Func1(var1)
```

Watch(監視)ウィンドウまたはQuickWatch(一瞬監視)ダイアログから関数を呼ぶ場合、呼ぶ関数はデータを変更するかもしれず、副作用を生じます。関数評価からの起こり得る予期せぬ副作用を避ける1つの方法はOptions(任意選択)ダイアログ枠で自動関数評価をOFFにすることです。これはプロパティのようなより新しい言語機能の自動的な評価を禁止します。けれども、それはより安全です。

注: Watch(監視)ウィンドウで式を調べる時に緑の円内で違う方向で回る2つの緑矢印に似ている更新アイコンを見るかもしれません。これは特に自動関数評価をOFFにした場合にありそうです。この更新アイコンは異常または古い値を示します。

Microchip Studioデバッグはそれらの最も重要な要素を見るために共通データ型を自動的に拡張します。独自データ型に対する拡張を追加することもできます。

注: あなたが見るダイアログ枠とメニュー指令はあなたの有効な設定や改訂版に依存してヘルプで記述されるそれらと違うかもしれません。設定を変更するにはTools(ツール)メニューでImport and Export Settings(設定のインポートとエクスポート)を選んでください。より多くの情報については「10. メニューと設定」をご覧ください。

監視ウィンドウで式を評価

1. **Watch**(監視)ウィンドウに於いて**Name**(名前)列で空の行をクリックしてください。デバッガはこの時点で中断動作でなければなりません。監視したい変数名または式を入力または貼り付けしてください。
--または、--
Watch(監視)ウィンドウ内の行に変数をドラッグしてください。
2. **Enter**を押してください。
3. 結果が**Value**(値)列に現れます。配列またはオブジェクト変数の名前を入力した場合、**Name**(名前)列の名前の次に樹状表示部が現れます。**Name**(名前)列で変数を展開または格納してください。
4. 式はあなたがそれを取り去るまで**Watch**(監視)ウィンドウに留まります。

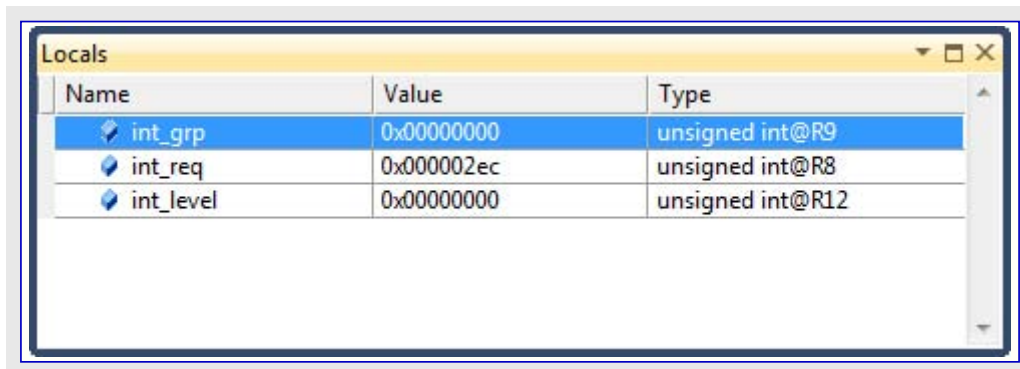
一瞬監視ウィンドウで式を評価

1. **QuickWatch**(一瞬監視)ダイアログ枠で**Expression**(式)テキスト枠に変数、レジスタ、または式を入力または貼り付けしてください。
2. **Reevaluate**(再評価)をクリックするか、または**Enter**を押してください。
3. **Current**(現在)値枠に値が現れます。
4. 配列またはオブジェクト変数の名前を入力した場合、**Current**(現在)値枠の名前の次に樹状表示部が現れます。**Name**(名前)列で変数を展開または格納してください。

一瞬監視ウィンドウで以前の式を評価

1. **QuickWatch**(一瞬監視)ダイアログ枠で**Expression**(式)テキスト枠の右側に現れる下向き矢印をクリックしてください。
2. 引き落とし一覧から以前の式を選んでください。
3. **Reevaluate**(再評価)をクリックください。

4.9.2. 局所ウィンドウ



Locals(局所)ウィンドウは現在の状況に対して局所の変数を表示します。

4.9.2.1. 局所ウィンドウを表示

Debug(デバッグ)メニューから**Windows**(ウィンドウ)を選んで**Locals**(局所)をクリックしてください。(デバッガは走行しているか、または中断動作でなければなりません。)

4.9.2.2. 代替状況を選ぶ

既定の状況は現在の実行位置を含む関数です。**Locals**(局所)ウィンドウで表示するための代替状況を以下のように選ぶことができます。

- 望む関数、スロット、またはプログラムを選ぶのに**Debug Location**(デバッグ位置)ツールバーを使ってください。
- **Call Stack**(呼び出しスタック)または**Threads**(スレッド)のウィンドウで項目上をダブルクリックしてください。

Locals(局所)ウィンドウで情報を見るまたは変更するにはデバッガが中断動作でなければなりません。**Continue**(継続)を選ぶ場合、プログラム実行の間に**Locals**(局所)ウィンドウでいくつかの情報が現れるかもしれませんが、これはプログラムが次回中断する(換言すると、中断点に当たる、または**Debug**(デバッグ)メニューから**Break All**(全て中断)を選ぶ)まで現在のものではありません。

4.9.2.3. 局所ウィンドウで変数の値を変更

1. デバッガは中断動作でなければなりません。
2. Locals(局所)ウィンドウで編集したい値をダブル クリックするか、またはTabキーを使うことによって選んでください。
3. 新しい値を入力してEnterを押してください。

! **注意:** 浮動小数点値の編集は小数部分の10進⇒2進変換のため僅かな不正確さに帰着し得ます。表面上は害のない編集でさえ、浮動小数点変数で下位側のいくつかのビットへの変更に帰着し得ます。

4.9.2.3.1. 数値形式設定

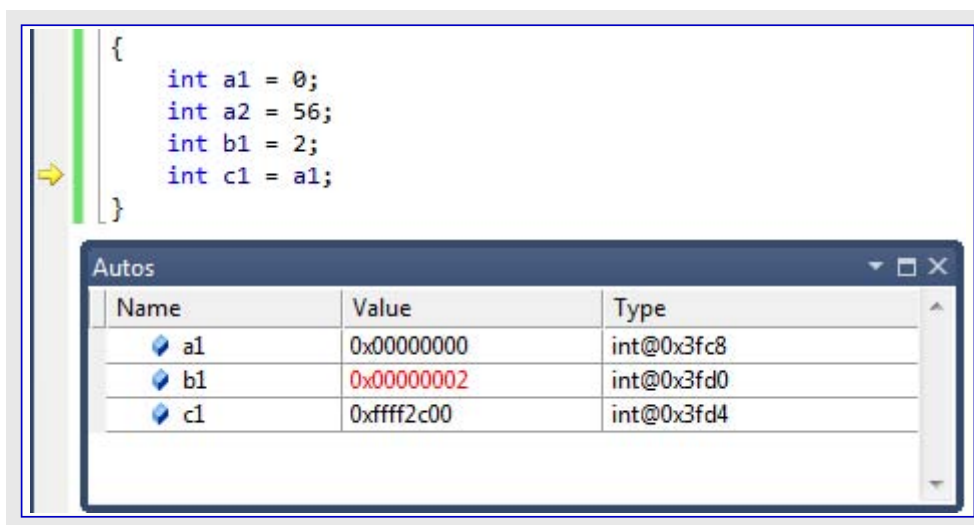
デバッガ ウィンドウで使われる数値形式を10進または16進に設定することができます。Locals(局所)ウィンドウの内側を右クリックし、Hexdecimal display(16進表示)メニュー項目をチェック設定/チェック解除してください。

4.9.3. 自動ウィンドウ

Autos(自動)ウィンドウは現在の文と前の文で使われる変数を表示します。

現在の文は現在実行位置の文(実行が続く場合は次に実行される文)です。デバッガはこれらの変数、故にウィンドウ名を自動的に識別します。

構造体と配列の変数は要素を表示または隠すのに使うことができる樹状表示部を持ちます。



自動ウィンドウを表示

Debug(デバッグ)メニューからWindows(ウィンドウ)を選んでAutos(自動)をクリックしてください。(デバッガが走行しているか、または中断動作でなければなりません。)

4.9.3.1. 自動ウィンドウで変数の値を編集

1. デバッガは中断動作でなければなりません。
2. 必要ならば、Autos(自動)ウィンドウを表示してください。
3. Value(値)列で変更したい値をダブル クリックしてください。
--または、--
行を選択するために単一クリックしてTabキーを押してください。
4. 新しい値を入力してEnterを押してください。

! **注意:** 浮動小数点値の編集は小数部分の10進⇒2進変換のため僅かな不正確さに帰着し得ます。表面上は害のない編集でさえ、浮動小数点変数で下位側のいくつかのビットへの変更に帰着し得ます。

4.9.3.2. 数値形式設定

デバッガ ウィンドウで使われる数値形式を10進または16進に設定することができます。Autos(自動)ウィンドウの内側を右クリックし、Hexdecimal display(16進表示)メニュー項目をチェック設定/チェック解除してください。

4.9.4. 一瞬監視と監視

デバッグの間に変数または式の値を追跡したいと思うかもしれません。それを行うためにカーソル下の式でダブル クリックしてAdd Watch(監視追加)またはQuickWatch(一瞬監視)を選んでください。

QuickWatch(一瞬監視)ダイアログ枠は変数と式を調べて評価させます。QuickWatchがモーダル ウィンドウ(使用者が応答しなければ親ウィンドウに制御を戻さない子ウィンドウ)のため、デバッグを続け得る前にそれを選ばなければなりません。QuickWatchで変数の値を編集することもできます。変数を監視する方法のより多くの情報については「4.9.1. 監視ウィンドウ」をご覧ください。

幾許かの使用者はQuickWatchが何故有用なのか不思議に思うかもしれません。何故Watch(監視)ウィンドウに変数や式を追加できないのでしょうか?。それは可能ですが、1つまたはより多くの変数を伴う迅速な走り書き計算を行いたい場合、そのような計算でWatch(監視)ウィンドウを乱雑にしたくはありません。それがQuickWatchダイアログ枠が特に有用な時です。

QuickWatchダイアログ枠の別の特徴は大きな変更可能なことです。大きなオブジェクトのメンバを調べたい場合、Watch(監視)、Locals(局所)、Autos(自動)のウィンドウよりも、展開して樹状QuickWatchを調べることがしばしばより簡単です。

QuickWatchダイアログ枠は同時に1つよりも多い変数または式を見ることを許しません。また、QuickWatchがモーダル ウィンドウのため、QuickWatchが開いている間にコードを通して段階実行するような操作を実行することができません。これらのことを実行したい場合、代わりにWatch(監視)ウィンドウを使ってください。

いくつかの式は変数の値を変更する副作用を持ち、また、さもなくばそれらが実行される時にプログラムの状態を変更します。QuickWatchダイアログ枠での式の評価はコードで式が実行するのと同じ効果を持ちます。これは式の副作用を考慮しない場合に予期せぬ結果を生成し得ます。

注: Microchip Studioでは変数の上にカーソルを置くことによって変数値を見ることができます。DataTip(データ情報)と呼ばれる小さな枠が現れます。

一瞬監視ダイアログ枠を開く

中断動作の間にDebug(デバッグ)メニューでQuickWatch(一瞬監視)を選んでください。

追加した変数と共に一瞬監視ダイアログ枠を開く

中断動作の間にソース ウィンドウ名で変数名を右クリックしてQuickWatch(一瞬監視)を選んでください。これはQuickWatch(一瞬監視)ダイアログ枠内に自動的にその変数を配置します。

一瞬監視の式を監視ウィンドウに追加

QuickWatch(一瞬監視)ダイアログ枠でAdd Watch(監視に追加)をクリックしてください。

QuickWatchダイアログ枠に表示されたどの式もWatch(監視)ウィンドウの式の一覧に追加されます。この式は通常Watch1(監視1)ウィンドウに追加されます。

4.9.5. 表現形式

Microchip Studioデバッグは「4.9.4. QuickWatch(一瞬監視)とWatch(監視)」、「4.15. Memory View(メモリ表示)」、「4.9.1. Watch(監視)ウィンドウ」、または直のウィンドウで式を入力する時に作動する式評価部を含みます。式評価部はBreakpoints(中断点)ウィンドウとデバッグ内の他の多くの場所でも動作します。

一般的な構文

val, formatString (値, 表現形式文字)

値用型指定子

下表はデバッグによって認証される型指定子を示します。

表4-3. 値用デバッグ型指定子

指定子	型	表現	表示される値
d,i	符号付き10進整数	0xF000F065, d	-268373915
u	符号なし10進整数	0x0065, u	101
b	符号なし2進数	0xaa,b2	0b10101010
o	符号なし8進整数	0xF065, o	0170145
x,X	16進整数	61541, x	0x0000F065
1,2,4,8	d,i,u,o,x,Xに対してバイト数を指定する接尾辞として	00406042,x2	0x0c22
s	文字列	0x2000, s	"Hello World"
f	符号付き浮動小数点数	(3./2.), f	1.500000
e	符号付き指数表記	(3./2.), e	1.500000e+000
g	符号付き符号小数点または符号付き指数表記、どちらもより短い	(3./2.), g	1.5
c	単一文字	0x0065, c	101 'e'

配列としてのポインタ用大きさ指定子

配列として見たいオブジェクトに対するポインタがある場合、以下のように配列の要素の番号を指定するのに整数を使うことができます。
ptr,10 または array,20

メモリ型指定子

以下のメモリ型指定子はメモリ参照を特定のメモリ型に強制します。メモリウィンドウのアドレス領域で使うには配列として見たいオブジェクトに対するポインタを持つべきです。配列の要素の番号を指定するのに整数を使うことができます。

表4-4. デバッグ メモリ型指定子

指定子	表現	表示される値
flashまたはprogram	プログラム メモリ	0,flash
data	データ メモリ	0x2000,data
sram	SRAM	0x100,sram
reg or registers	レジスタ	1,reg
io, eeprom, fusebytes, lockbytes, signature, usersign, prodsign	同じ名前を持つメモリ型	

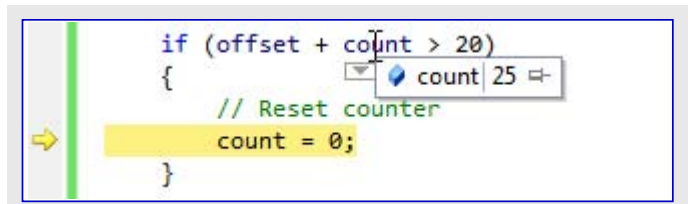
4.10. データ情報 (DataTips)

データ情報(DataTips)はデバッグ中にプログラム内の変数についての情報を見る便利な方法を提供します。データ情報は中断動作でだけで且つ現在の実行の可視範囲にある変数でだけ動きます。

Microchip Studioではデータ情報をソース ファイルの指定位置にピン止めすることができ、またそれらを全てのMicrochip Studioウィンドウの上部で浮かせることができます。

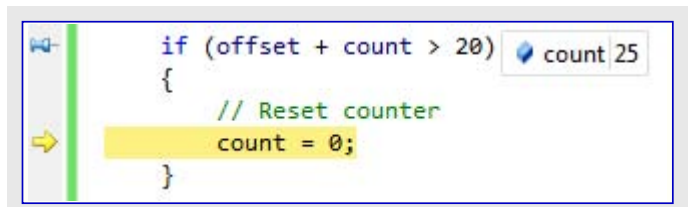
データ情報(DataTips)を表示 (中断動作でだけ)

1. ソース ウィンドウで現在の可視範囲内のどれかの変数上にマウス ポインタを置いてください。データ情報(DataTip)が現れます。



2. マウス ポインタを取り去ると、データ情報(DataTip)が消えます。それが開いたままで留まるようにデータ情報をピン止めするにはPin to source(ソースにピン止め)アイコンをクリックするか、または、
 - 変数上で右クリックし、その後にPin to source(ソースにピン止め)をクリックしてください。

ピン止めされたデータ情報はデバッグ作業終了時に閉じます。



データ情報のピンを外して浮きにする

- ピン止めされたデータ情報(DataTip)でソース アイコンからUnpin(ピン外し)をクリックしてください。

ピン止めアイコンはピンを外された位置に変わります。データ情報は今や開いたどのウィンドウ上にも浮きます。浮いているデータ情報はデバッグ作業終了時に閉じます。

浮いているデータ情報を再度ピン止め

- データ情報(DataTip)でピン アイコンをクリックしてください。

ピン アイコンはピン止めされた位置に変わります。データ情報がソース ウィンドウの外側の場合、ピン アイコンが禁止され、データ情報をピン止めすることができません。

データ情報を閉じる

- マウス ポインタをデータ情報(DataTip)上に置き、その後に閉じる(Close)アイコンをクリックしてください。

全てのデータ情報を閉じる

- Debug(デバッグ)メニューでClear All DataTips(全データ情報を解除)をクリックしてください。

指定ファイルに対して全てのデータ情報を閉じる

- Debug(デバッグ)メニューでClear All DataTips Pinned to File(ファイルにピン止めされた全データ情報を解除)をクリックしてください。

4.10.1. 情報の展開と編集

そのメンバを見るために配列、構造体、オブジェクトを展開するのにデータ情報(DataTip)を使うことができます。データ情報から変数の値を編集することもできます。

その要素を見るために変数を展開するには、

- データ情報で変数名の前に来る+符号上にマウスポインタを置いてください。樹状形式でその要素を見るために変数が展開します。変数が展開されると、上下移動にキーボードの矢印キーを使うことができます。代わりに、マウスを使うこともできます。

データ情報(DataTip)を用いて変数の値を編集

- データ情報(DataTip)で値をクリックしてください。これは読み込み専用値に対して禁止されます。
- 新しい値を入力してEnterを押してください。

4.10.2. データ情報の透明化

データ情報(DataTip)の向こう側のコードを見たい場合、そのデータ情報を一時的に透明にすることができます。これはピン止めまたは浮きにされたデータ情報には適用されません。

データ情報を透明にする

- データ情報(DataTip)でCtrlを押してしてください。
データ情報はCtrlキーを押さえつけている限り透明に留まります。

4.10.3. 複雑なデータ型の可視化

データ情報(DataTip)内の変数名の次に拡大鏡アイコンが現れる場合、そのデータ型の変数に対して1つまたは複数の可視器が利用可能です。もっと意味の或る、通常の図画、方法で情報を表示するために可視器を使うことができます。

可視器を用いて変数の内容を見る

- データ型に対して既定の可視器を選ぶために拡大鏡アイコンをクリックしてください。
--または、--
データ型に対して適切な可視器の一覧から選ぶために可視器の次のポップアップ矢印をクリックしてください。
可視器は情報を表示します。

4.10.4. 監視ウィンドウへの情報追加

変数の監視を続けたい場合、その変数をデータ情報(DataTip)からWatch(監視)ウィンドウへ追加することができます。

監視ウィンドウに変数を追加

- データ情報(DataTip)を右クリックし、その後にAdd Watch(監視に追加)をクリックしてください。

変数が監視ウィンドウに追加されます。複数の監視ウィンドウを支援する版を使う場合、この変数はWatch1(監視1)に追加されます。

4.10.5. データ情報のインポートとエクスポート

データ情報(DataTip)をXMLファイルにエクスポートすることができ、それは仲間と共用したり、テキストエディタで編集することができます。

データ情報(DataTip)をエクスポート

- Debug(デバッグ)メニューでExport Data Tips(データ情報をエクスポート)をクリックしてください。
Export Data Tips(データ情報エクスポート)ダイアログ枠が現れます。
- XMLファイルを保存したい場所に誘導するのに標準的なファイル技法を使い、File name(ファイル名)にファイルに対する名前を入力し、その後OKをクリックしてください。

データ情報(DataTip)をインポート

- Debug(デバッグ)メニューでImport Data Tips(データ情報をインポート)をクリックしてください。
Import Data Tips(データ情報インポート)ダイアログ枠が現れます。
- 開きたいXMLファイルを見つけるためにダイアログ枠を使ってOKをクリックしてください。

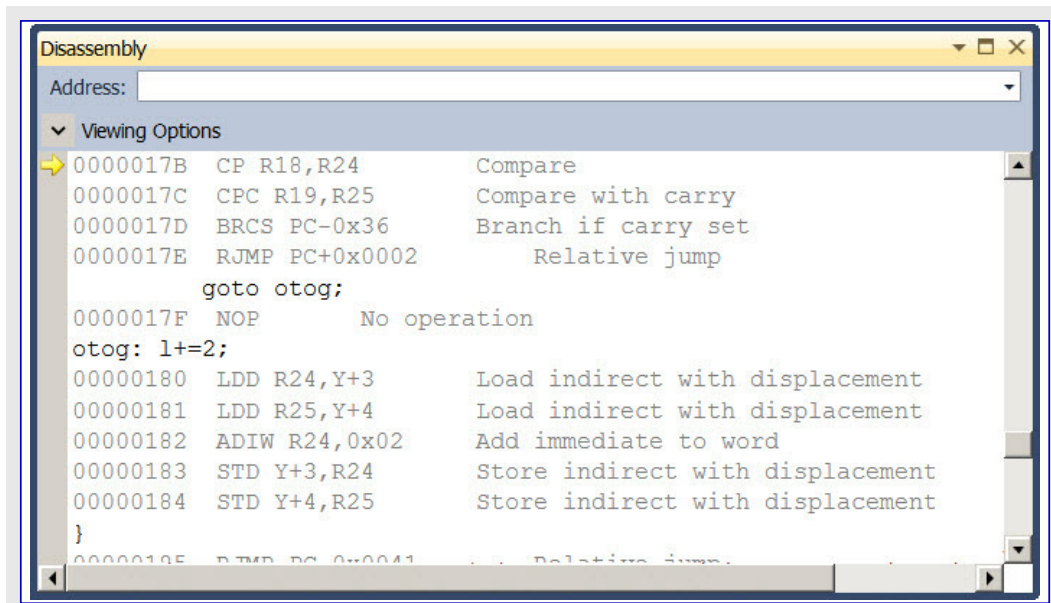
4.11. 逆アセンブリ ウィンドウ

Disassembly(逆アセンブリ)ウィンドウはデバッグ時にだけ利用可能です。支援されるどれかの高位言語が使われると、ソースウィンドウが自動的に表示され、逆アセンブリウィンドウがOFFになります。デバッグ作業中に、Debug(デバッグ)⇒Windows(ウィンドウ)⇒Disassembly(逆アセンブリ)を選ぶか、またはCtrl+Alt+Dによってそれを許可してください。

逆アセンブリウィンドウは逆アセンブルしたプログラムコードを示します。プログラム実行とAVR命令はこの表示部で進めることができます。逆アセンブリウィンドウの内側のクリックによって中断点を設定、カーソルの位置から走行、ソースコードに行くことができます。逆アセンブリウィンドウからソースコードを編集することはできません。

アセンブリ命令に加えて、逆アセンブリウィンドウは以下の任意情報を示すことができます。

- 各命令が置かれたメモリアドレス。本来のアプリケーションに関して、これは実際のメモリアドレスです。
- アセンブリコードを派生したソースコード
- 具体的なマシンのコード バイトのバイト表現
- メモリアドレスのシンボル名
- ソースコードに対応する行番号



アセンブリ言語命令は命令名に対する略語であるニーモニックと変数、レジスタ、定数を表すシンボルから成ります。各機械語命令は通常、1つまたはより多くの変数、レジスタ、または定数が後続する1つのアセンブリ言語ニーモニックによって表されます。

アセンブリコードはプロセッサのレジスタ、または管理されたコードの場合の共通言語走行時レジスタを重く信頼するため、度々それを見つけるのに、レジスタ内容を調べることを許すRegisters(レジスタ)ウィンドウと共に逆アセンブリウィンドウを使うことが有用です。

注: 明確なアドレスで動く命令での矛盾を見るかもしれません。これはAVRアセンブラ、アセンブリ言語、GCCアセンブラ、より大きなコンピュータシステムで使われるアセンブリ言語間の歴史的な違いから発しています。従って、下のもののように見える逆アセンブリに出会うかもしれません。

```
13:      asm volatile ("JMP 0x0001778A");
0000007D 0c. 94. c5. bb      JMP 0x0000BBC5      Jump >
```

ここでアセンブリ命令のJMP 0x0001778AはGCCアセンブラによってアセンブルされ、そしてMicrochip Studio内の組み込み逆アセンブラを用いて逆アセンブルされ、それは0x0000BBC5への分岐に解析し、初期アセンブリでのアドレスの正確に半分です。

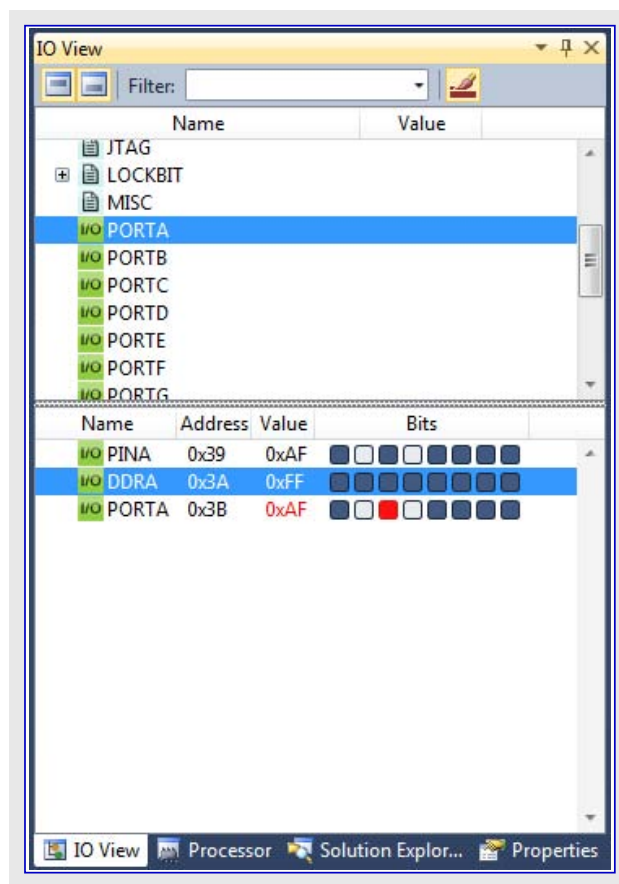
アドレスは常に逆アセンブリで示される行アドレスと同じ次元(特性)で、故にコードは機能的に同じです。

4.12. I/O表示部

4.12.1. I/O表示部について

I/O表示部の目的は現在のプロジェクトに対して目的対象デバイスのレジスタの概要を提供することです。これは設計中の素早い参照として扱い、プロジェクトがデバッグ動作の時にレジスタ値を表示する能力です。この表示部は32ビットと8ビットのデバイスを同様に支援します。

ツールウィンドウの既定表示部は上半分に周辺機能群、下半分にレジスタを持つ水平分割ウィンドウです。各周辺機能は代表的に(上半分)の周辺機能表示部でレジスタを展開することによって表示することができる定義された設定の組と値の列挙を持ちます。(下半分の)レジスタ表示部は選択した周辺機能群に属する全てのレジスタを表示します。周辺機能が全く選択されていない場合、この表示部は空です。各レジスタはそのレジスタに属する予め定義された値の群を表示するために展開することもできます。



4.12.2. I/O表示部ツールの使い方

I/O表示部は開発環境内で単一ツールウィンドウに制限されます。同時にI/O表示部の1つの実体が有り得るだけです。このウィンドウを開くにはDebug(デバッグ)⇒Windows(ウィンドウ)⇒I/O View(I/O表示部)を選んでください。設計動作時、I/O表示部は入力に対して禁止されます。未だ配置や選別を変更することと表示部を操ることが可能ですが、値を設定したり読んだりには全くできません。レジスタの値を読んだり変更したりするにはAVR Studioがデバッグ中断動作(実行一時停止)でなければなりません。この動作ではI/O表示部の全ての部分が許可され、表示部で値を読んで更新することができます。

レジスタの値の簡単な表示に加えて、I/O表示部は独立した列でレジスタ内の各ビットを表示します。設定されたビットは既定によって灰色で、解除されたビットは無色(既定は白)です。ビットを変更するには単にそれをクリックします。値は交互に切り替えられます。

4.12.3. 中断動作での値とビットの編集

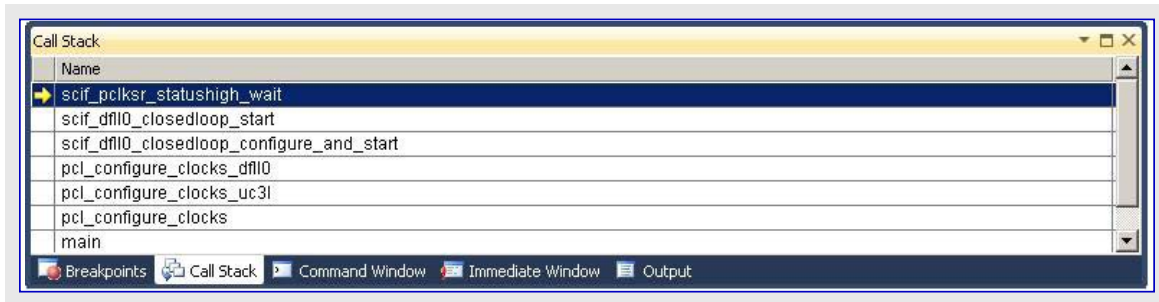
プロジェクトがデバッグ中断動作の時に、値は値領域をクリックすることによって変更することができ、新しい値を書くことができます。いくつかの値とビットはそれらが読み込み専用のため変更することができず、いくつかのビットは書き込み専用かもしれません。より多くの情報については各デバイスに対する資料をご覧ください。ビットまたは値が設定されると、デバイスから直ちに読み戻され、I/O表示部がデバイスからの実際の値だけを表示することを保証します。新しい値が設定されるけれど、意図したように更新されない場合、レジスタが書き込み専用または単にアクセス不可かもしれません。

レジスタが最後に表示された時から更新されてしまった時に、赤色の値とビットでの表示でこれを示します。最後の時からビットが設定(1)されてしまった場合、一様の赤になります。解除(0)されてしまった場合は単純に赤境界になります。この機能はツールバーでONまたはOFFに交互切り替えすることができます。

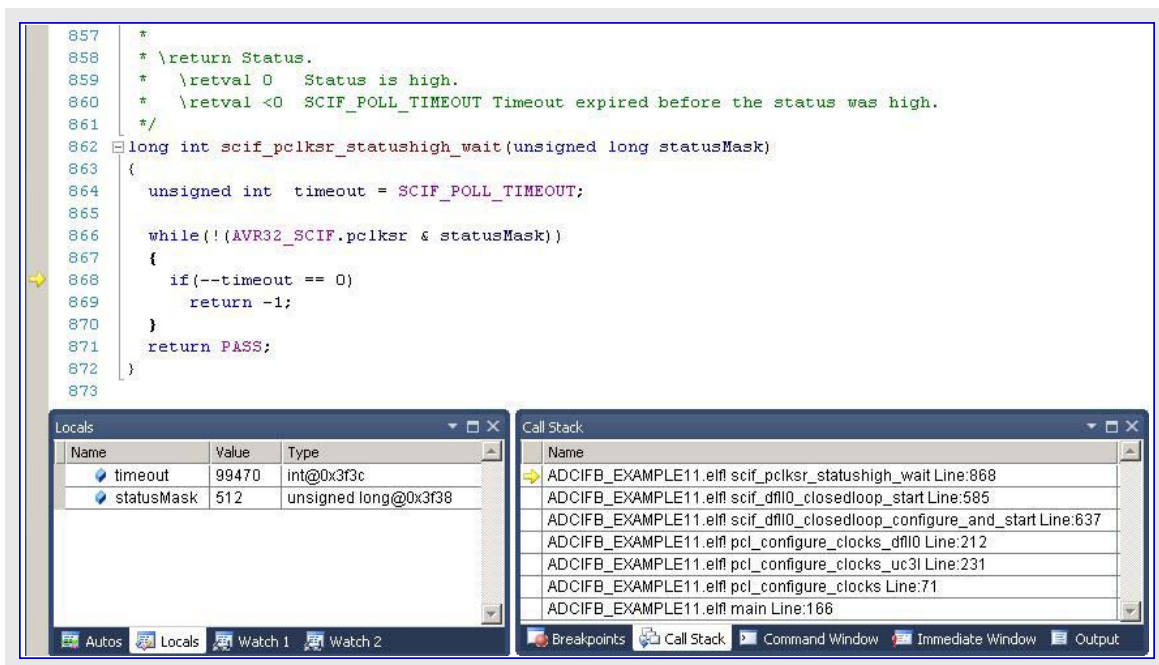
4.16. 呼び出しスタック表示部

注: Call Stack(呼び出しスタック)ウィンドウは現在、32ビット デバイスに対してだけ支援されます。

呼び出しスタックは現在のメソッド(方法)の呼び出し側の階層的な情報を示します。既定によって、呼び出しスタック ウィンドウは各関数の名前を表示します。呼び出しスタックを表示するにはメニューでDebug(デバッグ)⇒Windows(ウィンドウ)⇒Call Stack(呼び出しスタック)をクリックしてください。



関数名と共に、単位部名、行番号などのような任意選択情報も表示されるかもしれません。この任意選択情報の表示はONまたはOFFに切り替えることができます。表示される任意選択情報をON/OFF切り替えするには呼び出しスタック ウィンドウを右クリックしてShow [情報名][情報名]を選択または非選択にしてください。



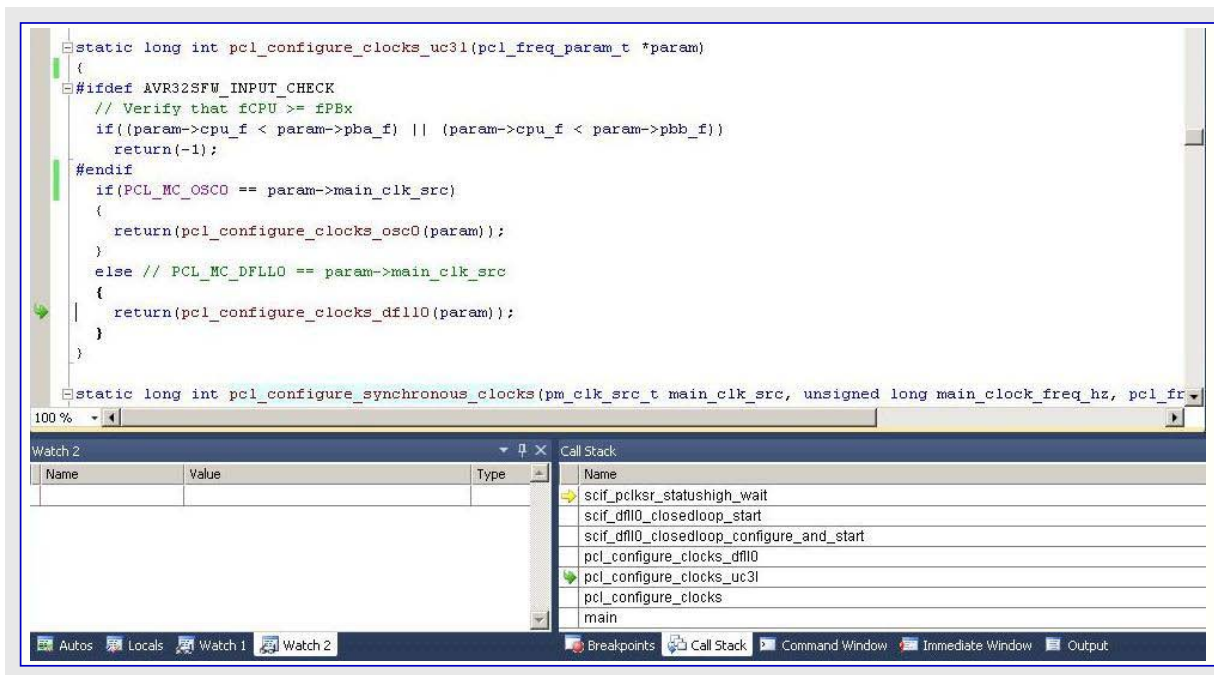
黄色矢印はスタック フレームの現在の実行ポインタを示します。既定によって、これはソース、逆アセンブリ、Locals(局所)、Watch(監視)、Autos(自動)ウィンドウで現れる情報枠です。またスタック フレームの状況は呼び出しスタック ウィンドウで表示される別のフレームに対して変更され得ます。

警告 呼び出しスタックは最適化レベル-O1とそれ以上で全ての呼び出しフレームを示さないかもしれません。

別のスタック フレームに切り替え

1. 呼び出しスタック ウィンドウで見たいコードとデータを持つフレームを右クリックしてください。
2. **Switch to Frame**(フレーム切り替え)を選んでください。

巻きあがった尾を持つ緑の矢印は変更したスタック状況を示します。実行ポインタは未だ黄色の矢印で記される元もフレームに留まります。**Debug**(デバッグ)メニューから**Stop**(停止)または**Continue**(継続)を選ぶ場合、実行は選んだフレームではなく、黄色矢印から継続されます。

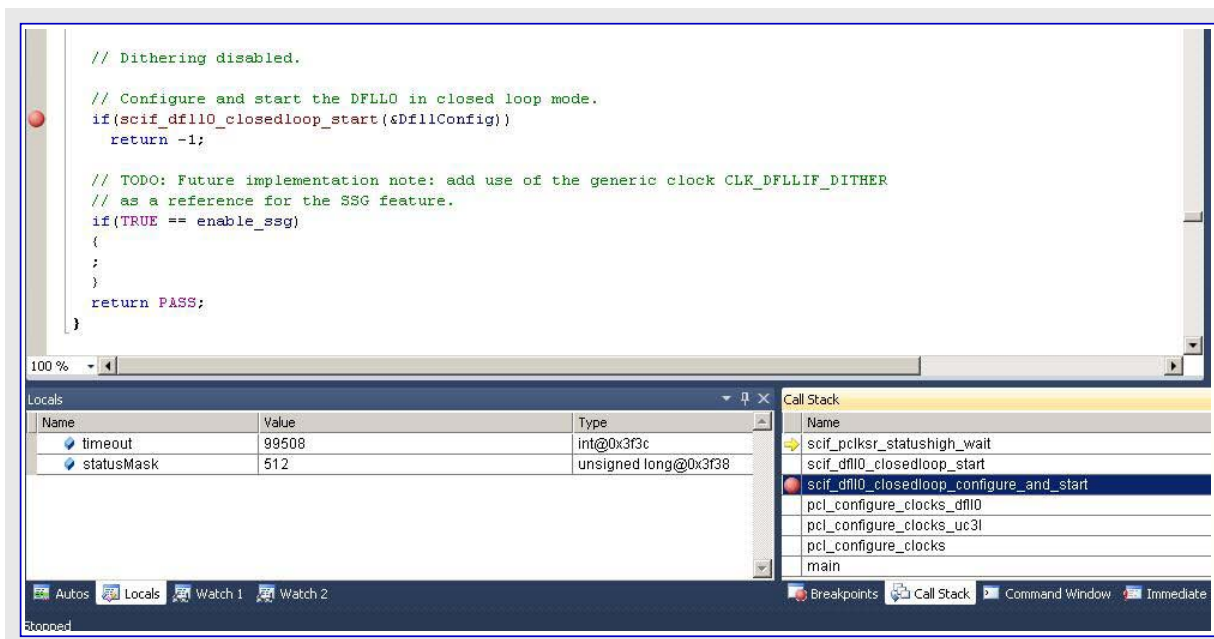


呼び出しスタック上の関数に対してソース/逆アセンブリコードを表示

1. 呼び出しスタック ウィンドウで見たいソース コードを持つ関数を右クリックし、**Go To Source Code**(ソース コードへ行く)を選んでください。
2. 呼び出しスタック ウィンドウで見たい逆アセンブリ コードを持つ関数を右クリックし、**Go To Disassembly**(逆アセンブリへ行く)を選んでください。

関数呼び出しの出口点に中断点を設定

呼び出しスタック ウィンドウで中断点を追加したいスタック フレームを右クリックしてください。中断点を追加するために**Breakpoint**(中断点)⇒**Insert Breakpoint**(中断点挿入)を選んでください。



4.17. オブジェクト ファイル形式

Microchip Studioが好ましいオブジェクト ファイル/デバッグ情報の形式としてELF/DWARFを使う一方で、あなたは他の形式でデバッグするかもしれません。様々なコンパイラ供給者からいくつかのオブジェクト ファイル形式が支援されます。それらのファイルを開いてデバッグすることができますが、Microchip Studio内からコードを編集することができないかもしれません。コードを編集するのにあなたのエディタを、そして(再読み込み鉤を使って)デバッグするのにMicrochip Studioを使い、未だ強力なコード/デバッグ環境を持つでしょう。

全ての外部デバッグ作業はMicrochip Studioによって支援されるオブジェクト ファイルを読み込む必要があります。デバッグ用のオブジェクト ファイルは通常、公開版では含まれないシンボリック情報を含みます。このデバッグ情報はデバッグ時に拡張された可能性、例えば、Cのような高位言語でソースファイルの段階実行や中断点設定を与えることをMicrochip Studioに許します。

予めコンパイルされたオブジェクト ファイルはメニュー命令のOpen file(ファイルを開く)⇒Open Object File for Debugging(デバッグ用にオブジェクト ファイルを開く)を用いることによって開くことができます。より多くの情報については「3.6. Microchip Studioでのオブジェクト ファイルのデバッグ」項をご覧ください。

表4-5. Microchip Studioによって支援されるオブジェクト ファイル形式

オブジェクト ファイル形式	拡張子	説明
UBROF	.d90	UBROFはIAR専有形式です。デバッグ出力ファイルは全ての監視形式を支援するための完全なデバッグ情報一式とシンボルを含みます。UBROF8とそれ以前版が支援されます。これはIAR EW 2.29とそれ以前版の既定出力形式です。IAR EW 3.10とそれ以降版でUBROF8生成に強制する方法を下でご覧ください。
ELF/DWARF	.elf	ELF/DWARFデバッグ情報は開放標準です。このデバッグ形式は全ての監視形式を支援するための完全なデバッグ情報一式とシンボルを含みます。Microchip Studioによって読まれる形式版はDWARF2です。DWARF2出力用に構成設定されるAVR-GCC版はこの形式を生成することができます。
AVRCOFF	.cof	COFFはMicrochip Studioによって支援される拡張ツールを作成する第三者の供給者を意図された開放標準です。
AVRアセンブラ形式	.obj	AVRアセンブラ出力ファイル形式はソース段階実行用のソース情報を含みます。これは内部Microchip形式専用です。いくつかの監視情報を得るために自動的に.mapファイルが解析されます。

デバッグ前に、上の形式の1つでデバッグ情報を持つオブジェクト ファイルを生成するためにあなたの外部コンパイラ/アセンブラを構成設定を確実にしてください。

第三者のコンパイラ供給者はMicrochip Studioでの支援を確実にするためにELF/DWARFオブジェクト ファイル形式を出力すべきです。任意でデバッグとコンパイルの両方の支援を持つためにあなたは拡張を提供することができます。より多くの情報については「1.4. 問い合わせ情報」をご覧ください。

 **助言:**  IARW32でAVR互換ELFファイルを生成する方法:

Project options(プロジェクト任意選択)⇒Output format(出力形式)ダイアログでelf/dwartを選び、Project options(プロジェクト任意選択)⇒Format variant(形式変種)でArm-compatible “-yes”を選んでください。

 **助言:**  IAR EW 3.10とそれ以降版をUBROF8生成に強制する方法:

既定によってIAR EW 3.10とそれ以降版はUBROF9を出力します。現在、Microchip Studioはこの形式を読むことができません。デバッグ形式をUBROF8に強制するにはプロジェクト任意選択ダイアログを開き、Output format(出力形式)設定をubroff8(forced)に変更してください。この任意選択設定時、既定のファイル名拡張子が’.d90’から’.dbg’に変わることに注意してください。’.d90’拡張子を保つにはOverride default(既定上書き)チェック鉤をクックして拡張子を変更してください。

4.18. 追跡

Microchip Studioではプラグインに基づいて追跡(Trace)が提供され、これはMicrochip Studioの核から独立した各種プラグインが追跡を可視化するための違うグラフィック表示部の提供物であることを意味します。

追跡の部門ではデバイスとツールの組み合わせが支援する各種追跡供給元を記述するいくつかの用語があります。これらの高位供給元名は各種基本構造特有追跡供給元に置かれます。

以下の項は利用可能かもしれない高位追跡供給元のいくつかを記述し、それは目的対象基本構造に置かれます。デバイス特有の詳細は各々のデータシートで利用可能ななので、各種供給元の高位記述だけが与えられます。

注: Microchip Studioで追跡能力を発見するための基本構造はチップ自身が報告するものに基づき、これはその能力が探れるように走行することをデバッグ作業が必要であることを意味します。これは監視供給元を活性にする時に、デバイスが開始中の間に問われた供給元を支援しない場合にMicrochip Studioが失敗するかもしれません。

4.18.1. 応用出力

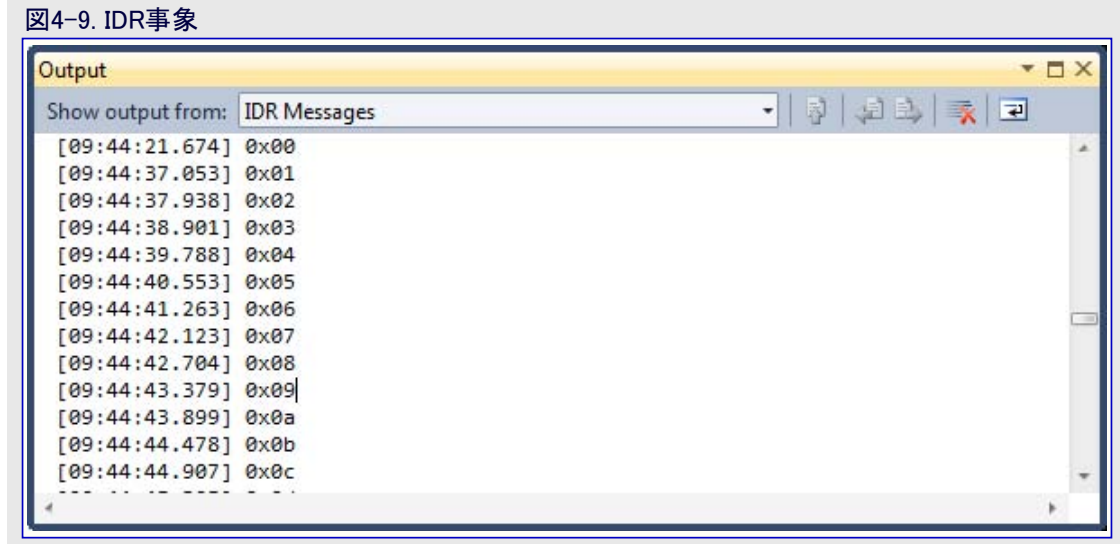
応用出力は起因(stimuli)ポートとして知られるものを提供する技法に対する一般的な名前です。これは接続されたデバッグにデータを出力するためにデバイスで走行する応用に対していくつかの手段を含みます。

4.18.1.1. ITM

ITMはArmコアでのデバッグシステムの任意選択部分です。この単位部は応用がそこにデータを書き込めるレジスタの組を提供し、これはデバッグに流れ出されます。

4.18.1.2. IDR事象

デバッグされつつある間にアプリケーションプログラムがAVRデバイスのOCDDRレジスタにデータバイトを書くと、デバッグはこの出力値を読んで下図で示されるようにOutput(出力)ウィンドウでIDR事象としてそれを表示します。OCDDRレジスタは与えられ間隔でポーリングされ、故にデバッグが信頼に足る結果を生じないように指定されたものよりも高い頻度でそれに書きます。デバイスのデータシートは与えられた値が読まれたことを調べる方法を説明します。



IDR事象がデバッグから送られなかった場合にOutput(出力)ウィンドウが”IDR Messages(IDRメッセージ)”引き落としメニューを持たないことに注意してください。

4.18.2. プログラム カウンタ採取

この追跡供給元は周期的にデバイスのプログラク カウンタを読み出す何らかの採取システムを伴います。これはその後いくつかの統計的平均に基づいて、費やされる実行時間のグラフに使うことができます。

4.18.2.1. Arm®実装

Arm Cortexコアではプログラム カウンタを採取する2つの方法があります。1つ目はコード自身でのどんな影響もなしにプログラム カウンタをデバッグに放出するデバッグシステム内の任意単位部を使います。プログラム カウンタはSWOピンで放出されます。

全てのCortex実装がプログラム カウンタを放出する訳ではないため、Microchip Studioはコアが走行している間にプログラム カウンタの周期的な読み出しを行うことも支援します。これはデバッグシステムがメモリバスをアクセスすることが必要なために応用を走行する上で小さな影響を持ち、コアが走行中のメモリ読み出しを殆どのCortexデバイスが支援するために可能です。

4.18.2.2. AVR® 32ビット実装

AVR 32ビット コアでのプログラム カウンタ読み取りは、コアが走行している間に実際のメモリの読み出しをコアが支援するため、「4.18.2.1. Arm®実装」で言及したのと同じ方法で可能です。

4.18.3. 変数監視

変数監視は通常、データ中断点(ブレイクポイント)によって網羅されます。「4.8. データ中断点」をご覧ください。けれども、いくつかのシステムに於いて、コアを停止することなく情報をデバッグへ放出することをデータ中断点にさせるのも可能で、例えばデータ中断点が失敗を起こす何らかの外部タイミング必要条件を持つ応用での変数監視が可能なることを意味します。

4.18.3.1. Arm®実装

Cortexコアでのデータ中断点はデバッグシステムが必要とした単位部を実装する場合に追跡パケットを放出するように変更することができ、これはコアでの実行に影響を及ぼすことなく、特定のメモリ位置に対して読み書きについての情報を得ることが可能なことを意味します。

これへ戻るため、コアが実行中に与えられた間隔でメモリ位置を読むことも可能です。これはどの特定事象データでもなく、コアが生メモリ読み出しを支援する場合、メモリのいくつかの部分の採取が可能なることを意味します。これはデバッグシステムがメモリバスへのアクセスを持つことを必要とするため、実行に於いて微細な影響を持ちます。

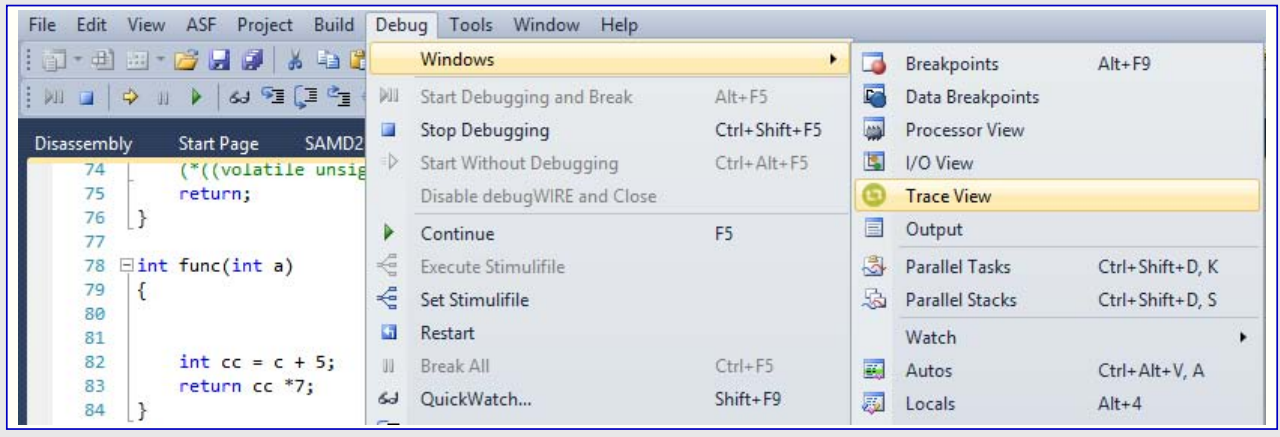
4.18.3.2. AVR® 32ビット実装

AVR 32ビット コアはメモリ位置の生読み出しも支援します。これはコアが実行中にMicrochip Studioがメモリ位置をポーリングすることができ、変数が時間に渡ってどう変化するかを統計的表示を与えます。

4.19. 追跡表示部

Trace(追跡)表示部は目的対象が走行している時にプログラムカウンタの足跡を記録することを許します。プログラムカウンタ分岐は各々のソース行情報に配置されます。それは実行されたソース行に対する適用範囲と統計も含まれます。追跡表示部を開くにはDebug(デバッグ)⇒Windows(ウインドウ)⇒Trace View(追跡表示部)へ行ってください。プロジェクトは追跡表示部の機能を使うためにMicrochip Studioで開かれなければなりません。

図4-10. メニューから追跡表示部を開く



4.19.1. 追跡表示部任意選択

Trace(追跡)表示部ツールバーは以下の要素を持ちます。

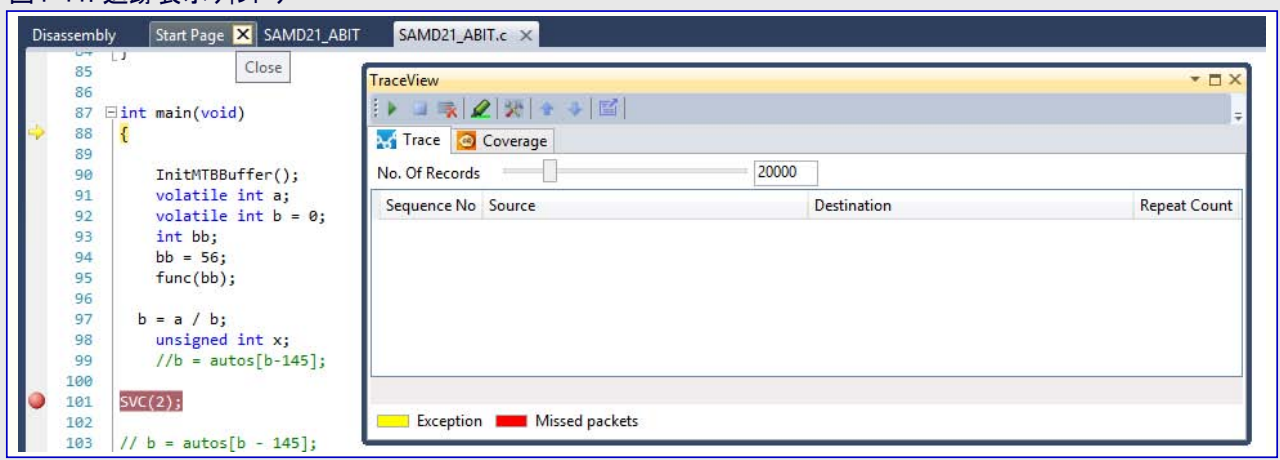
- ▶ - プログラム追跡開始
- - プログラム追跡停止
- ✖ - プログラム追跡解消
- 🖍 - ソースコードの強調表示交互切り替え
- 🔧 - プログラム追跡を記録するようにデバイスを構成設定
- ⬆️⬆️ - Trace Stack(追跡スタック)表示部で例外記録を探す
- 📄 - 網羅率統計をxml/xslt報告にエクスポート

4.19.1.1. プログラム追跡開始

Trace(追跡)表示ウインドウで開始鈕をクリックすることによってプログラム追跡を開始してください。開始鈕はデバッグの間に許可されます。追跡はデバッグ作業で何度でも開始と停止を行うことができます。新しい追跡作業の開始は以前の追跡作業の全ての追跡情報を解消します。

注: SRAMの領域はデバイスに追跡を記録させるように割り当てられなければなりません。より多くの情報については「4.19.1.5. 追跡表示部設定」を参照してください。

図4-11. 追跡表示ウインドウ



4.19.1.2. 追跡停止

追跡は走行動作またはデバッグ動作で停止することができます。追跡作業はデバッグ作業が終わる時に使用者の介在なしで自動的に終了されます。

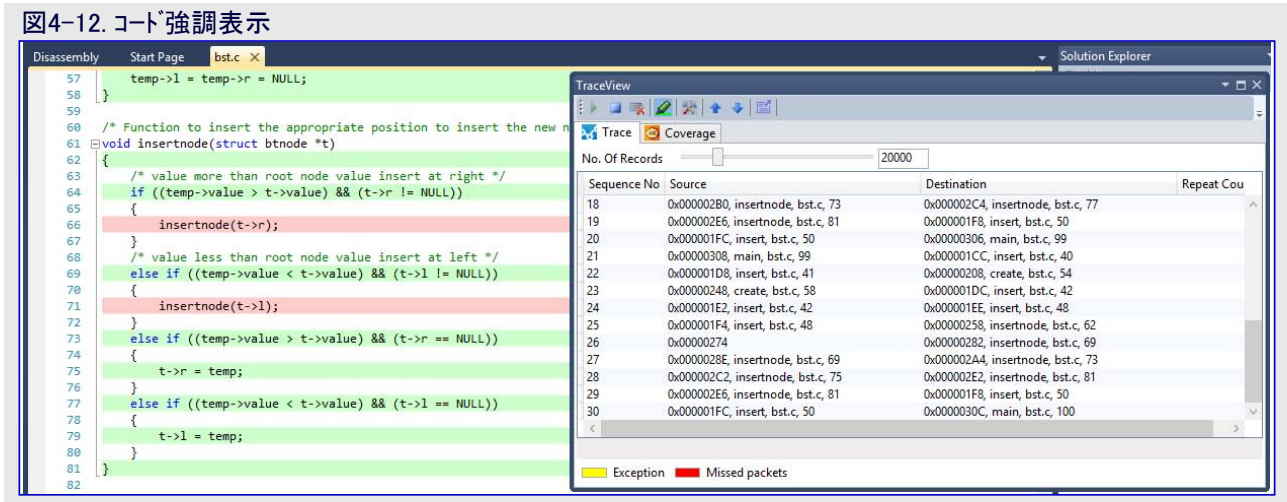
4.19.1.3. 追跡解消

解消鈕はTrace(追跡)表示ウィンドウで追跡を解消します。新しい追跡情報は継続する連番を持つ同じツール ウィンドウで記録されます。解消は追跡作業内で何回でも行うことができます。一旦追跡データが解消されると、復元することはできません。

4.19.1.4. ソースコード強調表示

強調表示はソースコードの強調表示と非強調表示を交互に切り替える交互切替鈕です。扱われたソースコードは緑色で強調表示され、赤色を持つ残りのソース行は現在の実行に対して扱われなかったソースコードを表します。

図4-12. コード強調表示



注: コンパイルできる行だけが考慮に入れます。例えば、注釈と変数宣言を持つ行は無視されます。

4.19.1.5. 追跡表示部設定

デバイスはSRAMで追跡情報を記録するように構成設定されなければなりません。プログラム追跡を記録するのに割り当てられる大きさはこの設定から構成設定することができます。メモリは以下で割り当てられ得ます。

- ・ソースコード - 全域配列を割り当て
- ・リンクスクリプト - メモリ配置の量を予約

プログラム追跡を記録するために割り当てられるべきメモリの大きさを選んでください。CopyToClipboard(クリックホードに複製)鈕をクリックすることによって構成設定ウィンドウで表示されるコードの断片を複製してあなたのソースコード内に張り付けてください。追跡能力を許可するためにダイアログで与えられる命令に従ってください。

注: リンクスクリプトとソースコードの両方が追跡用に構成設定される場合、リンクスクリプト設定がソースコード設定を超える高い順位になります。

図4-14. リンクスクリプトを通す追跡設定ウィンドウ

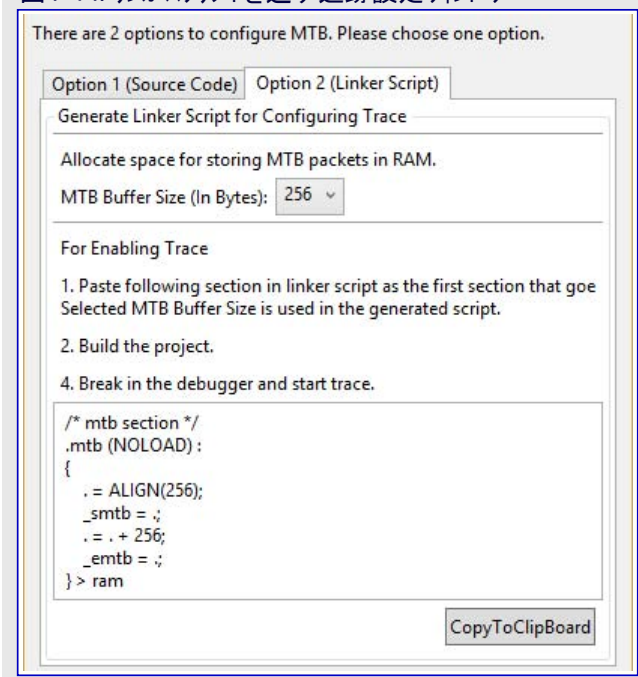
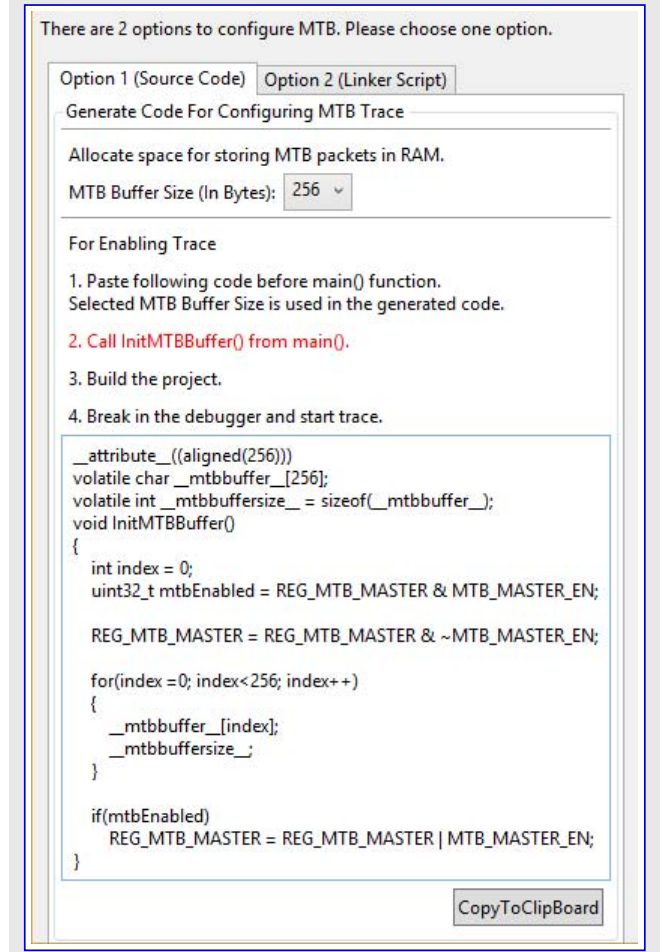


図4-13. ソースコードを通す追跡設定ウィンドウ



4.19.2. 追跡表示部解釈

Trace(追跡)表示ウィンドウは以下の項目を含みます。

- Trace Stack(追跡スタック)表示部
- Coverage(網羅率)表示部

4.19.2.1. 追跡スタック表示部

Trace Stack(追跡スタック)表示部は目的対象が走行しているか、またはデバッグしている状態の間にプログラム追跡情報を取り入れられます。

追跡スタック表示部はSequence No.(連番)、Source(供給元)とDestination(行き先)のアドレス、Repeat Count(繰り返し数)を含みます。新しいプログラム追跡記録は、例えば関数呼び出しや関数からの復帰として、目的対象で分岐する命令が起こる時に示されます。

- Sequence No.(連番) - 記録順の経緯を保つ番号。この番号は毎回の追跡作業に対してリセットされます。この番号はツールバーの解消ボタンを用いて追跡が解消される時に中断なしで継続されます。
- Source(供給元) - 分岐が起きた命令/ソース行を表します。例えば、関数呼び出しで、Sourceは関数を呼びつつあるソース行です。
- Destination(行き先) - 分岐が起きた先の命令/ソース行を表します。例えば、関数呼び出しで、Destinationは関数の開始行のソースコードです。
- Repeat Count(繰り返し数) - 同じ供給元と行き先の組み合わせが連続して起こされた回数を表します。例えば、同じパケットを記録する遅延がある場合、それは集められ、発生記録回数は繰り返し数と呼ばれます。

Source(供給元)とDestination(行き先)は命令アドレス、関数名、ソースファイル名、行番号を含みます。ソース行を配置することができない場合、命令アドレスだけが与えられます。供給元または行き先上のダブルクリックはエディタでカーソルを適切な行へ誘導します。誘導キーの↑、↓、←、→とTabは追跡記録に対してソース/逆アセンブリ表示部で位置を特定するのに使うことができます。

Trace Stack表示部で示されるプログラム追跡は既定によって最後の20000記録に切り落とされます。この閾値は摺動子を用いて変更することができます。

プログラム追跡記録は分岐する命令が予期されなかった時に黄色で強調表示されます。予期せぬ分岐は通常、いくつかの例外のために起き、例外処理部の入り口と出口が黄色で強調表示されます。例外の内側の分岐は強調表示されません。

図4-15. 例外記録

Sequence No	Source	Destination	Repeat Count
1	0x0000280, main, SAMD21_ABIT.c, 88	0x0000280, main, SAMD21_ABIT.c, 88	
2	0x0000288, main, SAMD21_ABIT.c, 90	0x00001F8, InitMTBBuffer, SAMD21_ABIT.c, 5	
3	0x0000202, InitMTBBuffer, SAMD21_ABIT.c, 15	0x0000216	
4	0x000021A	0x0000204, InitMTBBuffer, SAMD21_ABIT.c, 17	256
5	0x0000220, InitMTBBuffer, SAMD21_ABIT.c, 24	0x000028A, main, SAMD21_ABIT.c, 92	
6	0x0000298, main, SAMD21_ABIT.c, 95	0x000025C, func, SAMD21_ABIT.c, 79	
7	0x000027A, func, SAMD21_ABIT.c, 84	0x000029A, main, SAMD21_ABIT.c, 97	
8	0x00002A4, main, SAMD21_ABIT.c, 97	0x0000300	
9	0x0000302	0x0000388	
10	0x000038A	0x000039A	
11	0x00003A4	0x00003C0	
12	0x00003C0	0x00002A6, main, SAMD21_ABIT.c, 97	
13	0x00002AC, main, SAMD21_ABIT.c, 105	0x000022C, SVC_Handler, SAMD21_ABIT.c, 34	
14	0x0000242, SVC_Handler, SAMD21_ABIT.c, 37	0x000025C, func, SAMD21_ABIT.c, 79	
15	0x000027A, func, SAMD21_ABIT.c, 84	0x0000244, SVC_Handler, SAMD21_ABIT.c, 41	
16	0x0000246, SVC_Handler, SAMD21_ABIT.c, 41	0x00002AC, main, SAMD21_ABIT.c, 105	
17	0x00002B8, main, SAMD21_ABIT.c, 107	0x000025C, func, SAMD21_ABIT.c, 79	
18	0x000027A, func, SAMD21_ABIT.c, 84	0x00002BA, main, SAMD21_ABIT.c, 107	

助言: 次と前の例外記録は追跡表示ウィンドウのツールバーで下矢印を使うことによって容易に誘導することができます。

いくつかのプログラム追跡を落とし得るいくつかの例外的な場合があります。その場合、それらは順番に於いてプログラム追跡情報のいくつかの非連続があることを表す赤色のパケットになります。落としたパケット数が未知のため、Trace Stack表示部で支援される連番は、それに対する連番を持つ赤色にされたパケットの追加を除いてどんな中断もなしも続けられます。

注: 逆アセンブリ表示部は誘導キーが使われる時に支援されませんが、マウスを使って記録がダブルクリックされた時にそれを支援します。

4.19.2.2. 網羅率表示部

Coverage(網羅率)表示部は現在の目的対象実行の一部として網羅されたソースの統計を示します。全てのファイルと関数がソースファイル内の総行数に対して網羅された、または網羅されない行数の情報と共に網羅率表示部で一覧にされます。

図4-16. コード網羅率

Name	Coverage (%)	UnCovered/Total Lines
cmsis/src/startup_samd21.c	0%	15/15
Reset_Handler	0%	13/13
Dummy_Handler	0%	2/2
cmsis/src/system_samd21.c	100%	0/4
GccApplication12.c	28%	28/39
InitMTBBuffer	100%	0/9
insert	0%	6/6
create	0%	5/5
insertnode	0%	10/10
main	22%	7/9

注: 統計に関してコンパイル可能な行だけが考慮に入れます。例えば、注釈と変数宣言を持つ行は考慮されません。

網羅率報告はエクスポートすることができます。エクスポート動作を呼び出すには追跡表示部ツールバーで エクスポート アイコンをクリックしてください。

5. プログラミング ダイアログ

5.1. 序説

(プログラミング ダイアログとしても知られる)Device Programming(デバイスプログラミング)ウィンドウはデバッグとプログラミングのツールの最も下位の制御を与えます。それでデバイスの各種メモリ、ヒューズ、施錠ビットのプログラミング、メモリ消去、使用者識票書き込みを行うことができます。電圧とクロック生成部のようなスタートキット特性のいくつかを調整することもできます。

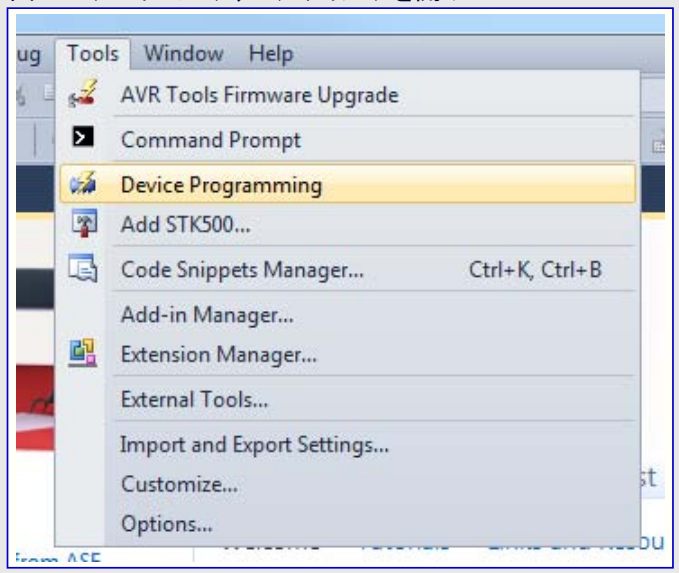
注: Microchip Studioでコードプロジェクトを編集してデバイス内にそのコードをダウンロードすることによってコンパイルの結果を見たい場合、**Satrt without Debugging**(デバッグなしで開始)命令で見てください。それはプログラミング ダイアログに代わる一種の1クリックプログラミングです。より多くの情報については「4.5. デバッグなしでの開始」項をご覧ください。

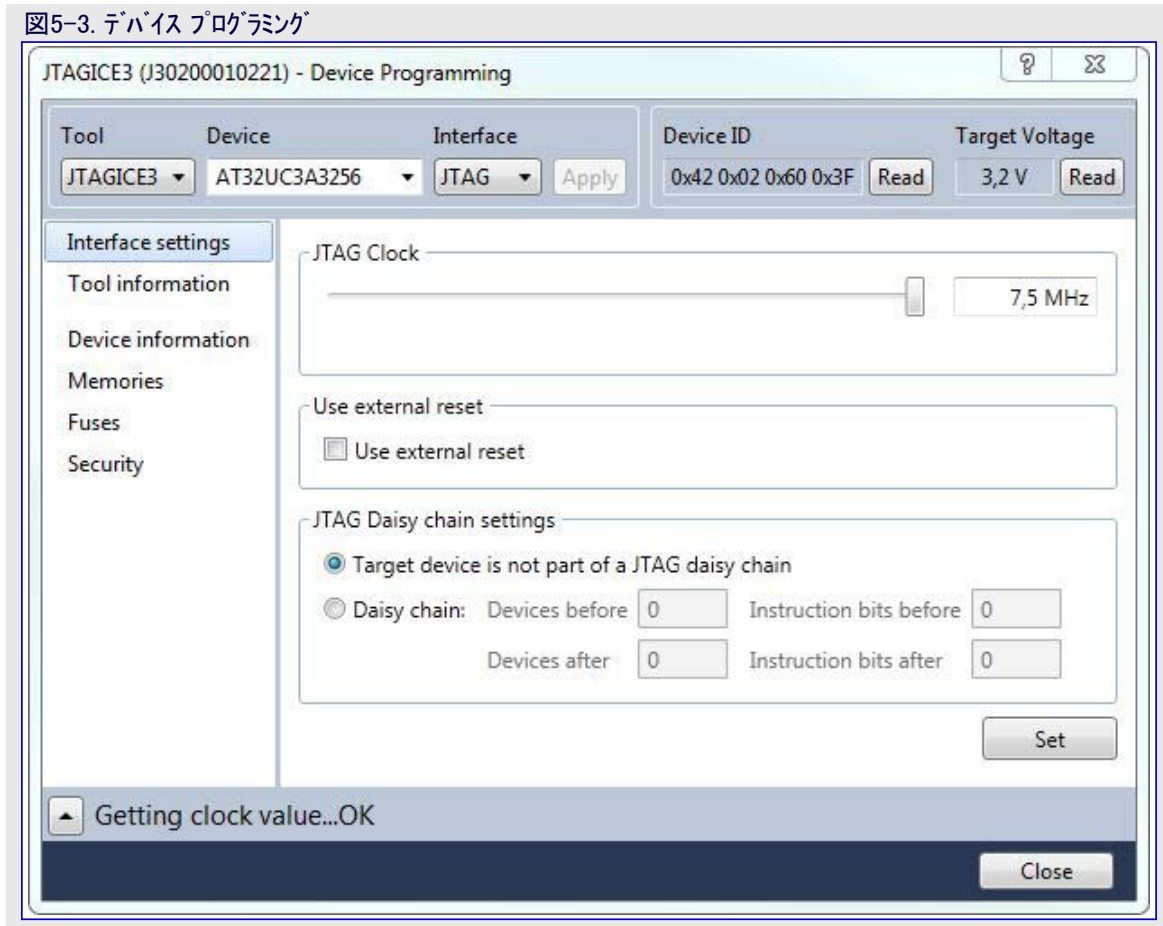
プログラミング ダイアログは標準ツールバー上の釘、またはメニューのTools(ツール)⇒Device Programming(デバイスプログラミング)からアクセス可能です。

図5-1. デバイスプログラミング アイコン



図5-2. デバイスプログラミング ダイアログを開く





プログラミング ダイアログは以下の任意選択とタブを含みます。

上部状態バー

Tool (ツール)

この引き落としメニューから使いたいツールを選ぶことができます。PCに接続されたツールだけが一覧にされます。また、ツールがデバッグ作業で使われている場合、それは一覧にされません。同じ形式のいくつかのツールを同時に接続することができます。それらを識別するため、一覧のツール名の下に通番が示されます。

ツール選択時、名前(と通番)がデバイスプログラミング ダイアログのタイトル バーで示されます。

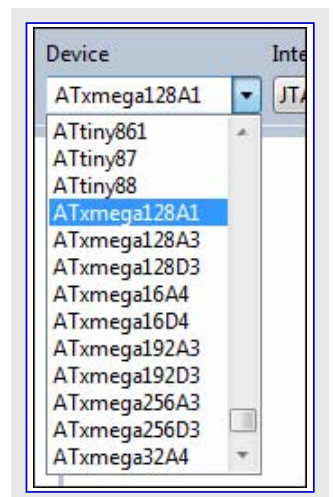
注: シミュレータはプログラミング ダイアログ機能に対して限定された支援だけを提供します。シミュレータは持続性メモリを持たず、故にシミュレートするどのデバイスに対しても恒久的に変更することはできません。

Device (デバイス)

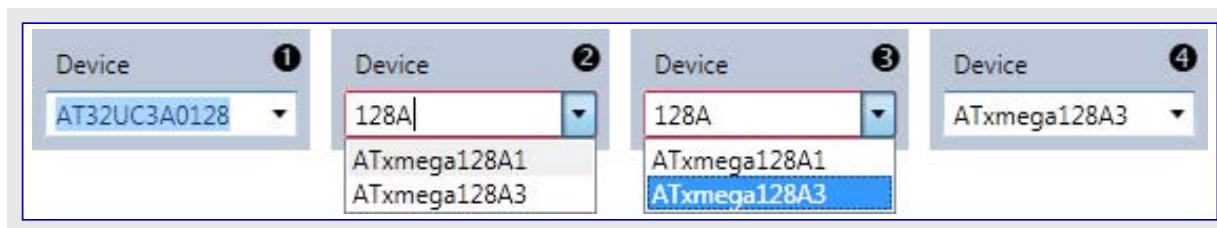
ツールが選ばれると直ぐに、デバイス一覧がそのツールによって支援される全てのデバイスを示します。デバイスを選ぶのに2つの方法があります。

- 一覧から選んでください。

矢印をクリックしてください。これは支援するデバイスの一覧を見せます。選ぶためにクリックしてください。



- 入力することによって選んでください。この例ではATxmega128A3を選びます。



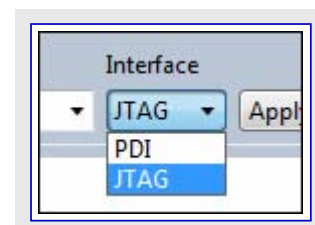
- 既に存在する文字を選択するようにテキスト領域でダブルクリックしてください。
- デバイス名の或る部分の入力を開始してください。この例では128Aです。入力中に一覧が更新され、入力されたものを含む全てのデバイスを示します。
- 一覧内で選択を移動するためにキーボードの↓を押してください。誘導するには↑と↓のキーを使ってください。選択するにはEnterを押してください。
- 今やATxmega238A3が選ばれます。

注: デバイス選択部周辺の赤境界は入力された文字が有効なデバイス名でないことを示します。デバイス名が完了するまで入力を続けるか、または一覧から選んでください。

Interface (インターフェース)

ツールとデバイスが選ばれると、インターフェース一覧が利用可能なインターフェースを示します。ツールとデバイスの両方で利用可能なインターフェースだけがこのメニューに現れます。

デバイスをプログラミングするのに使うインターフェースを選んでください。



Apply(適用)釦

ツール、デバイス、インターフェースが選ばれると、選択を有効にするためにApply(適用)釦を押してください。これはツールへの接続を確立します。ウィンドウの左側の一覧は選んだツールに関連するページで更新されます。

違うツール、デバイス、またはインターフェースが選ばれた場合、新しい選択を有効にするために再びApply(適用)釦が押されなければなりません。

Device ID (デバイスID)

デバイスから識別票ハバを読むにはRead(読み込み)釦を押してください。デバイスの固有標識がこの領域に現れ、ツールの適合性調査と、お客様支援またはAVR Freaks®の人々のどちらかからの手助けを得るのに使うことができます。

Target Voltage (目的対象電圧)

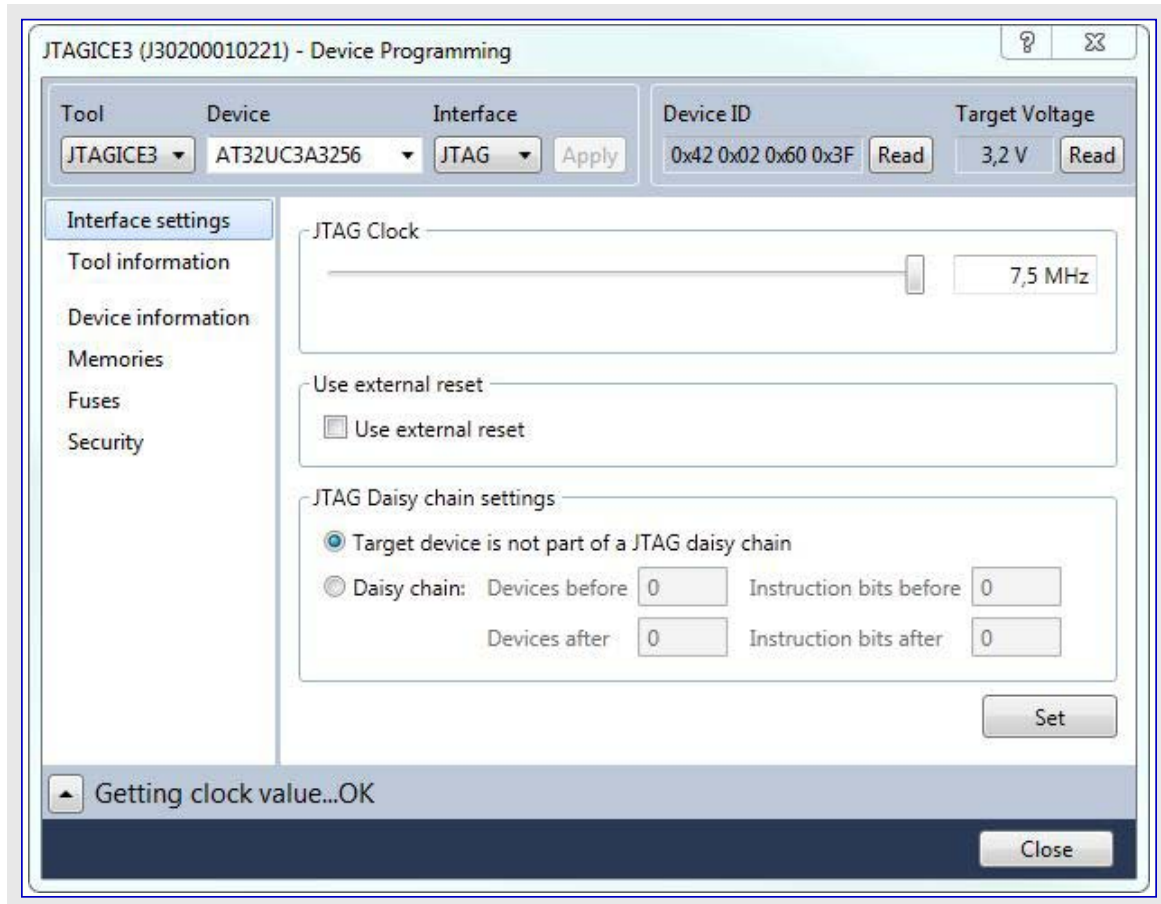
全てのツールは目的対象の動作電圧を測定する能力があります。新しい測定にするには更新Read(読み込み)釦を押してください。測定した電圧が選んだデバイスに対する動作電圧の外側の場合に警告メッセージが現れ、目的対象枠が赤に切り替わります。

5.2. インターフェース設定

プログラミング インターフェースは異なる設定を持ちます。いくつかのインターフェースは全ての設定を持たない一方で、いくつかのインターフェース設定はいくつかのツールでだけ利用可能です。この項は全ての設定を記述しますが、それらは全てのツールとデバイスで利用可能な訳ではありません。

JTAG

プログラム インターフェースとしてJTAGを選んだ場合、クロック速度、外部リセットの使用、デジタイズチェーン設定が利用可能かもしれません。これはツールとデバイスに依存します。



JTAG Clock (JTAGクロック)

JTAGクロックはツールがデバイスをクロック駆動しようとする最大速度です。クロック範囲は各種ルールとデバイスに対して異なります。制限がある場合、それらはクロック摺動子の下のメッセージで述べられます。

Use external reset (外部リセットを使用)

チェックされると、ツールはデバイスに接続しようとする時に外部リセット線をLowに引きます。

JTAG Daisy chain settings (JTAGデジタイズチェーン設定)

プログラミングのためにデバイスに関連するJTAGデジタイズチェーン設定を指定してください。

Target device is not part of a JTAG daisy chain (目的対象はJTAGデジタイズチェーンの一部ではありません。)

目的対象デバイスがデジタイズチェーンの一部でない時にこの任意選択を選んでください。

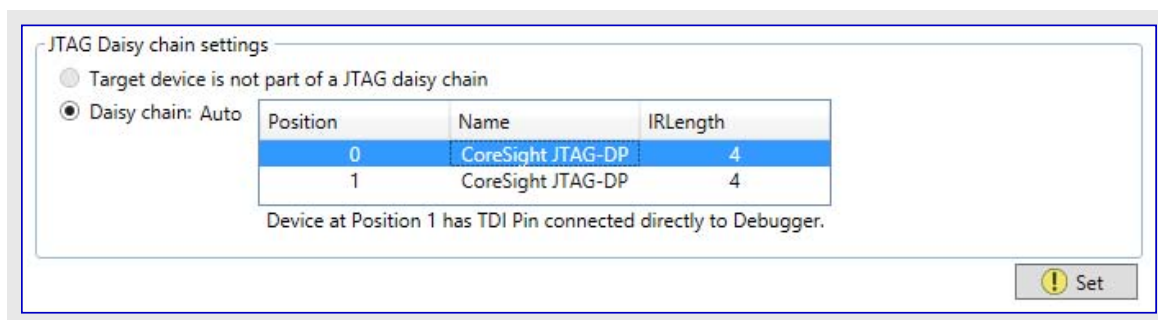
Daisy chain - 手動 (デジタイズチェーン：手動設定)

基板上システムでプログラミングする場合、手動でJTAGデジタイズチェーンを構成設定することを許します。

- **Devices before** - 目的対象デバイスに先行するデバイス数を指定してください。
- **Instruction bits before** - 目的対象デバイスに先行する全デバイスの命令レジスタの総(ビット)数を指定してください。
- **Devices after** - 目的対象デバイスに後続するデバイス数を指定してください。
- **Instruction bits after** - 目的対象デバイスに後続する全デバイスの命令レジスタの総(ビット)数を指定してください。

Daisy chain - 自動 (デジタイズチェーン：自動設定)

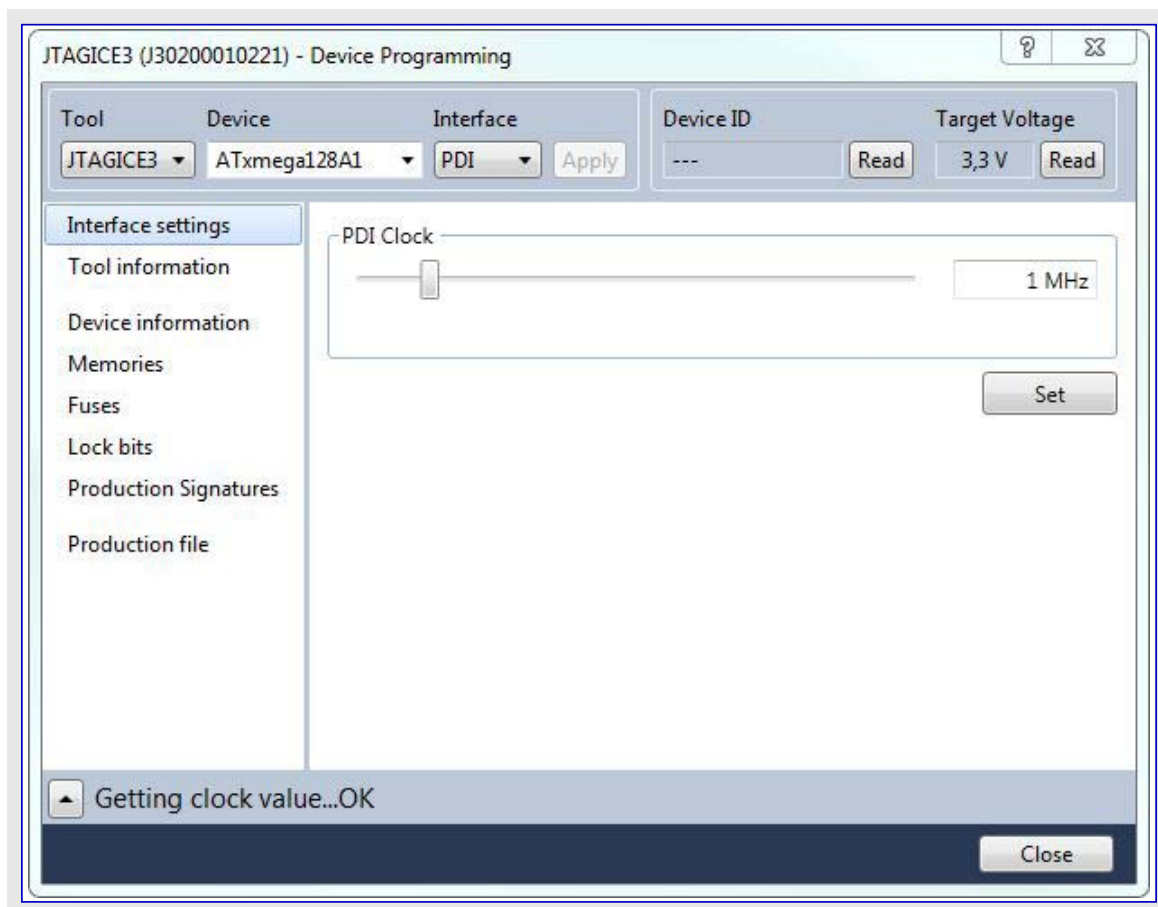
JTAGデジタイズチェーンでデバイスを自動的に検出します。JTAGデジタイズチェーンでデバイスを選ぶことを許します。自動検出はSAMデバイスに対してだけ支援されます。



変更を受け入れてツールを構成設定するにはSet(設定)釦を押してください。

PDI

PDIインターフェースはPDIクロック速度の1つの設定だけを持ちます。

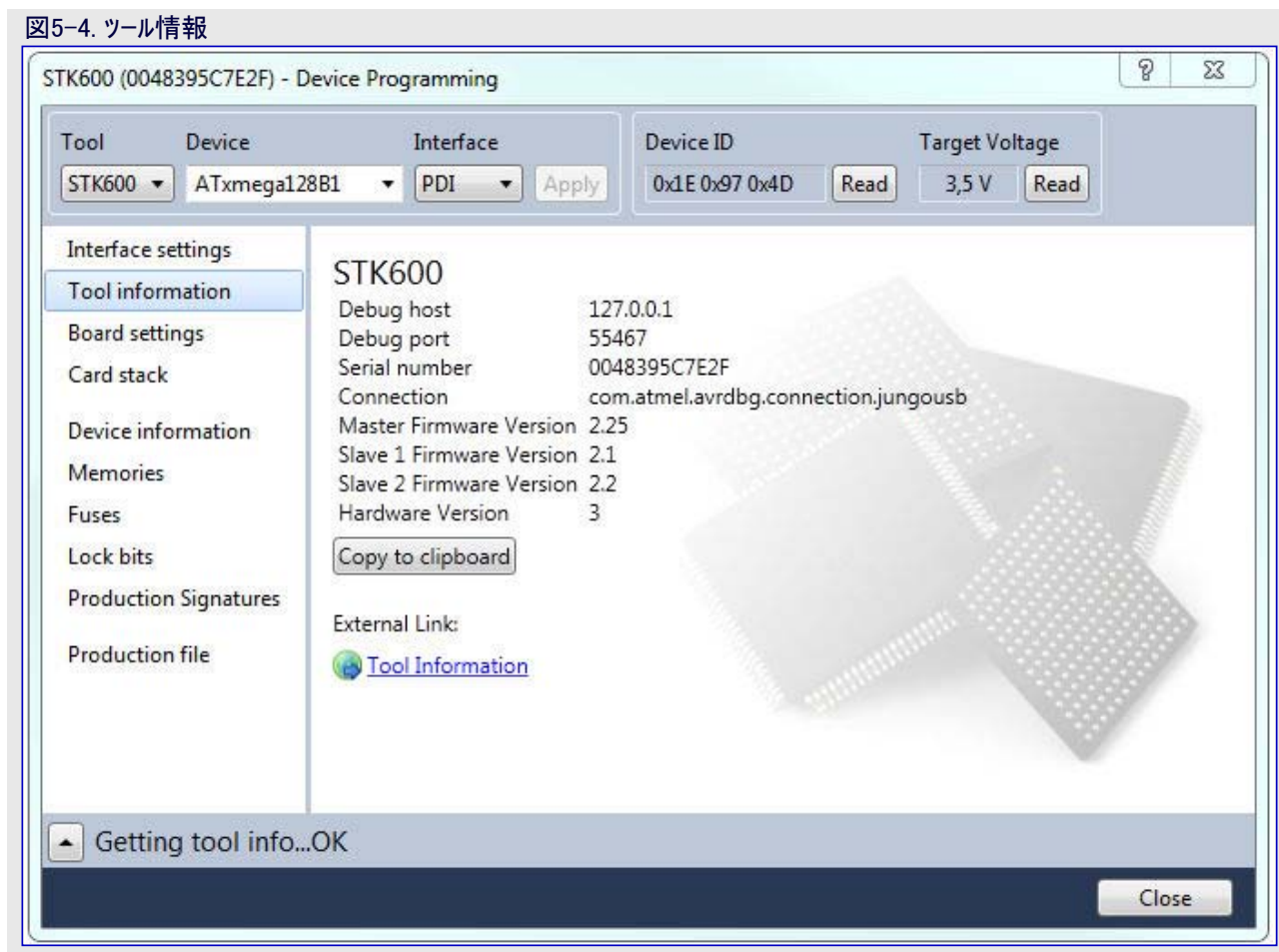


PDI Clock(PDIクロック)はデバイスをクロック駆動しようとする最大速度です。クロック範囲は各種ルールとデバイスに対して異なります。制限がある場合、それらはクロック摺動子の下のメッセージで述べられます。

変更を適用してツールを構成設定するにはSet(設定)釦を押してください。

クロックは全てのツールで調整することができず、故に空のInterface settings(インターフェース設定)ページが提示されます。

5.3. ツール情報



Tool information(ツール情報)ページはいくつかの有用なツール項目を含みます。

ツール名は接続したツールに対する一般名を示します。

Debug host(デバッグ ホスト)は遠隔デバッグの場合のデバッグ作業のホストIPアドレスです。ツールがPCに接続される場合、それはループバックインターフェースIP(127.0.0.1)が見えます。

Debug port(デバッグ ポート)は遠隔デバッグがデバッグ ツールにアクセスするために特に開かれたポートです。このポートはMicrochip Studio開始時に自動的に割り当てられ、常は4711です。

Serial number(通番) - ツールの通番

Connection(接続) - ツールをPCに接続するのに使ったMicrosoft Driver枠組み法の名前

xxx version(xxx版本号) - ファームウェア、ハードウェア、FPGAファイル版本号がここで一覧にされます。

ダイアログのリンクを用いてツール オンラインで広範囲な情報にアクセスすることができます。

5.4. 基板設定/ツール設定

いくつかのツール(Power Debugger、STK500、STK600、QT600)は基板上の電圧とクロックの生成部を持ちます。これらは基板設定/ツール設定のページから制御することができます。

5.4.1. Power Debugger

Power Debuggerは単一電圧源と、電圧/電流測定用の2つのチャネルを持ちます。

電圧出力(VOUT)は摺動子、または摺動子の下のGenerated(生成)テキスト枠に電圧を入力することによって調整されます。

設定点調節後、その変更を適用するためにWrite(書き込み)釦を押してください。値はその後にツールへ送られて、Measured(測定)値が読み戻されます。

Power Debuggerから設定点(Generated)とMeasured値の両方を読むにはRead(読み込み)釦を押してください。

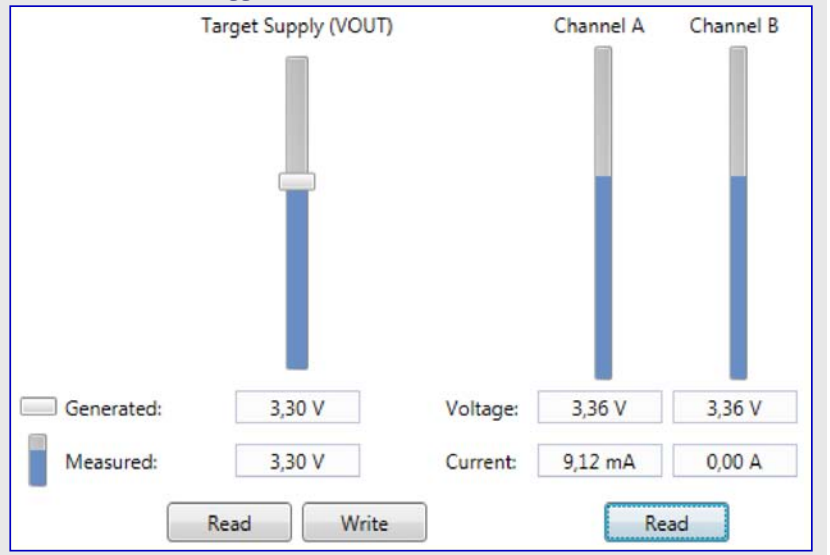
注: GeneratedとMeasuredの電圧間は僅かに違うかもしれません。

出力範囲は1.6~5.5Vです。

Channel A(チャネルA)とChannel B(チャネルB)の測定はPower Debuggerによって取られたアナログ読み取りの瞬撮です。このツールは電圧と電流の実時間監視用に最適化され、故にこの瞬撮はほぼ正確です。これは校正補償を実行せず、読み取りは最高電流範囲で固定化されます。最良の結果のため、Atmel Data Visualizer(データ可視器)を使ってください。

注: 測定チャネルに負荷が全く接続されない時に、0以外の測定が予想されます。

図5-5. Power Debuggerツール設定



5.4.2. STK600

STK600は3つの電圧源と1つのクロック生成部を持ちます。

3つの電圧源(VTarget、ARef0、ARef1)は3つの摺動子の方法によって調整されます。これは摺動子の下のGenerated(生成)テキスト枠に電圧を入力することでも可能です。摺動子を引き摺ると、テキスト枠が更新します。そしてテキスト枠に値を入力すると、摺動子が動きません。

設定点調節後、その変更を適用するためにWrite(書き込み)釦を押してください。値はツールへ送られ、Measured(測定)値が読み戻されます。

測定はMeasured行で示され、摺動子制御の一部として示されます。

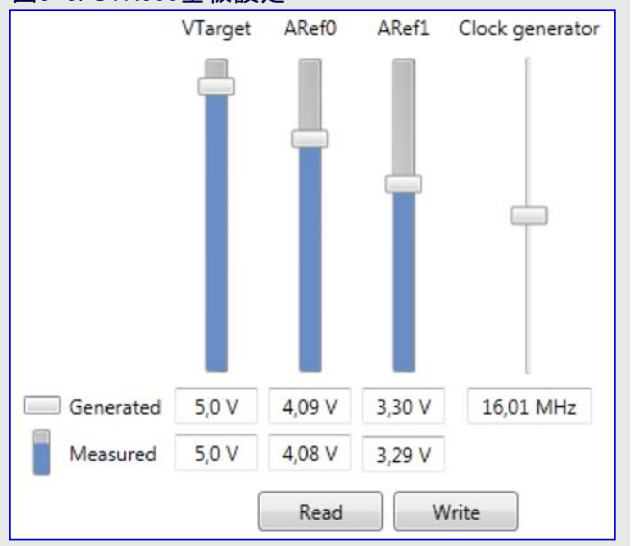
STK600から設定点(Generated)とMeasured値の両方を読むにはRead(読み込み)釦を押してください。

注: GeneratedとMeasuredの電圧間の違いは何?。Generated(指定した)電圧は調整可能な電源で設定します。測定した電圧は組み込み電圧計から読み出されます。測定値が指定電圧と異なる場合、これは目的対象回路が電圧生成部から沢山の電流を引き出していることを示しているかもしれません。

注: STK600のVTARGETジャンパが実装されない場合、VTARGET網に外部電圧が印加されない限り、測定した電圧は0になるでしょう。

Clock generator(クロック生成部)は摺動子を引き摺るか、または下のテキスト枠に入力することによっても調整することができます。新しい値を適用するためにWrite(書き込み)釦を押してください。

図5-6. STK600基板設定



5.4.3. QT600

QT600はVtarget電圧の1つの設定だけを持ちます。この電圧は0、1.8、2.7、3.3、5Vの5つの固定電圧に設定することができます。変更を適用するにはWrite(書き込み)釦を押してください。

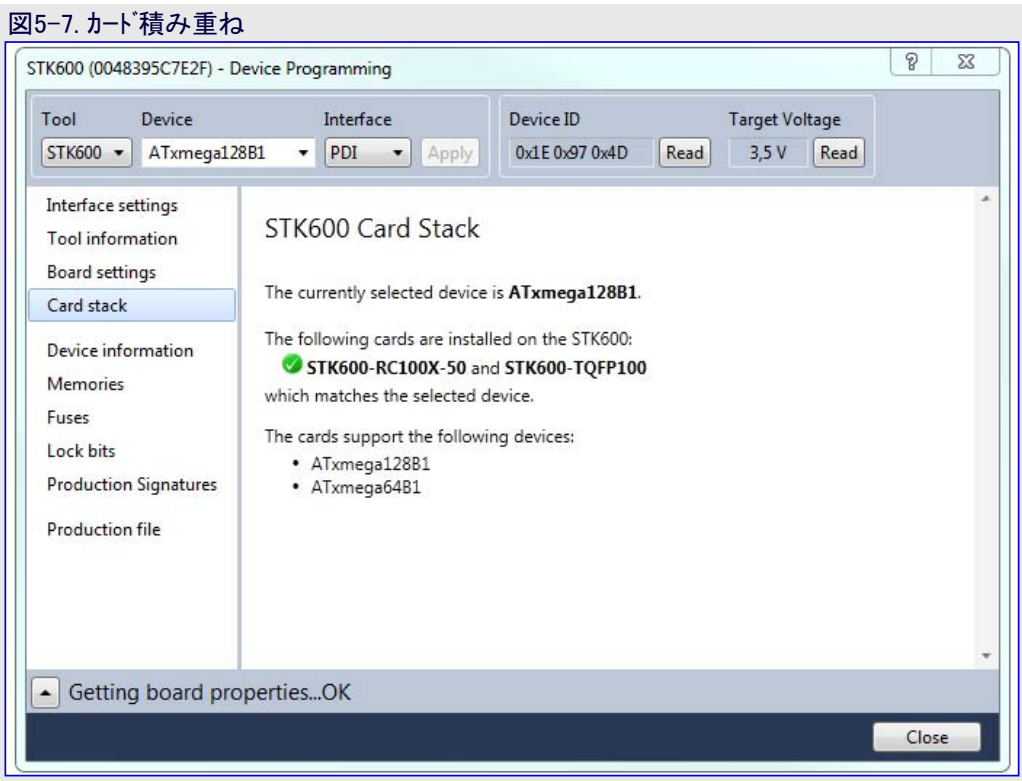
実際のVtarget値はWrite釦押下時に自動的に読み戻されます。これはRead(読み込み)釦を使って手動で読み戻すことも可能です。

5.4.4. STK500

STK500はSTK600と同じ設定を持ちますが、1つのAref電圧だけで組み合わせられた生成(指定)/測定値です。

5.5. カード積み重ね

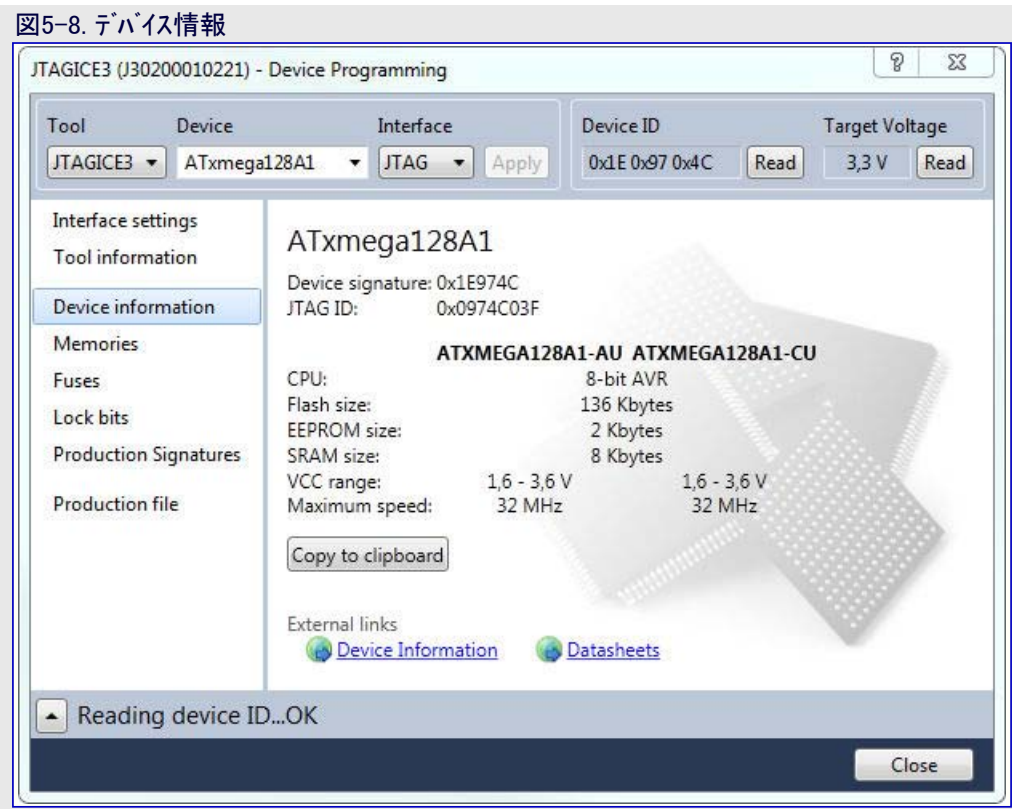
STK600は全てのAVRデバイスに乗らせるように配線とソケットのカードの組み合わせを使います。与えられたデバイスは配線とソケットのカードの或る組み合わせだけが有効です。Card stack(カード積み重ね)ページはこれについての情報を持ちます。



Card stack(カード積み重ね)ページはSTK600でどのカードが乗せられるかとそれらが選んだデバイスを支援するかを告げます。それらが一致なら、そのカードの組み合わせによって支援されるデバイスのが一覧にされます。

乗せられたカードが一致しない場合、示唆されるカードの組み合わせが一览にされます。

5.6. デバイス情報



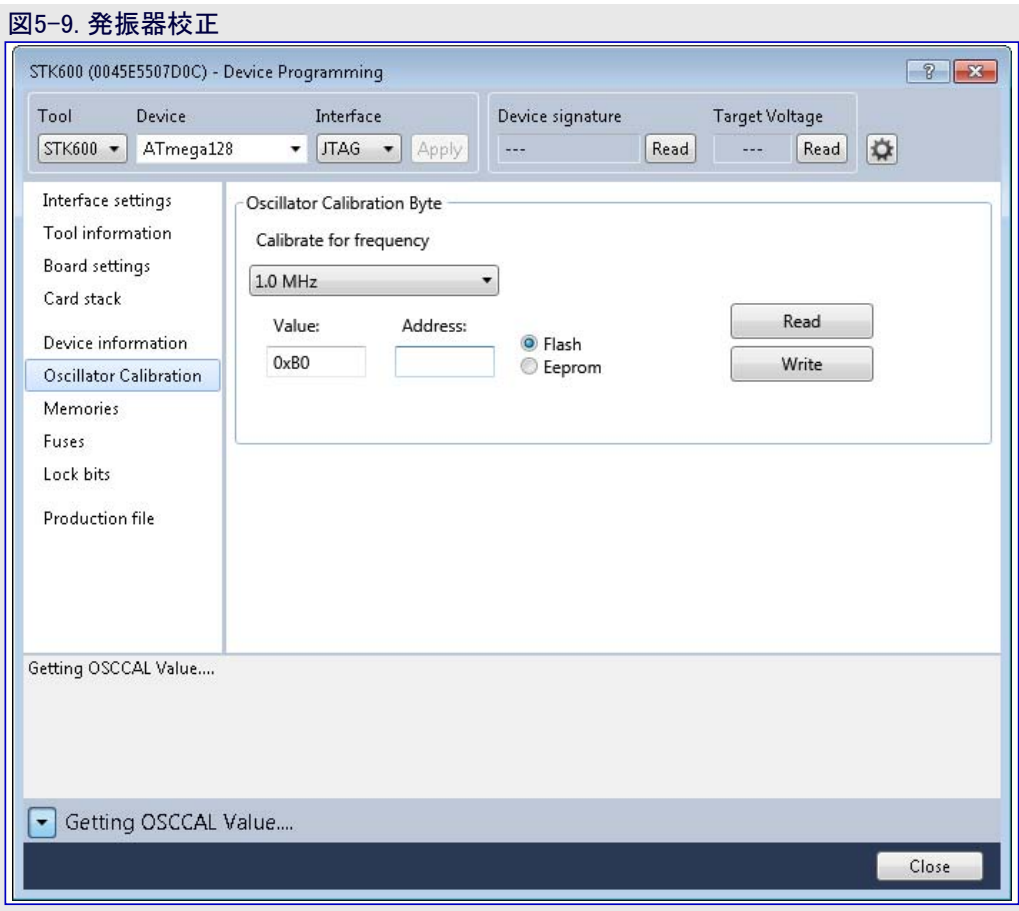
Device information(デバイス情報)ページは選んだデバイスの基本的な情報を含みます。このページがアクセスされると、接続したデバイスからJTAG(またはデバイス)識票を読もうとします。

ダイアログの上部では、デバイス名、その識票、JTAG部識別番号、(JTAG識票から抽出された)デバイス改訂を見ることができます。

ダイアログの下部では、デバイス変種と各変種の特性、使うことができるチップ上のメモリ量、受け入れ可能な電圧範囲、その後に最大動作クロック速度を見ることができます。

ダイアログの2つのリンクは購入カタログ オンラインでちょっとより多くの詳細なデバイス情報を見ること、または目的対象デバイスの完全なデータシートをダウンロードすることを提供します。

5.7. 発振器校正



ATtinyとATmegaデバイス用Oscillator Calibration Byte (発振校正バイト)

Advanced(詳細)タブからATtinyとATmegaデバイス用の発振校正バイトを読むことができます。発振校正バイトは可能な限り選んだクロック周波数近で走行するように内部RC発振器を調整するために、選んだデバイスで見つかる発振校正(OSCCAL)レジスタに書くことができます。

プログラミング

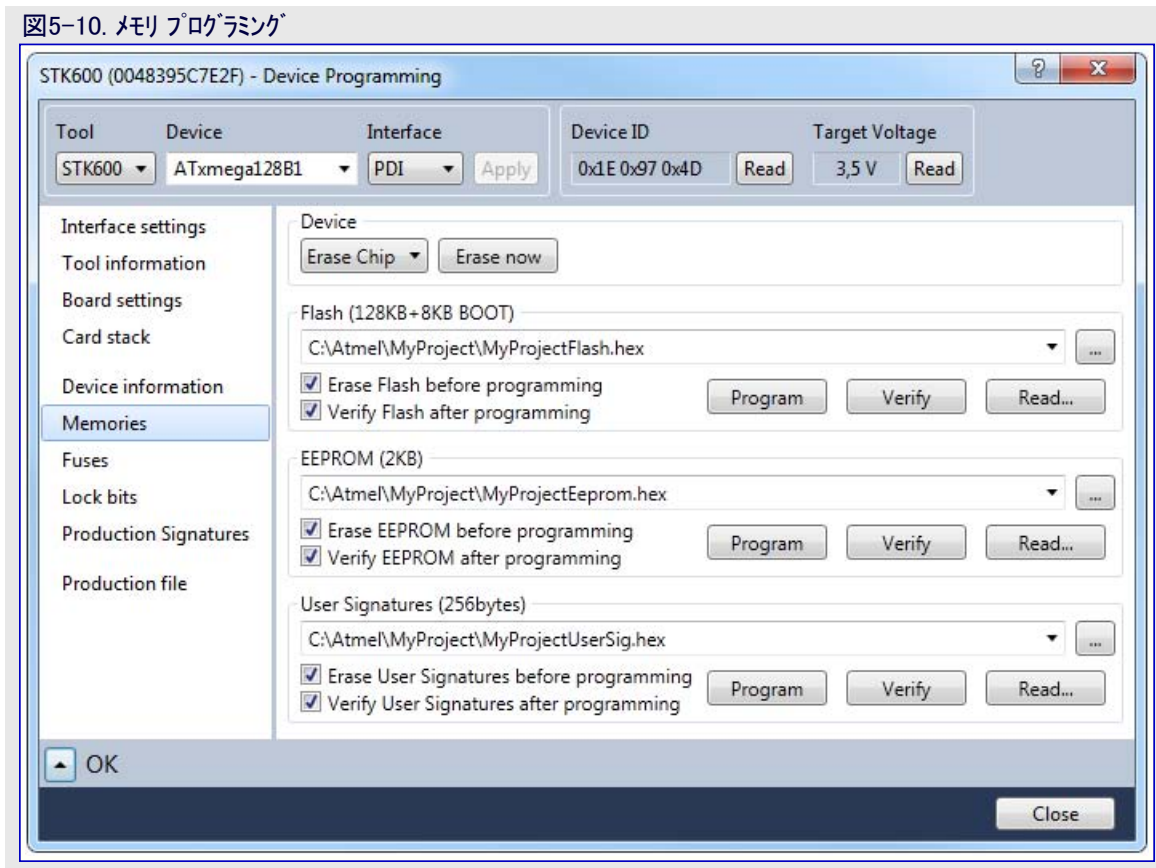
発振校正バイトは製造中にデバイス内に格納され、使用者によって消去や変更をすることができません。これはデバイスに依存して、デバイス始動中にOSCCALレジスタに自動的に転送されるか、またはプログラム初期化中に設定してください。プログラム初期化中にそれを応用が設定するデバイスでは、プログラミング ダイアログがコマンド行ツールを用いて、それが先にフラッシュメモリかEEPROMに転送されなければなりません。

ATtinyとATmegaデバイス用発振校正バイトの読み込みと書き込み

校正値はRead(読み込み)ボタンによってデバイス内の記憶域から読まれてValue(値)テキスト枠で表示されます。

校正バイトはWrite(書き込み)ボタンによってフラッシュメモリかEEPROMに書かれます。メモリ形式とアドレスが先に指定されなければなりません。

5.8. メモリ



Memories(メモリ)タブから目的対象デバイスで書き込み可能な全メモリをアクセスすることができます。メモリは最初にメモリ形式を選び、その後にErase(消去)鈕をクリックすることによって消去されます。Erase Chip(チップ消去)を選ぶと、デバイスが使用者ページを含む場合にそれを除き、フラッシュメモリ、(EESAVEヒューズが設定されていなければ)EEPROM、施錠ビットを含むデバイスの全体内容を消去します。

書き込み

デバイスのフラッシュメモリにファイル(内容)を書くには、Flash(フラッシュメモリ)領域内のコンボイ枠に完全なパスとファイル名を書いてください。または、検索(...)鈕を押下することによってファイルを選んでください。

次に、ファイル(内容)をメモリに書くためにProgram(書き込み)鈕を押してください。

Erase xxxx before programming(書き込み前にxxxxを消去)チェック枠がチェックの場合、書き込み操作開始前にxxxx消去操作が実行されます。

Verify xxxx after programming(書き込み後にxxxxを検証)チェック枠がチェックの場合、書き込み操作終了後に内容が検証されます。

いくつかのデバイスはフラッシュローダを通して書くこともでき、これは主に高度な技法ですが、通常、書き込み速度でかなりの速度向上を与えます。これが支援されるデバイスについては、Program flash from RAM(RAMからのフラッシュ書き込み)と名付けられたチェックボックスが示されます。この枠がチェックされる場合、フラッシュローダの位置の基準アドレスが与えられることが必要です。

検証

デバイスのフラッシュメモリとファイル内容を検証するには、最初にそれに対して検証したいファイルを選び、その後にVerify(検証)鈕を押してください。

読み込み

フラッシュメモリの内容はRead(読み込み)鈕を用いてIntel®16進ファイル形式で読み出すことができます。Read鈕押下はファイルが保存される場所を指定することを申し入れるダイアログを提示します。

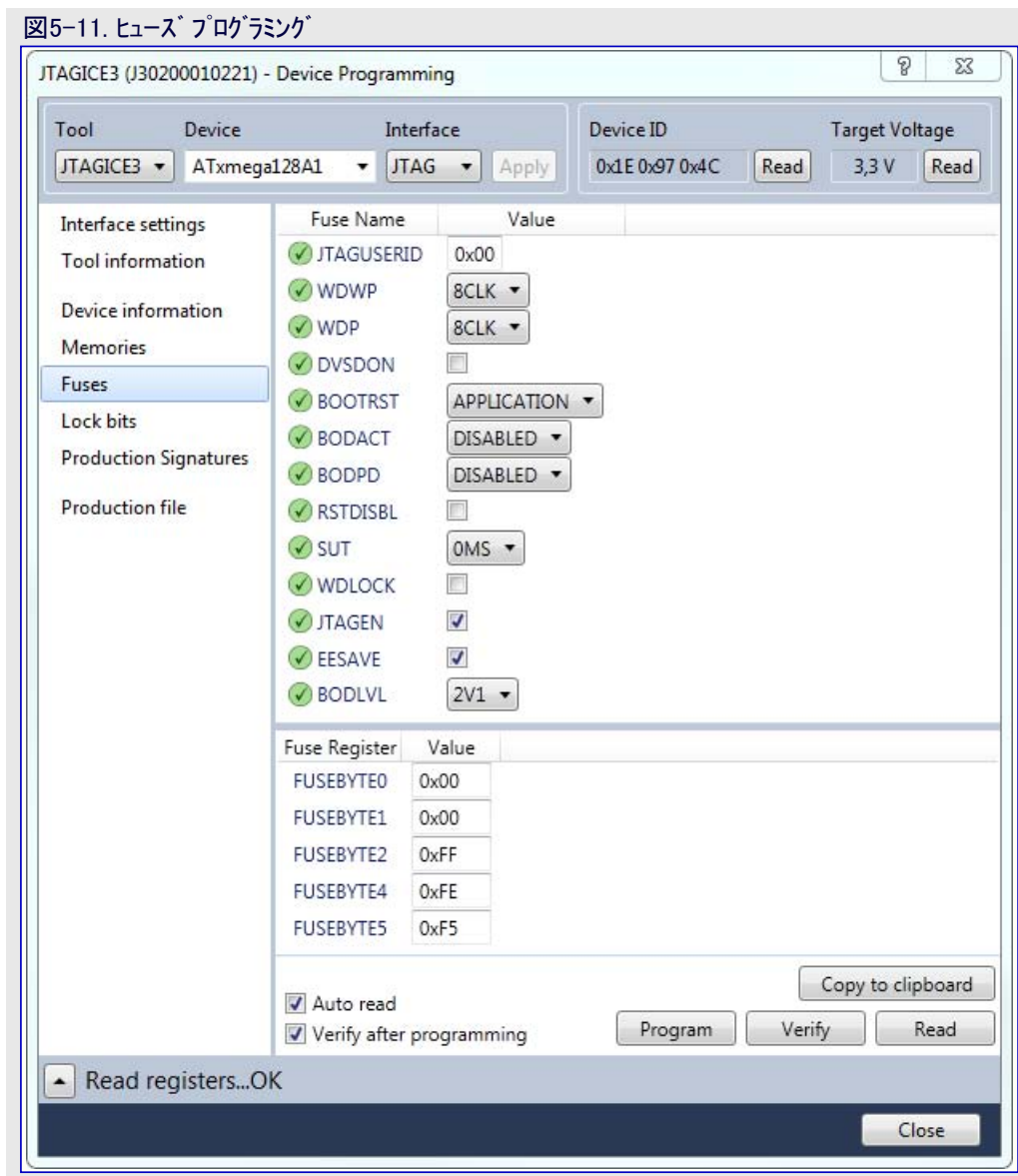
EEPROM

デバイスのEEPROMは同様にプログラミングすることができます。

使用者識票

XMEGAデバイスのUser Signature(使用者識票)は同じ方法でプログラミングすることができます。

5.9. ヒューズ プログラミング



Fuses(ヒューズ)ページは選んだデバイスのヒューズを提示します。

ヒューズの現在値を読むにはRead(読み込み)鈕を押し、現在のヒューズ設定をデバイスに書くにはProgram(書き込み)鈕を押ししてください。ヒューズ設定はチェック枠または引き落とし一覧として提示されます。

各種プログラミング動作で利用可能なヒューズの詳細な情報とそれらの機能はデバイスのデータシートで見つけることができます。選んだヒューズ設定がチップ消去周期でデバイスを消去すること(例えば、Memories(メモリ)ページでのErase Chip(チップ消去)鈕押下)によって影響を及ぼされないことに注意してください。

ヒューズ値は下側枠で16進値としてヒューズレジスタに直接書くこともできます。

Auto read (自動読み込み) : このチェック枠がチェックされる場合、Fusesページに入る度にデバイスからヒューズ設定が読まれます。

Verify after programming (書き込み後に検証) : このチェック枠がチェックされると、書き込み操作が完了した後にその設定が検証されます。

ヒューズ絵記号の外観はヒューズ情報がデバイスの状態と比べて最新かどうかを説明します。

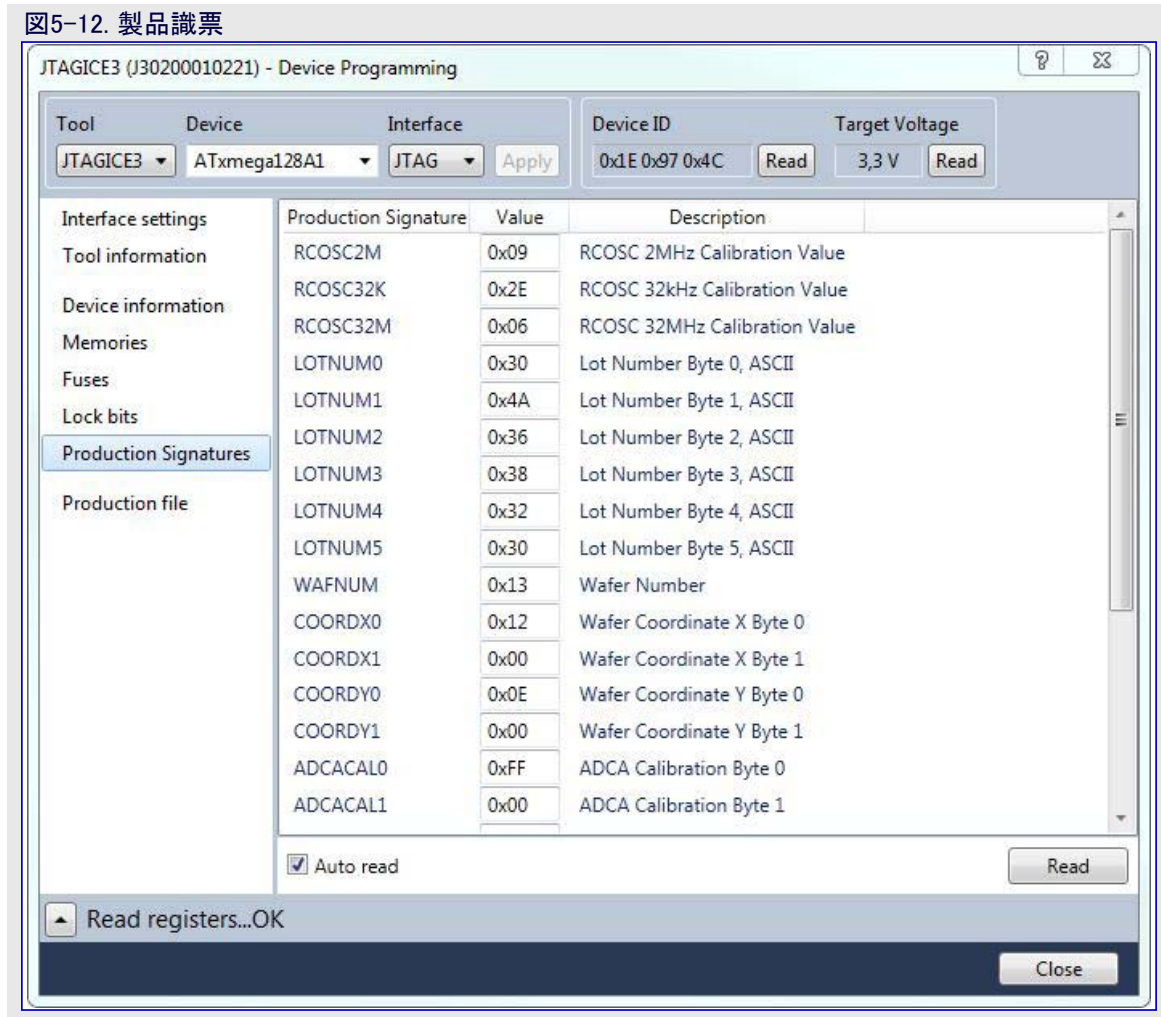
- ✔ ヒューズ値は最新です。即ち、デバイスと同じ状態です。
- ! ヒューズは使用者によって変更され、未だデバイスに書かれていません。
- ? ヒューズの状態は未知です。デバイスから読めなく、また、使用者によって変更されません。

5.10. 施錠ビット

Lock bits(施錠ビット)ページはFuses(ヒューズ)ページと同様です。使い方については「5.9. ヒューズ プログラミング」項をご覧ください。

5.11. 製品識票

Production Signatures(製品識票)ページはAVR XMEGAデバイスに対してだけ見え、製品識票列内の工場書き込みされたデータを示します。これは発振器とアナログ単位部のような機能に対するデータを含みます。製品識票列は書くことや消去することができません。

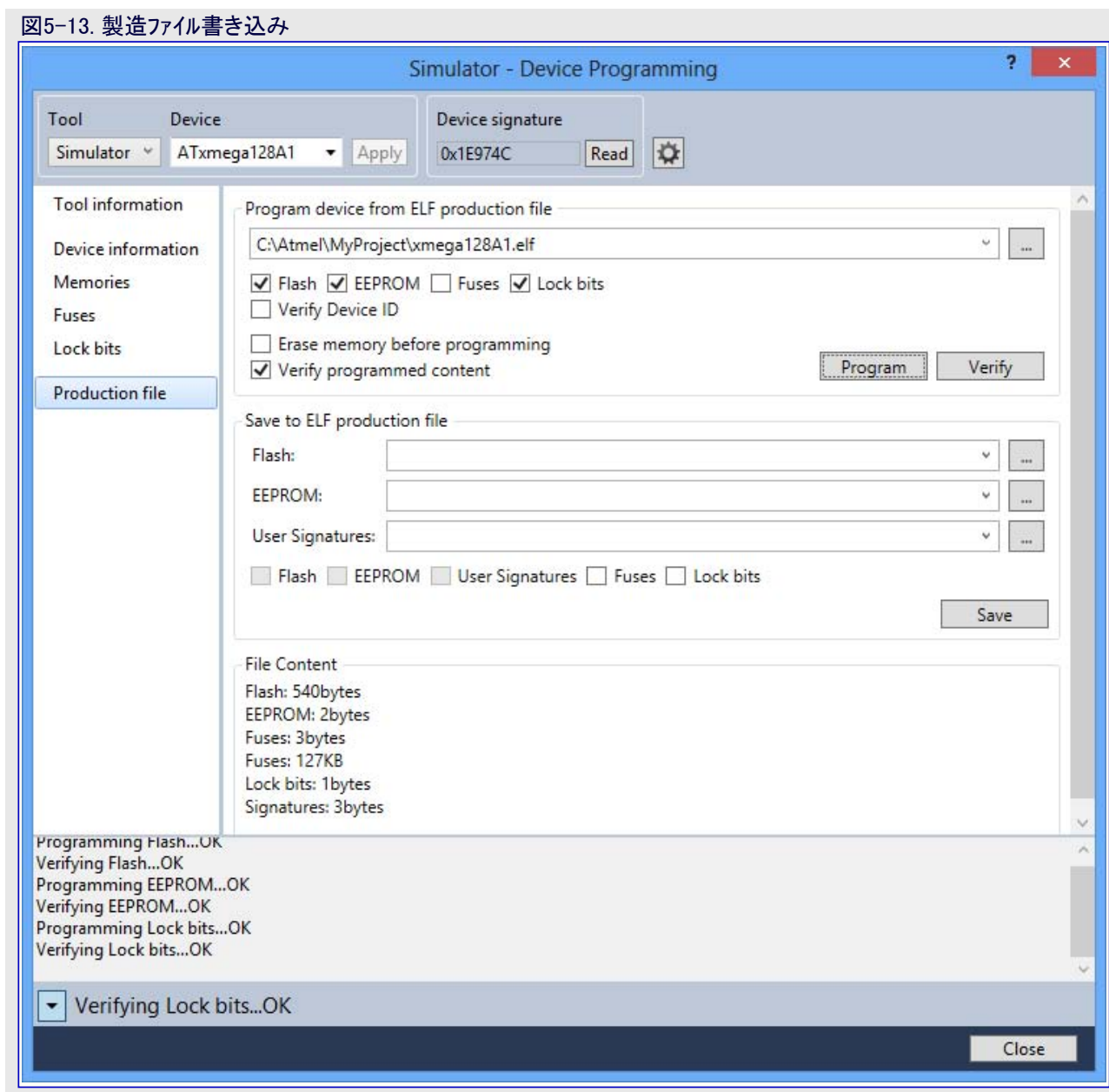


5.12. 製造ファイル

ELF製造ファイル形式は1つの単一ファイルでフラッシュメモリとEEPROMの両方、使用者識票(XMEGAデバイスのみ)だけでなく、ヒューズと施錠ビットの構成設定の内容をも保持することができます。この形式は実行可能/リンク可能形式(ELF:Executable and Linkable Format)に基づきます。

製造ファイル形式は現在、tinyAVR、megaAVR、XMEGAが支援されます。このようなファイルを生成するようにプロジェクトを構成設定する方法の記述については「[3.2.7.7. 他のメモリ形式でのELFファイル作成](#)」をご覧ください。

図5-13. 製造ファイル書き込み



Program device from ELF production file (ELF製造ファイルからデバイス書き込み) : ELFファイルからデバイスを書き込むには最初にコンボイにその完全なパスを入力するか、または検索 [...] 鈕を押すことによってソース ファイルを選ばなければなりません。ファイルの内容に依存して、各種メモリ区分に対するチェック枠が活性化されます。

ELF製造ファイルが含むメモリ区分の1つまたはいくつかを選ぶことが可能です。その後、1つの単独操作でこれらの区分の内容でデバイスを書いて検証することができます。対応するチェック枠をチェックすることによって書き込みたいメモリ区分を選んでください。

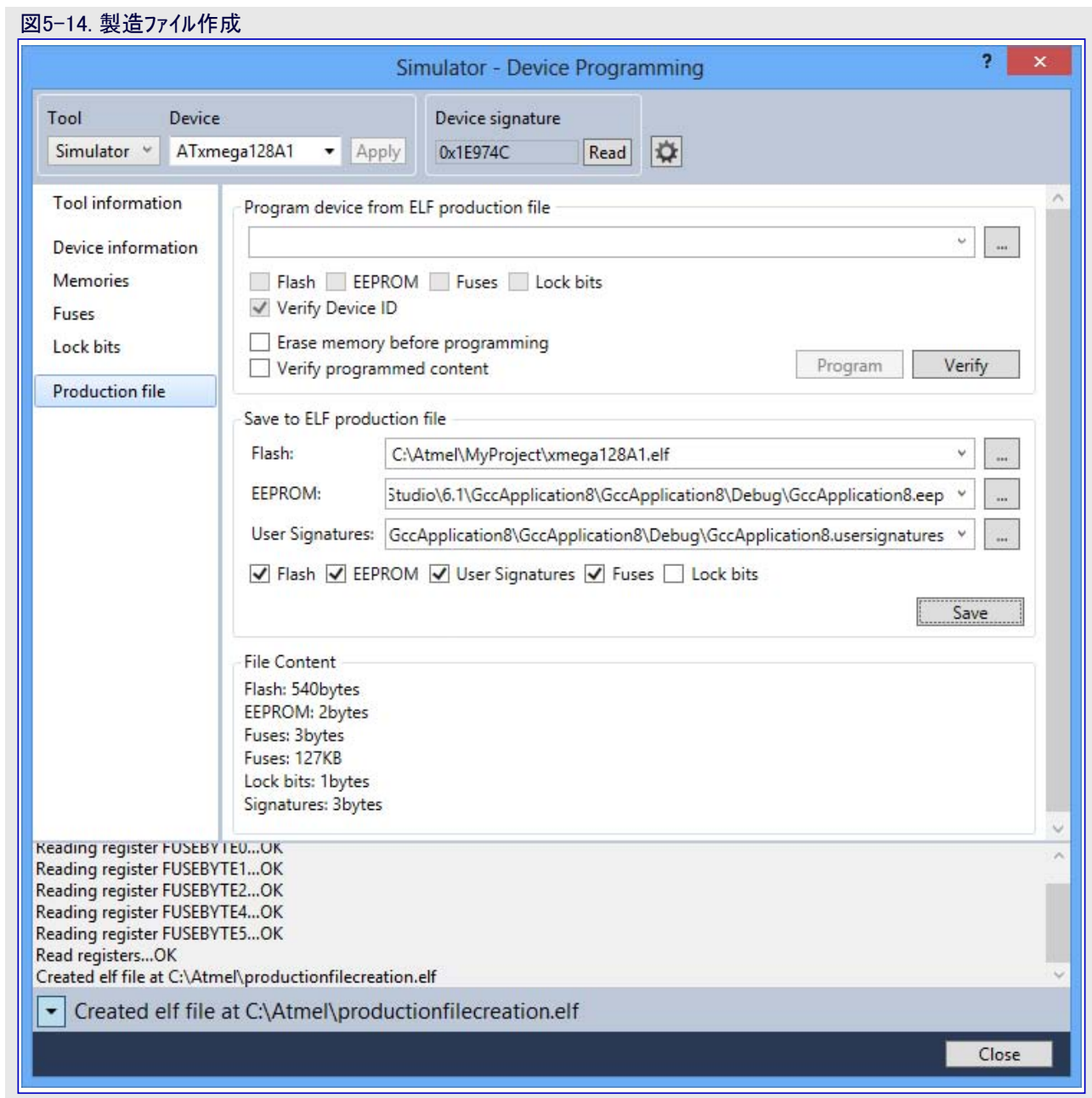
書き込み操作の前に消去操作を実行したい場合、**Erase memory before programming**(書き込み前にメモリ消去)チェック枠を選んでください。

注: このメモリ消去動作はデバイス選択に依存します。tinyAVRとmegaAVRについてはどのメモリが選ばれるかに関わらず、フラッシュメモリとEEPROMの両方と施錠ビットが消去(チップ消去)され、一方XMEGAについては選んだメモリだけが消去されます。

書き込み操作が終わった後で内容を検証したい場合、**Verify programmed content**(書き込み内容を検証)チェック枠を選んでください。接続したデバイスとファイルに格納された内容(識別バイト)を検証したい場合、**Verify Device ID**(デバイスID検証)チェック枠を選んでください。次に、ファイル(内容)をメモリに書き込むために**Program**(書き込み)鈕を押してください。

Verify(検証)鈕を押下することによってELFファイルに対してデバイスの内容を確認することができます。この検証は選ばれたメモリ区分の内容だけを確認します。

図5-14. 製造ファイル作成

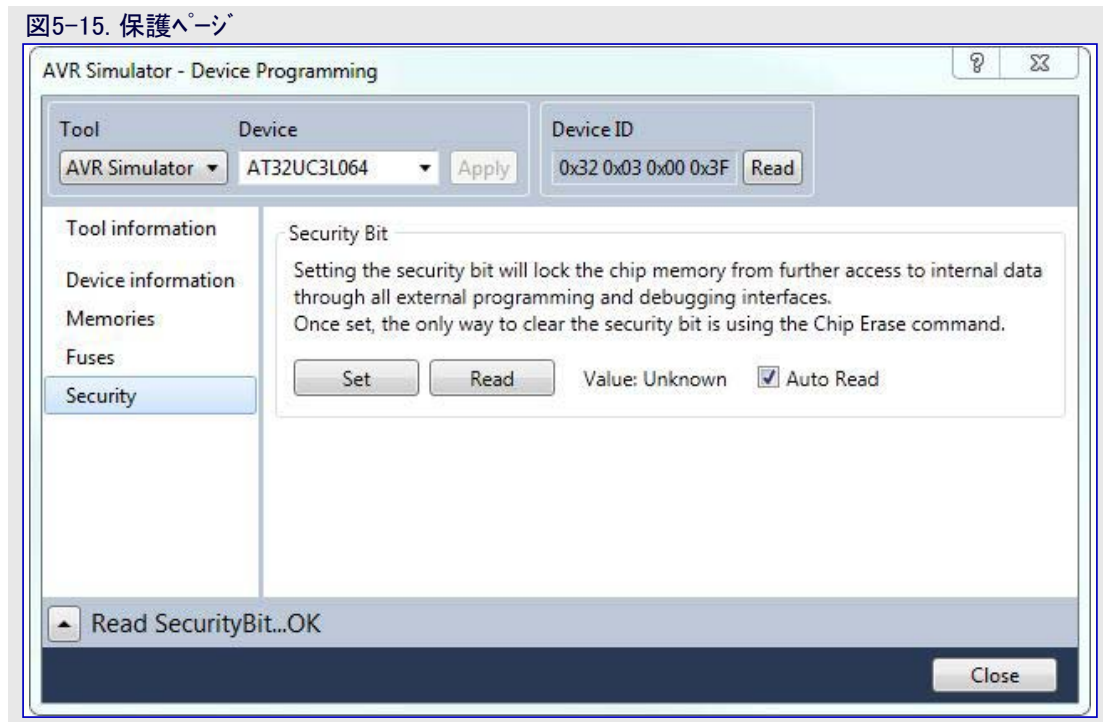


Save to EFL production file (ELF製造ファイルに保存) : ELFファイルの作成に先立って、Production file(製造ファイル)タブでFlash(フラッシュメモリ)、EEPROM、User Signatures(使用者識票)に対する入力ファイルパスを指定してください。その後に対応するタブでヒューズと施錠ビットを構成設定してそれを書き込んでください。デバイスに書かれたヒューズと施錠ビットはELFファイル作成中に入力として使われます。Production fileタブに戻り、ELFファイルを生成するためにSave(保存)釦を押してください。

対応するチェック枠をチェックすることによって製造ELFファイルに存在すべきメモリ区分を指定しなければなりません。

5.13. 保護

保護ビットはコードの安全のために外部JTAGまたは他のデバッグアクセスから施錠されることをチップ全体に許します。一旦設定されると、保護ビットを解除する唯一の方法はErase Chip(チップ消去)命令を通すことです。

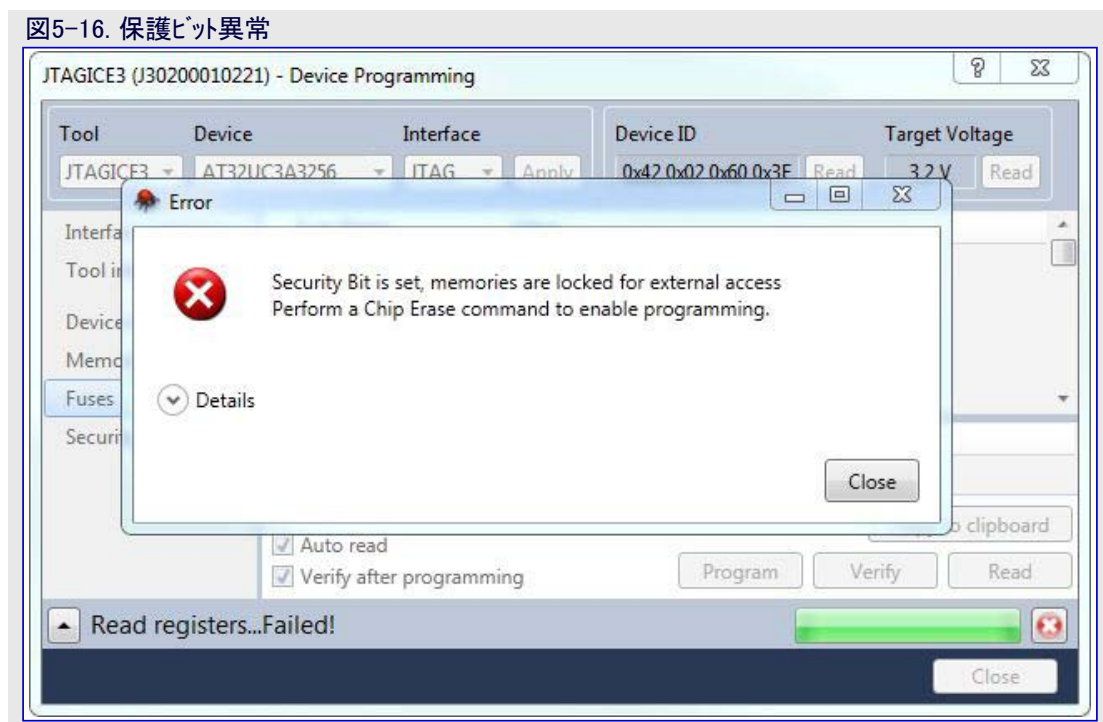


保護ビットの状態を調べるにはプログラミングダイアログのSecurity(保護)頁でRead(読み込み)鈕を押してください。値は直ちにClearedまたはSetを読むべきです。Setは保護ビットが設定されているのを意味します。Clearedはそれが設定されていないのを意味します。Auto Read(自動読み込み)チェック枠がチェックされる場合、Read(読み込み)操作はSecurity頁が開かれる時に自動的に実行されます。

保護ビットを設定するにはプログラミングダイアログのSecurity頁でSet(設定)鈕を押してください。直ぐにデバイスはErase Chip(チップ消去)命令を除いて更なる全てのJTAGまたはaWireのアクセスに対して施錠されます。

施錠されたデバイス

保護ビットが設定されると、デバイスは大抵の外部デバッグアクセスに対して施錠されます。どれかのメモリまたはヒューズの書き込みや読み込みの試みは異常メッセージを出現させるでしょう。



保護ビットを解除するにはErase Chip(チップ消去)命令を発行してください。これはMemories(メモリ)ページから行うことができます。「5.8. メモリ」をご覧ください。

5.14. 自動ファームウェア更新検出

「6.5. ファームウェア更新」項で言及されるように、Device Programming(デバイスプログラミング)ダイアログを開く時にツールのファームウェアが古いことを述べるダイアログに出会うかもしれません。

6. その他のウィンドウ

6.1. デバイス一括管理部

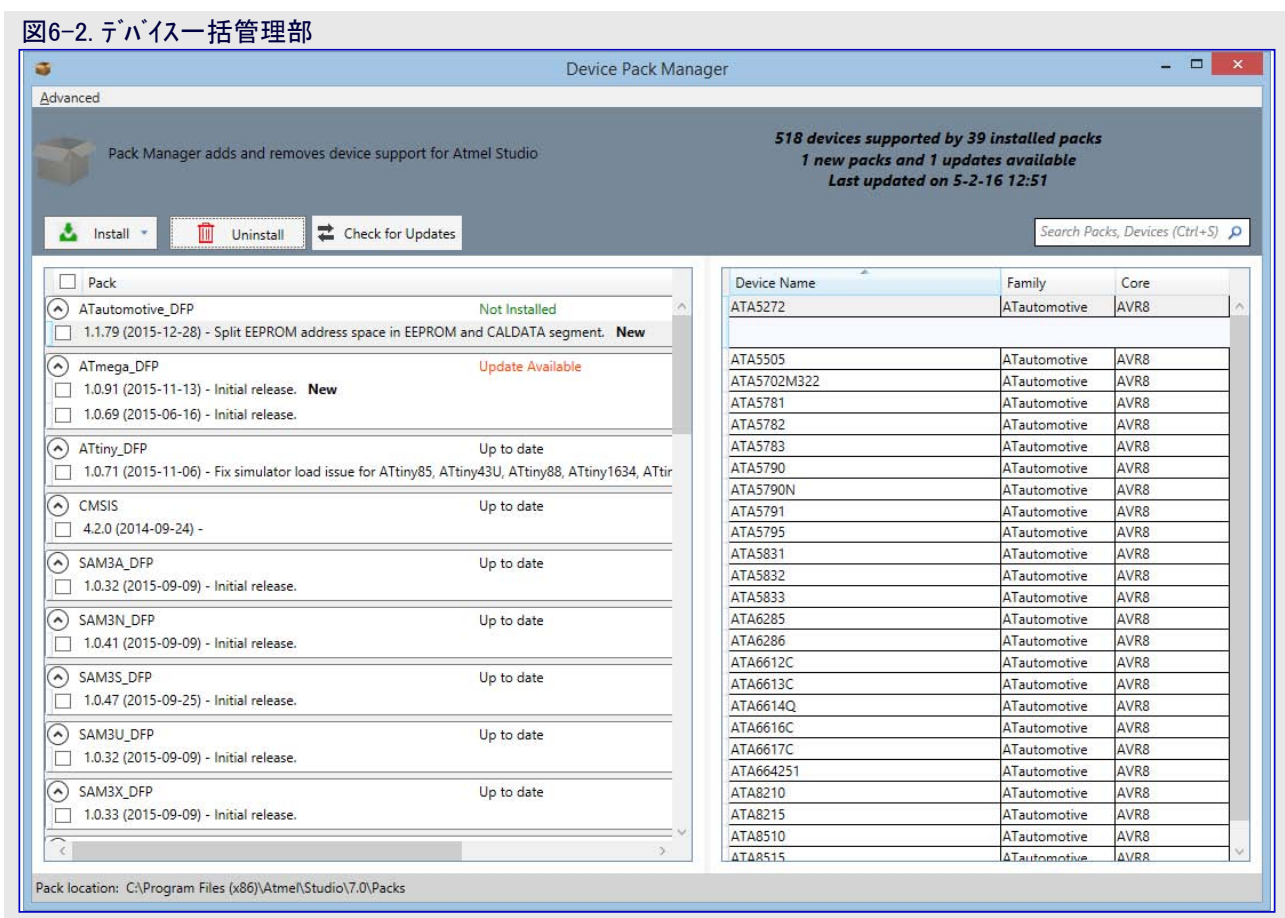
Device Pack Manager(デバイス一括管理部)はMicrochip Studioによって支援されるデバイスを管理します。

デバイス一括管理部はTools(ツール)⇒Device Pack Manager(デバイス一括管理部)から開始されます。

図6-1. デバイス一括管理部メニュー



図6-2. デバイス一括管理部



Device Pack Manager(デバイス一括管理部)は2つの枠から成ります。左側の枠はインストールされた一括(パック)の一覧を示します。右側の枠は左側の枠で選ばれた一括によって提供されるデバイスを示します。

一括は以下の情報のどれかを持ちます。

- Up to date** : 一括は既に最新で最終です。
- Update Available** : 新しい更新が利用可能です。
- Not Installed** : 一括はインストールされていませんが、ダウンロードすることができます。

活動

- Install selected packs** : 版の横のチェック枠を用いて選ばれた全ての一括をダウンロードしてインストール
- Install all updates** : 利用可能な全ての更新をダウンロードしてインストール
- Browse pack file** : 既にダウンロードした一括ファイルをインストール
- Uninstall** : 版の横のチェック枠を用いて選ばれた全ての一括をアンインストール
- Check for Updates** : 新しい若しくは更新された一括を調査
- Search** : 検索枠は特定の一括または一括のどれか内のデバイスを探すのに使うことができます。
- Reset cache** : キャッシュ リセットはインストールされた全一括を再指標化します。これはどれもアンインストールや除去しません。それは **Advanced**(詳細)メニューです。

注: 一括のインストール、更新、削除後、Microchip Studioは変更が見えるようになるのに先立って再始動されなければなりません。

6.2. 使用者インターフェース プロファイル選択

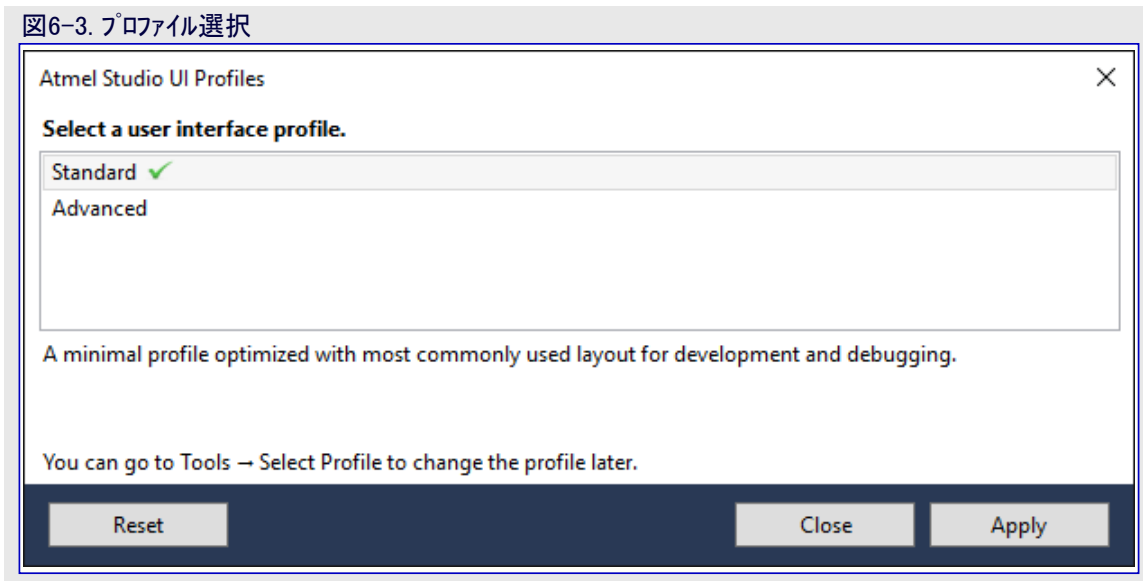
違う使い方に向けた異なる使用者インターフェース プロファイルがMicrochip Studioで利用可能です。

使用者インターフェース プロファイルはMicrochip Studioのメニュー、ウィンドウ配置、ツールバー、状況メニュー、それと他の要素の見た目を制御します。以下の形態が利用可能です。

Standard (標準) : 既定プロファイル。使うウィンドウとメニューの殆どを含みます。

Advanced (詳細) : このプロファイルは高度なデバッグとコード整理ツールを含みます。

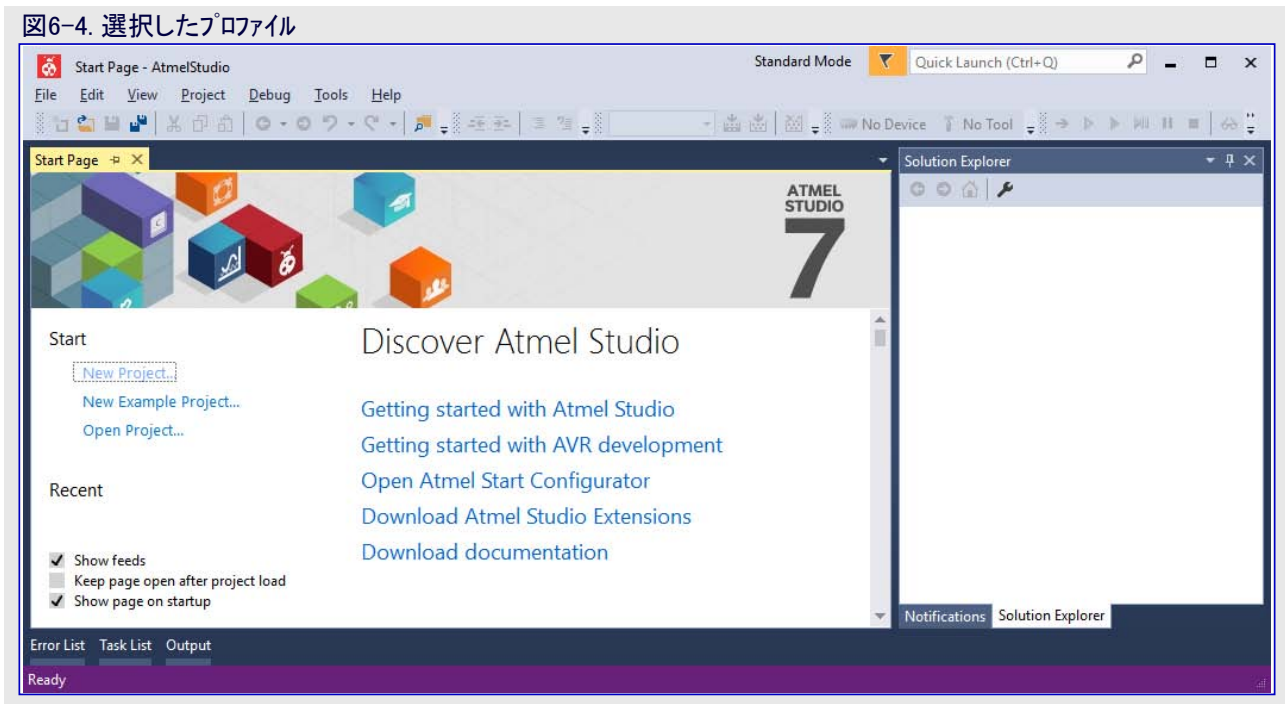
図6-3. プロファイル選択



プロファイル選択ウィンドウはMicrochip Studioが初めて開始される時に示されます。一覧でプロファイルを選択するとそのプロファイルの説明が示されます。**Apply**(適用) 鈕のクリックがMicrochip Studioにそのプロファイルを適用させます。

プロファイルは**Tools**(ツール)⇒**Select Profile**(プロファイル選択)へ誘導することによって、またはMicrochip Studioの右上角に表示されるプロファイル名をクリックすることによって何時でも変更することができます。

図6-4. 選択したプロフィール



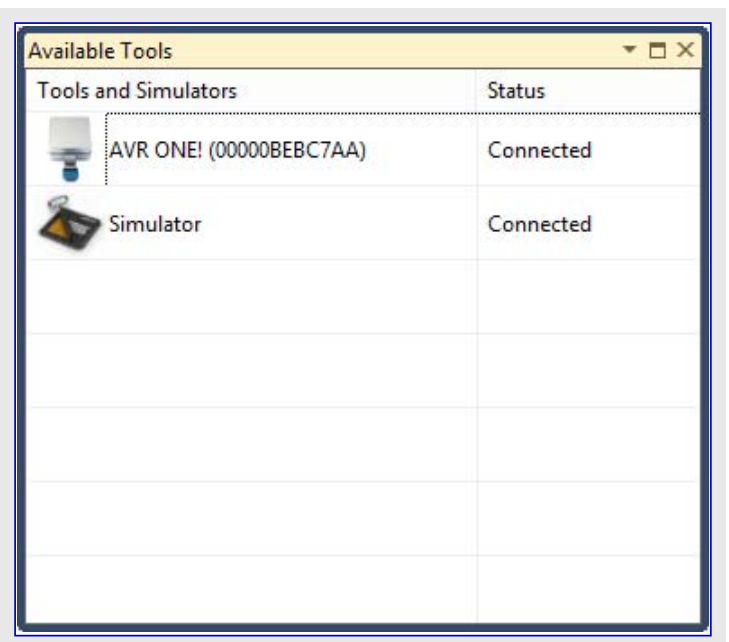
プロフィール切り替え時、活性なプロフィールに対して行われたどの変更も保存されます。以前のプロフィールへ戻すとそのプロフィールだけでなく変更も復元されます。

Reset(リセット)任意選択の使用はプロフィールに対して保存されたどの変更も破棄して既定プロフィールに回復します。

6.3. 利用可能なツール表示部

6.3.1. 序説

Available Tools(利用可能なツール)表示部(View(表示部)⇒Available Microchip Tools(利用可能なMicrochipツール))は書き込み器、デバッガ、スタータキットのような接続された全てのツールの一覧を含みます。シミュレータは常に存在します。他のツールはそれらがPCに接続された時に現れます。



6.3.2. ツールの動作

Available Tools(利用可能なツール)表示部でツールを右クリックすることによって以下の動作を選ぶことができます。

Device Programming : 予め選ばれたツールでデバイスプログラミング ウィンドウを開きます。

Self test : いくつかのツールは自己検査を実行する能力があります。表示された指示に従ってください。

Add target : 自動検出不能な利用可能なツールの一覧をツールに追加します。より多くの情報については「[6.3.3. 検出不能なツールの追加](#)」をご覧ください。

Upgrade : 選んだツールでファームウェア更新ツールを開始します。

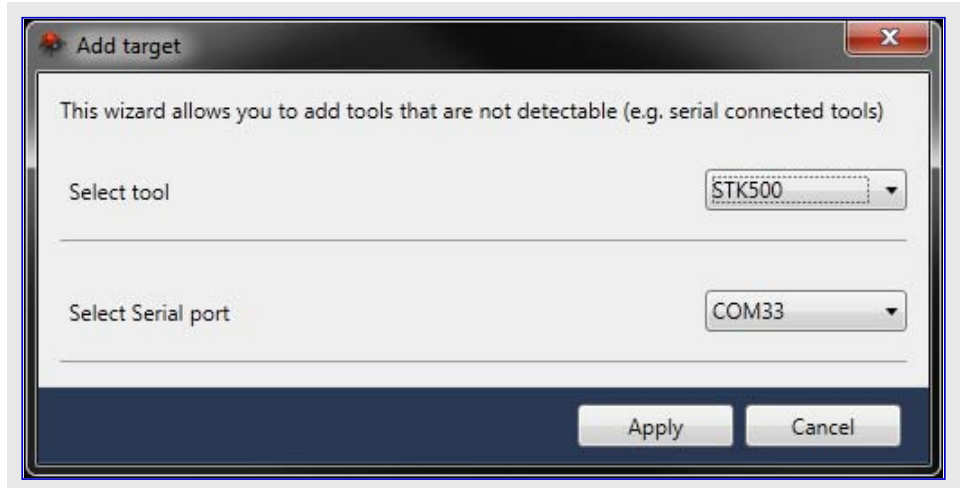
Show Info Window : Tool Info(ツール情報)ウィンドウを見せます。全てのツールがこの機能を支援する訳ではありません。より多くの情報については「[6.4. ツール情報ウィンドウ](#)」をご覧ください。

6.3.3. 検出不能なツールの追加

STK500はUSB接続を持たず、Microchip Studioによって自動的に検出できません。故にDevice Programming(デバイスプログラミング)ウィンドウによってそれが使われるのに先立って利用可能なツールの一覧にそれを追加しなければなりません。

STK500を追加するにはAvailable Tools(利用可能なツール)表示部の内側を右クリックしてAdd target(目的対象を追加)を選び、ツールとしてのSTK500とSTK500が接続されたCOMポートを選んでください。

Apply(適用)鈕を押してください。利用可能なツール一覧にSTK500が表示されます。



注: 追加されたSTK500は例え指定したCOMポートにSTK500が接続されなくても、Available Tools表示部で見えます。STK500を一覧から削除したい場合、状況メニューからRemove(削除)を選んでください。

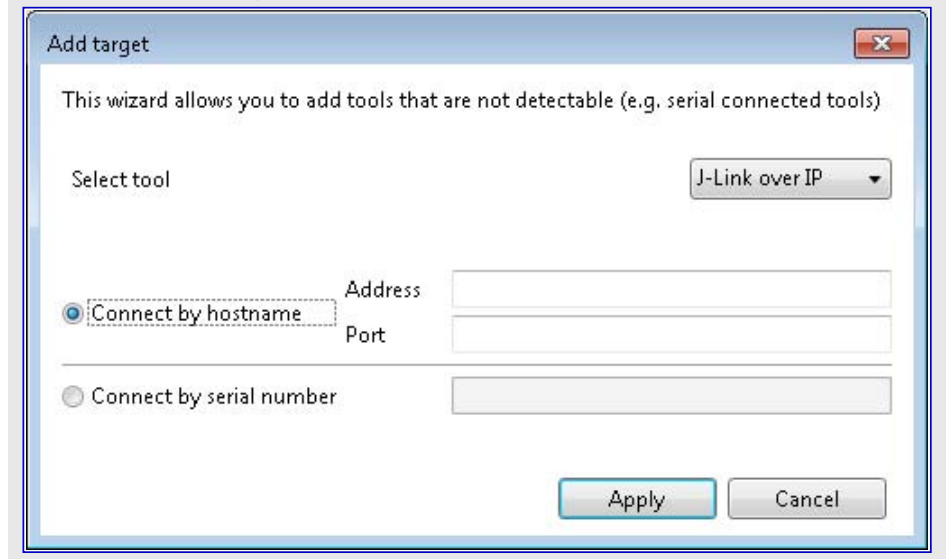
6.3.3.1. IP上のJ-Link追加

Add target(目的対象を追加)ダイアログで、J-Link PRO(脚注1)のような組み込みイーサネットでのデバッグ探針使用とJ-Link遠隔サーバーソフトウェア(脚注2)を用いることによる他のSegger探針の両方の遠隔Segger J-Linkデバッグ探針を追加することが可能です。

J-Link遠隔サーバーに接続されたデバッグ探針を追加するには、Connect by hostname(ホスト名によって接続)を選んでJ-Link遠隔サーバーが動いているコンピュータのIPアドレスかホスト名を入力してください。J-Link遠隔サーバーが非標準ポート(脚注3)で動いている場合、Port(ポート)も入力することが必要です。J-Link遠隔サーバーが既定ポートで動いている場合、Port(ポート)を空のままにすることができます。

組み込みイーサネットを持つデバッグ探針を追加するにはAdd targetダイアログでConnect by serial number(通番によって接続)を選び、デバッグ探針の通番を入力してください。

図6-5. IP上J-Linkの追加



注1: www.segger.com/jlink-pro.htmlをご覧ください。

注2: www.segger.com/jlink-remoteserver.htmlをご覧ください。

注3: J-Link遠隔サーバーの標準ポートはポート19020です。

6.4. ツール情報ウィンドウ

Tools Info(ツール情報)ウィンドウは接続したツールについての情報を示します。現時点では、Xplained Pro系統だけが支援されます。



ツールが接続されると、このウィンドウが開きます。これはツールの短い説明、ツールの画像、更にまたインターネットでの使用者の手引き、関連データシートなどへのリンクの部分を持ちます。

ファームウェア版番号、通番などのようなツールについての技術的な詳細を持つ表もあります。

6.4.1. Xplained Proキット

Xplained Pro系統の基板は広範囲な拡張基板を支援します。Xplained Pro基板が接続されると、Tool Info(ツール情報)ウィンドウはウィンドウの左側で主基板と接続された全ての拡張を含む一覧を示します。各種基板についての詳細を見るには主基板や拡張上をクリックしてください。

6.4.2. ツール情報ウィンドウ禁止

Show page on connect(接続でページを見せる)チェック枠を選択解除することによって、このウィンドウはMicrochip Studioが開いてキットに接続する時に自動的に開きません。

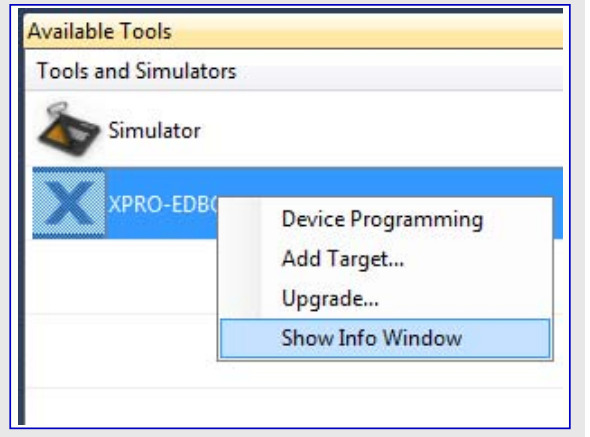
この機能はツール単位方式で動き、そしてこれはあなたが持つ全てのツールに対して、それらが接続された時にTool Info(ツール情報)ウィンドウを見せるべき場合に、それらを選ぶことができることを意味します。

6.4.3. ウィンドウの手動表示

Tool Info(ツール情報)ウィンドウを閉じた後で再び見たい場合、Available Tools view(利用可能なツール表示部)でツール上を右クリックしてShow Info Window(ツールウィンドウ表示)を選ぶことができます。

「6.3. 利用可能なツール表示部」もご覧ください。

図6-7. ツール情報ウィンドウ表示



6.5. ファームウェア更新

6.5.1. 序説

Microchip StudioはMicrochipツールの最新ファームウェアを含みます。新しいファームウェアは新デバイス用の支援やバグ修正を提供するかもしれません。

6.5.2. 自動更新

Microchip Studioは必要とされる時にツールのファームウェアを自動的に更新します。潜在的なファームウェア更新は一度ツールの使用を開始すると、起動されます。例えば、初めてデバッグ作業を開始する、または初めてDevice Programming(デバイスプログラミング)ダイアログでツールを選ぶ時です。

使用者が更新を選ばない場合、そのツールはMicrochip Studioによって使うことができません。

Avairable Tools(利用可能なツール)表示部(View(表示部)⇒Available Microchip Tools(利用可能なMicrochipツール))を用いることによってファームウェア更新を調べることができます。ツール上で右クリックしてUpgrade(更新)を選んでください。

手動更新、以前版への更新、独自ファームウェア イメージでの更新を行う方法の記述については「6.5.3. 手動更新」をご覧ください。

6.5.3. 手動更新

Microchip Studioは殆どのMicrochipツールの手動更新を行うのに使うことができるatfw.exeと呼ばれるコマンド行ユーティリティを含みます。atfw.exeはatbackend副フォルダにインストールされます。

atfw.exeは以下に使うことができます。

- スクリプトからの更新実行
- 独自ファームウェア ファイルを使う更新
- ファームウェア版番号の読み出し

このユーティリティを用いて更新する方法の詳細についてはatfw.exe -hを実行してください。

注: ツールがファームウェア更新動作で固定化されて通常のリセットが通常動作を回復しない場合、強制的なファームウェア更新がツールを動く状態にリセットするでしょう。

既に更新動作のツールでファームウェア更新を行うには、通常ファームウェア更新と同じようにatfwを実施してください。ツールはツールを更新動作に切り替えることが不可能のようないくつかの警告が表示されるかもしれませんが、更新を続行すべきです。

ツール一覧表示が行われた場合、そのツールはそれの動作に関連する名前を持ちます。けれども、atfwはそれが標準動作で使用者に提示されるようなツール名で実施されるべきです。

6.6. 検索と置換のウィンドウ

文書のコード内でテキスト文字列、式、実体名を検索するのに**Find and Replace**(検索と置換)ウィンドウを使うことができます。このウィンドウにアクセスするには**Edit**(編集)メニューから**Find and Replace**(検索と置換)をクリックし、その後に一覧にされた任意選択の1つを選んでください。

Find and Replaceウィンドウは1つが検索操作で、もう1つが置換操作の2つの引き落としを持つツールバーを含みます。操作を選ぶと、その操作に対応する任意選択が表示されます。テキスト、コード、シンボルに対して1つまたは複数のファイルや解決策全体で検索と置換をすることができます。

Quick Find(迅速検索)は文字列や式に対して1つまたはは複数の開いた文書のコードの検索を許します。一致から一致へ移動する選択はそれの周囲前後関係で各々の一致を再調査することを許します。

注: 見つけた一致は**Find Results**(検索結果)ウィンドウで一覧にされません。

Find and Replaceウィンドウで**Quick Find**を表示するのに以下の方法のどれかを使うことができます。

Quick Findを表示

1. **Edit**(編集)メニューで**Find and Replace**(検索と置換)を展開してください。
2. **Quick Find**(迅速検索)を選んでください。
--または--

Find and Replaceウィンドウが既に開いている場合、ツールバーで左側の引き落としで▼の**View**(表示)鈕をクリックし、その後**Quick Find**(迅速検索)を選んでください。

Quick Find(迅速検索)は挿入位置から進行方向と逆方向のどちらかで文書を通して検索することができます。検索は文書の最後または始めを過ぎて未検索の部分を自動的に続けます。文書全体が検索されてしまうと、メッセージが現れます。

Find what (何を検索)

これらの部分は一致する文字列や式の指定を許します。

この引き落とし一覧からそれを選ぶことによって最後の20検索文字列の1つを再利用するか、または見つける新しいテキスト文字列や式を入力してください。

図6-8. 検索と置換

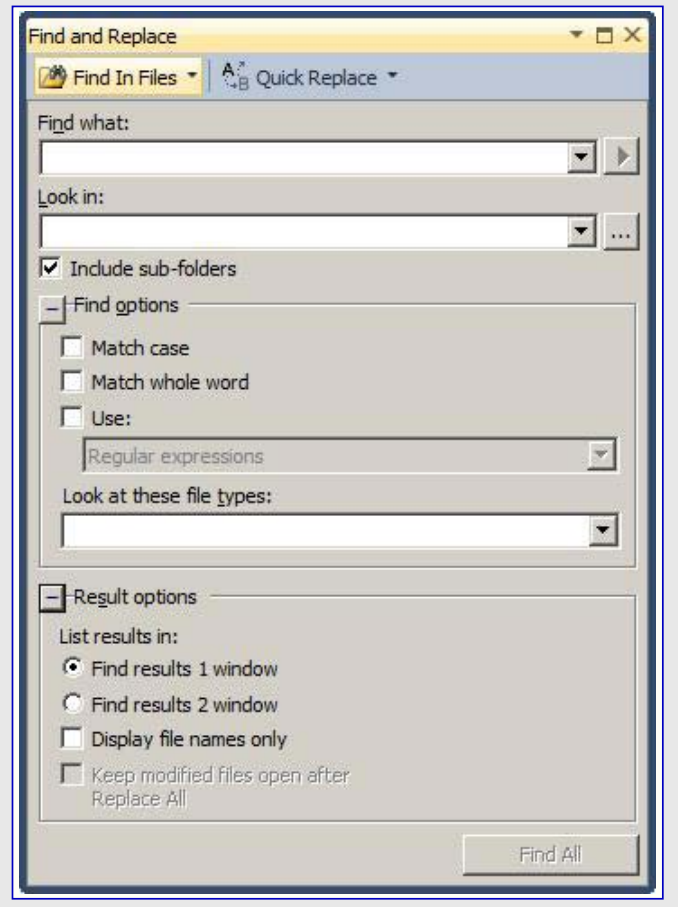


表6-1. 迅速検索

任意選択	説明
[ワイルドカード付き文字列]	検索文字列内でアスタリスク(*)と疑問符(?)のようなワイルドカードを使いたい場合、 Find options (検索任意選択)下の Use (使用)チェックボックスをチェックし、その後 Wildcards (ワイルドカード)を選んでください。
[正規表現]	正規表現を期待することを検索エンジンに教えるために Find options (検索任意選択)下の Use (使用)チェックボックスをチェックし、その後 Regular expressions (正規表現)を選んでください。

式構築部

Find what(何を検索)領域の次の三角鈕は**Find options**(検索任意選択)で**Use**(使用)チェック枠が選ばれて、引き落とし一覧に**Regular expressions**(正規表現)が現れる時に利用可能になります。選んだ**Use**任意選択に応じてワイルドカードまたは正規表現の一覧を表示するためにこの鈕をクリックしてください。この一覧からの何れかの項目選択がそれを**Find what**文字列に追加します。

Find Next (次を検索)

Look in(調査範囲)で選んだ検索範囲内で**Find what**文字列の次の実体を見つけるにはこの鈕をクリックしてください。

Bookmark All (全てを葉表示)

Find what文字列の実体が存在する各行を示すためにコードエディタの左端で青の葉を表示するにはこの鈕をクリックしてください。

調査範囲

Look in(調査範囲)引き落とし一覧から選んだ任意選択は現在の活性ファイルでだけ**Quick Find**(迅速検索)をするかどうかを決めます。

Look in (調査範囲)

この一覧から予め定義された検索範囲を選んでください。

表6-2. 調査範囲

任意選択	説明
Selection	選択：この任意選択はコードエディタで文字列が選択される時に利用可能です。現在活性的な文書で選択された文字列だけを検索します。
<Current Block>	<現在の塊>：この任意選択の名前はコードエディタでの挿入点の位置を示します。現在のプロシージャ、単位部、節、またはコードの塊内で検索。
Current Document	現在の文書：この任意選択はエディタで文書が開かれている時に利用可能です。Find what文字列に対して活性的な文書だけを検索。
Current Window	現在のウィンドウ：この任意選択はブラウザウィンドウでの表示部のような検索可能なツールウィンドウがフォーカスを持つ時に利用可能です。Find what文字列に対してこのウィンドウで表示された全ての内容を検索。
All open Documents	開いた全ての文書：それらが1つの文書であるかのように、編集用に現在開いた全てのファイルを検索。現在のファイルで検索の開始点に達すると、検索は自動的に次のファイルに移動し、Find what文字列に対して最後に開いたファイルが検索されてしまうまで続けます。
Current Project	現在のプロジェクト：それらが1つの文書であるかのように、現在のプロジェクト内の全てのファイルを検索。1つのファイルで検索の開始点に達すると、プロジェクト内の最後のファイルが検索されてしまうまで検索は次に続きます。

Find options (検索任意選択)

Find options(検索任意選択)領域を展開または格納することができます。以下の任意選択を選択または解除することができます。

Match case (大文字/小文字区別で一致)

大文字/小文字と内容の両方が一致するFind what(何を検索)文字列の実体だけを表示。例えば、Match case選択で”MyObject”に対する検索は”MyObject”を返しますが、”myobject”や”MYOBJECT”は返しません。

Match whole hidden text (隠された文字列全体の一致)

選択時、検索は設計時制御のメタデータのように、隠されたまたは折り畳まれた文字列も含みます。中抜きされた文章の隠された領域、または折りたたまれたクラスやメソッド。

Use (使用)

テキスト枠を持つFind what(何を検索)またはReplace(置換)で入力された特殊文字をどう解釈するかを示します。この任意選択は以下を含みます。

表6-3. 特殊文字での検索

任意選択	説明
Wildcards	ワイルドカード：アスタリスク(*)と疑問符(?)のような特殊文字は1つまたはより多くの文字を表します。一覧についてはワイルドカード(Visual Studio)をご覧ください。
Regular Expressions	正規表現：特殊表記法は一致するテキストの様式を停止します。一覧については正規表現(Visual Studio)をご覧ください。

ツールバー

2つの引き落としを持つツールバーがFind and Replace(検索と置換)ウィンドウの上部に現れます。これらの引き落とし矢印は実行を意図する検索または置換の形式を選び、一致に対するウィンドウで表示される任意選択の変更を許します。

表6-4. 検索と置換のツールバー

引き落とし	表示メニュー
Find (検索) (左側引き落とし)	Quick Find (迅速検索) Find in Files (ファイル内検索) Find Symbol (シンボル検索)
Replace (置換) (右側引き落とし)	Quick Replace (迅速置換) Replace in Files (ファイル内置換)

図6-9. Find Results (検索結果)

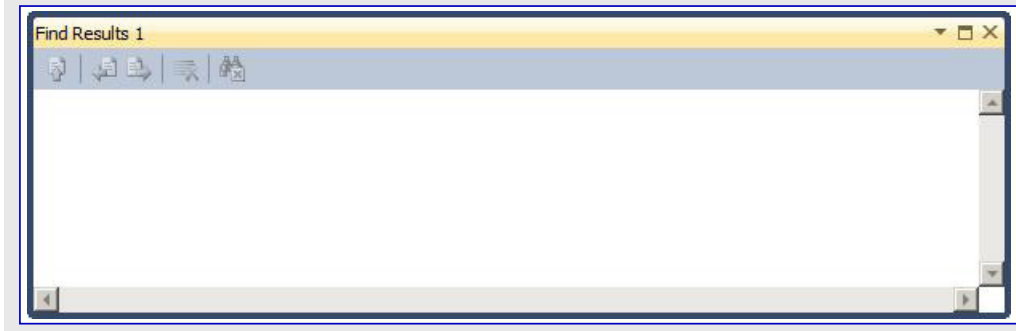
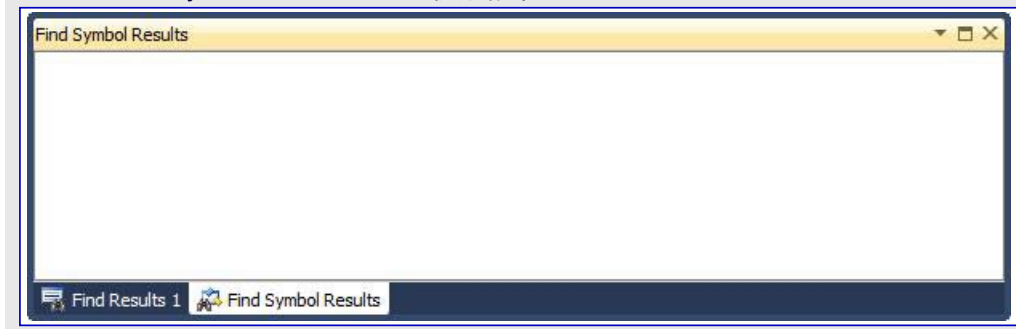


図6-10. Find Symbol Results (シンボル検索結果)



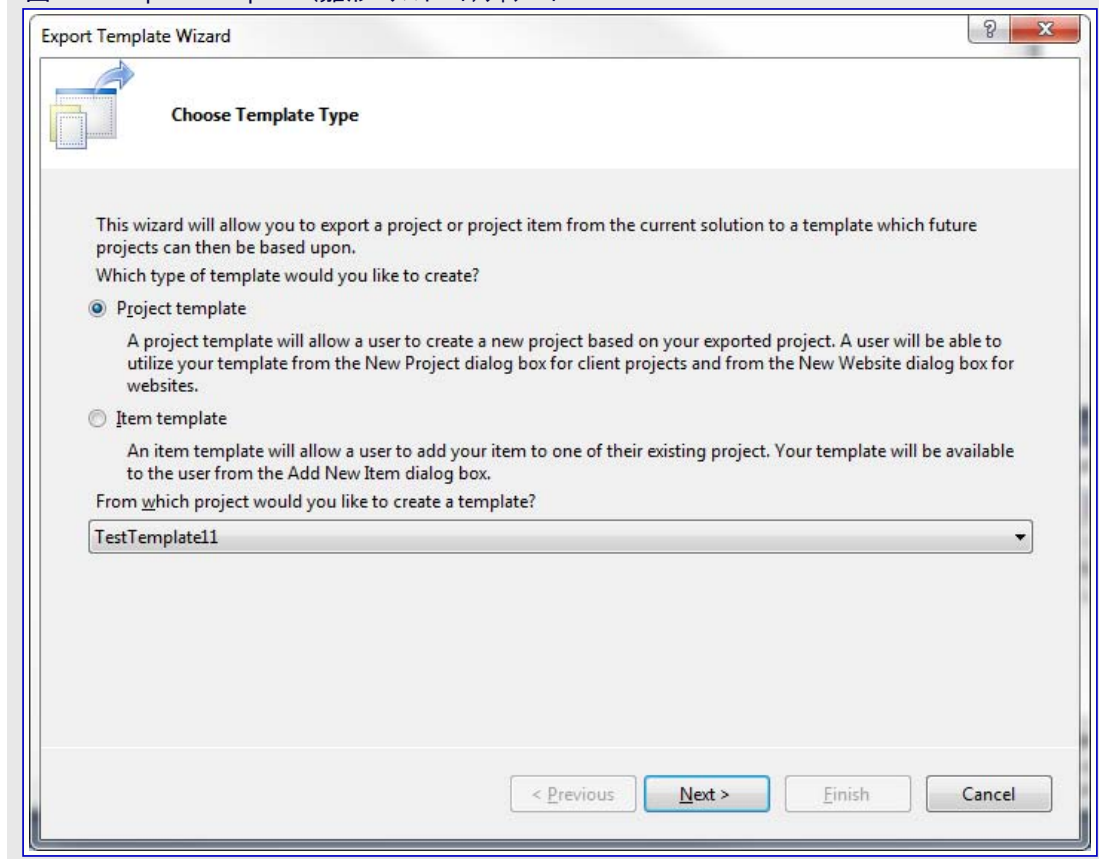
6.7. 雛形エクスポート ウィザード

Microchip Studioのプロジェクトと項目雛形は開発手順を加速する再利用可能で独自化可能なプロジェクトと項目を提供し、故に使用者は0から新しいプロジェクトと項目を作成する必要がありません。

注: この機能はMicrosoft Visual Studio®から受け継がれ、Microsoftからの資料は本項で言及されるものを超えます。

File(ファイル)⇒Export Template...(雛形エクスポート)をクリックすることによってExport Template(雛形エクスポート)ウィザードを開いてください。この開いたExport Templateウィザードは下図で示されます。

図6-11. Export Template (雛形エクスポート)ウィザード



6.7.1. プロジェクト雛形

プロジェクト雛形はプロジェクト全体を含む雛形です。この雛形は既定プロジェクトの容易な準備のために他の使用者へ再分配することができます。雛形であるべきコードは作成に於いて代わりに用いられるパラメータを含むことができます。この情報については「[6.7.3.2. 既定雛形パラメータ](#)」をご覧ください。

雛形ウィザードは殆ど自己説明的で、完了で作成された雛形はFile(ファイル)⇒File(ファイル)⇒New Project...(新しいプロジェクト)でのダイアログで利用可能です。

6.7.2. 項目雛形

項目雛形は単一ファイルまたはファイルの集合を含む雛形です。雛形にされるべきコードは作成に於いて代わりに用いられるパラメータを含むことができます。この情報については「[6.7.3.2. 既定雛形パラメータ](#)」をご覧ください。

雛形ウィザードは殆ど自己説明的で、完了で作成された雛形はファイルがプロジェクトに追加される時にファイル形式として利用可能です。

6.7.3. 雛形パラメータ

全ての雛形は雛形が実体化される時に、クラス名と名前空間のような鍵となるパラメータの置換を許すためにパラメータ代替を支援します。これらのパラメータは使用者がNew Project(新しいプロジェクト)またはAdd New Item(新しい項目を追加)のダイアログ枠でOKをクリックする時に背面で走行する雛形ウィザードによって置き換えられます。

6.7.3.1. 雛形パラメータの宣言と許可

雛形パラメータは\$parameter\$形式で宣言されます。

6.7.3.2. 既定雛形パラメータ

下表はどれかの雛形によって使われ得る予約された雛形パラメータを一覧にします。

注: 雛形パラメータは大文字/小文字を区別します。

表6-5. 雛形パラメータ

パラメータ	説明
\$itemname\$	使用者によってAdd New Itemダイアログ枠で提供された名前
\$machinename\$	現在のコンピュータ名
\$projectname\$	使用者によってNew Projectダイアログ枠で提供された名前
\$registeredororganization\$	HKLM¥Software¥Microsoft¥Windows NT¥CurrentVersion¥RegisteredOrganizationからのレジストリ キー値
\$safeitemname\$	除去された全ての危険文字と空白で、使用者によってAdd New Itemダイアログ枠で提供された名前
\$safeprojectname\$	除去された全ての危険文字と空白で、使用者によってNew Projectダイアログ枠で提供された名前
\$time\$	DD/MM/YYYY 00:00:00形式での現在時間
\$userdomain\$	現在の使用者ドメイン
\$username\$	現在の使用者名
\$year\$	YYYY形式での現在の年号
\$guid[1-10]\$	GUIDはプロジェクト ファイルでプロジェクトGUIDを置き換えるのに使われます。10までの固有GUIDを指定することができます(例えば、guid1)。

6.7.3.3. 独自雛形パラメータ

雛形に新しいパラメータを追加するためにあなたの.vstemplateファイルでCustomParameter要素を使うことができます。

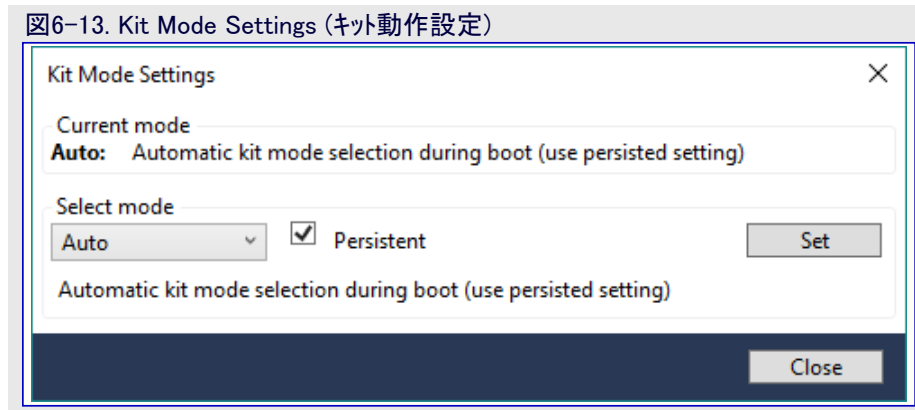
- 雛形に対する.vstemplateファイルでTemplateContent要素を配置してください。
- TemplateContent要素の子としてCustomParameter要素と1つまたはより多くのCustomParameter子要素を追加してください。

```
<TemplateContent>
~
<CustomParameters>
  <CustomParameter Name="$MyParameter1$" Value="MyValue2"/>
  <CustomParameter Name="$MyParameter2$" Value="MyValue2"/>
</CustomParameters>
</TemplateContent>
```

- 「[6.7.3.2. 既定雛形パラメータ](#)」で示されるように雛形でコード ファイルの1つまたはより多くでこのパラメータを使ってください。

6.8. キット動作設定

いくつかのキットは異なる動作形態で作動します。動作形態を変更するにはこのウィンドウを使ってください。



行い得る選択のいくつかの例が以下の表で一覧にされます。

Select mode	Persistent	結果としての動作
Mass Storage	チェック	自動、大容量記憶装置キットとして列挙(接続認識)
DGI	チェック	自動、DGIキットとして列挙(接続認識)
Mass Storage	非チェック	大容量記憶、前の動作に戻る前に一旦大容量記憶装置キットとして列挙(接続認識)
DGI	非チェック	DGI、前の動作に戻る前に一旦DGIキットとして列挙(接続認識)

注: Persistent(持続)動作が使われると、Persistent選択がキットの既定を変更するため、キットは自動動作でレポートします。

7. GNUツールチェーン

GNUツールチェーンはSAMとAVRのマイクロコントローラ用の応用を作成するのに使われる独立型コマンド行プログラムの組です。

7.1. GNUコンパイラ集合 (GCC)

GNUコンパイラ集合は構築段階でMicrochip Studioによって使われます。GNUコンパイラ集合の基本設計特定版はCコードのコンパイル、アセンブル、CとC++のリンクを支援します。

AVR GNUコンパイラ集合はGNU一般公開許諾(<http://www.gnu.org/licenses/gpl.html>)の条件の下で配布されます。この許諾書の複製はMicrochip Studioのインストールフォルダでも見つかります。

7.2. Arm® GNUツールチェーン任意選択: GUI

Arm GCCツールチェーンでの手助けを得るのに以下を行うことができます。

- GCCについての全般的な情報に関しては公式の[GNU GCCウェブ サイト](#)を訪ねてください。
- あるいは、`arm-none-eabi-gcc --help`を書いてその命令出力でパラメータのいくつかの説明を見ることができます。

本項はMicrochip StudioでArm GNUツールチェーン用に利用可能なGUI任意選択を説明します。

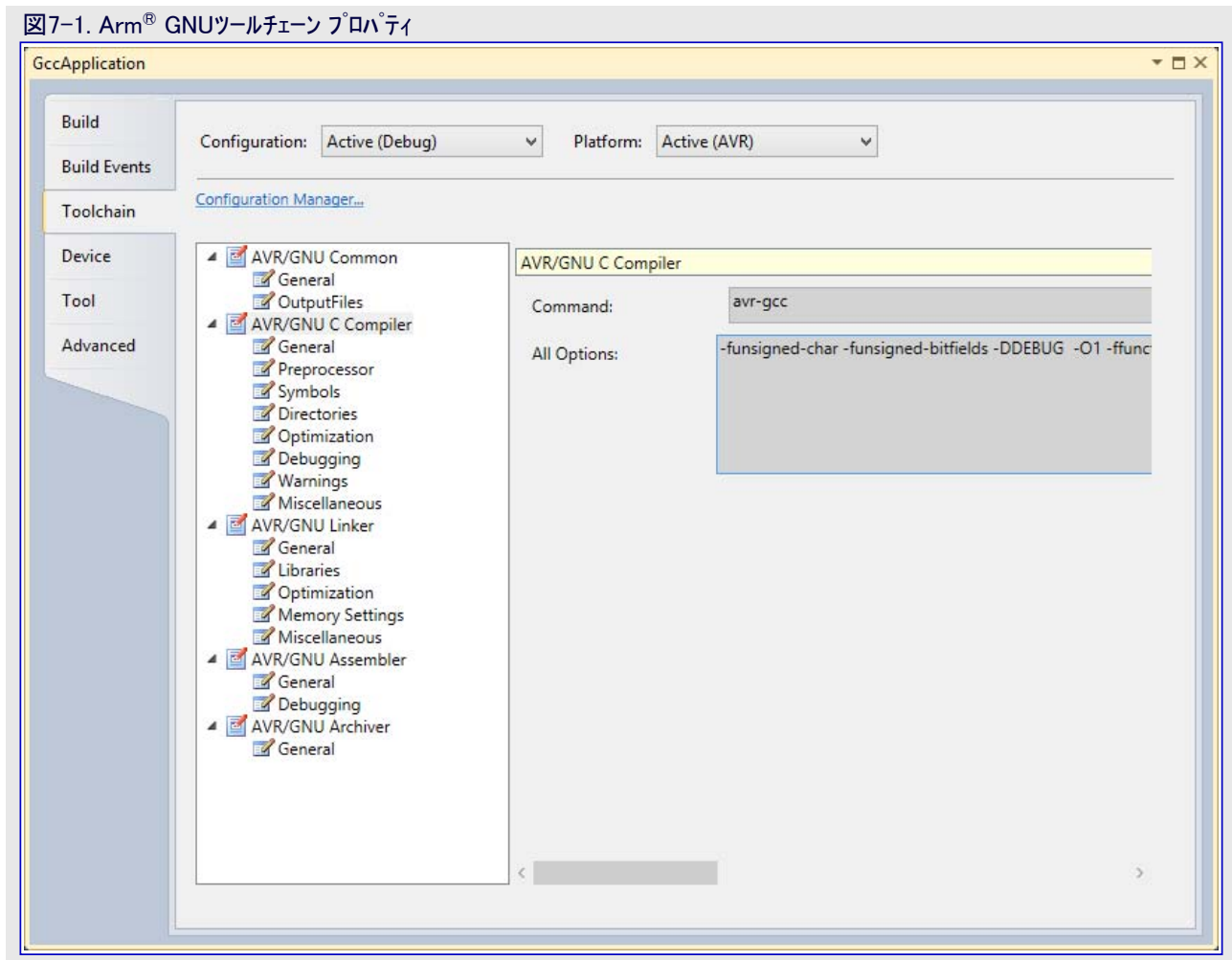


表7-1. Arm® GNU共通任意選択

任意選択	説明
Thumb(-mthumb)/Arm(-marm)	ArmとThumbのプロセッサ動作を切り替え

表7-2. Arm® GNU コンパイラ任意選択

任意選択	説明
プロセッサ任意選択	
-nostdinc	システム インクルード デイレクトリを検索しない。
-E	前処理(プリプロセッサ)のみ。コンパイル、アセンブル、リンクをしない。
シンボル任意選択	
ソース内のいくつかのシンボルを定義(-D)または定義解除(-U)することができます。新しいシンボル宣言は下のインターフェース鉤を用いて追加、変更、再整理をすることができます。	
<ul style="list-style-type: none"> •  新しいシンボル追加。これと以下の全アイコンはMicrochip Studioインターフェースの他の部分で同じ意味で再利用されます。 •  シンボル削除 •  シンボル編集 •  解析順に於いてシンボルを上へ移動 •  解析順に於いてシンボルを下へ移動 	
インクルード デイレクトリ	
既定インクルードパス	この任意選択の許可は選んだSAMデバイスに対して具体的なインクルードパスを追加します。
インクルードされるヘッダと定義指令の全て	を含み、シンボルと同じインターフェースを用いて編集することができます。
最適化任意選択	
最適化レベル(引き落としメニュー) -O0, -O1, -O2, -O3, -Os	最適化なし、速度最適化(レベル1~3)、大きさ最適化
他の最適化フラグ(手入力方式)	ここで基盤と必要条件に対して指定する最適化フラグを書くべきです。
-ffunction-sections	それ自身の区分(section)に各関数を配置
-funsafe-math-optimizations	安全でない一致の最適化を許可
-ffast-math	高速数値演算を許可
-fpic	位置無関係なコードを生成
デバッグ任意選択	
デバッグレベル(引き落としメニュー) none, -g1, -g2, -g3	ソースコードで残されるまたは挿入されるコードとヘッダを追跡してデバッグするレベルを指定
他のデバッグ任意選択(書式領域)	基本構造特有デバッグ任意選択
-pg	gprof情報を生成
-p	prof情報を生成
警告メッセージ出力任意選択	
-Wall	全ての警告
-Werror	警告を異常に格上げ
-fsyntax-only	構文だけ検査
-pedantic	GNUに対する適合性検査、非標準プログラミング実行で警告揭示
-pedantic-errors	同上、加えて警告を異常に格上げ
-w	全ての警告を禁止
その他の任意選択	
他のフラグ(書式領域)	他のプロジェクト特有フラグを入力
-v	詳細 (コンパイラによって呼び出されたプログラムを表示)
-ansi	ANSIプログラムを支援
-save-temps	一時ファイルを削除しない

表7-3. Arm® GCCリンカ任意選択

任意選択	説明
-Wl -nostartfiles	標準ファイルを使わない。
-Wl -nodefault	既定ライブラリを使わない。
-Wl -nostdlib	始動または既定ライブラリなし
-Wl -s	全てのシンボル情報を省く。
-Wl -static	静的にリンク
-Map	Mapファイルを生成
ライブラリ任意選択	
ライブラリ -Wl, -l (書式領域)	 の鉤を用いて、ここでライブラリ名の追加、優先付け、編集ができます。
ライブラリ検索パス -Wl, -L (書式領域)	リンカが動的にリンクされるライブラリを探す所のパスの追加、優先付け、編集ができます。上と同じインターフェースです。
最適化任意選択	
-Wl, -gc-sections	未使用領域ゴミ回収
-funsafe-math-optimizations	安全でない一致の最適化を許可
-ffast-math	高速数値演算を許可
-fpic	位置無関係なコードを生成
その他の任意選択	
他のリンカフラグ (書式領域)	他のプロジェクト特有フラグを入力

リンカ スクリプト

- “Linker(リンカ)⇒miscellaneous(その他)⇒linker flags(リンカ フラグ)”で、既定によって(`$LinkerScript_FLASH`)が追加されます。これは構築中に適切な(デバイス名)_flash.ldファイルによって置き換えられます。同様に(`$LinkerScript_SRAM`)は適切な(デバイス名)_sram.ldファイルで置き換えられます。
- あなた独自のリンカ スクリプト任意選択の-T“独自リンカスクリプト.ld”で(`$LinkerScript_FLASH`)または(`$LinkerScript_SRAM`)を置き換えることによって既定フラッシュ リンカ スクリプトを上書きすることができます。

注: これらのデバイス特有リンカ スクリプトは“ProjectFolder¥Linkerscripts”ディレクトリで利用可能です。プロジェクト作成後にデバイスを変える場合、Microchip Studioは選んだデバイス用の正しいリンカ スクリプトを自動的に追加します。

Arm®アセンブラ任意選択

表7-4. Arm®アセンブラ任意選択

任意選択	説明
最適化任意選択	
アセンブラ フラグ (書式領域)	その他のアセンブラ フラグ
インクルード パス (書式領域)	ここで基本構造と基盤特有のインクルードしたファイルへのパスの追加、優先付け、編集ができます。
-v	アセンブラ出力に於ける公表の版
-W	警告を禁止
デバッグ任意選択	
デバッグ レベル(引き落としメニュー) None, (-g)	シンボル デバッグとソース挿入デバッグを許可

Arm®前処理アセンブラ任意選択

表7-5. Arm®前処理アセンブラ任意選択

任意選択	説明
最適化任意選択	
アセンブラ フラグ (書式領域)	その他のアセンブラ フラグ
インクルード パス (書式領域)	ここで基本構造と基盤特有のインクルードしたファイルへのパスの追加、優先付け、編集ができます。
-v	アセンブラ出力に於ける公表の版
-W	警告を禁止
デバッグ任意選択	
デバッグ レベル(引き落としメニュー) None, -Wa -g	シンボル デバッグとソース挿入デバッグを許可

7.3. Arm® GNUツールチェーン任意選択

7.3.1. Arm®/GNU共通任意選択

- `Thumb(-mthumb)/Arm(-marm)` : プロセッサ動作選択を許します。

7.3.2. コンパイラ任意選択

7.3.2.1. プリプロセッサ (前処理部)

- `-nostdinc` : ヘッダファイルに対して標準システムディレクトリを検索しない。`-I`任意選択指定したディレクトリと適合する場合の現在のファイルのディレクトリだけが検索されます。
- `-E` : 前処理段後に停止、コンパイラを全く動かしません。出力は前処理されたソースコードの形式で、それは標準出力に送られます。前処理が必要でない入力ファイルは無視されます。

7.3.2.2. シンボル

- `-D`
 - `-D 名前` : 1の定義でマクロとして名前を事前定義します。
 - `-D 名前=値` : 値の定義でマクロとして名前を事前定義します。定義の内容は`#define`指令で第3段階への変換中にそれらが現れたかのように通票化されて処理されます。特に、定義は埋め込まれた新規行(改行)文字によって切り詰められます。
- `-U` : 組み込み、または`-D`任意選択で提供されたどちらかの以前のどの名前の定義も取り消します。

`-D`と`-U`の任意選択はそれらがコメント行で与えられた順に処理されます。全ての`-iマクロファイル`と`-iインクルードファイル`の任意選択は全ての`-D`と`-U`の任意選択後に処理されます。

7.3.2.3. ディレクトリ

- `-I ディレクトリ` : ヘッダファイルに対して検索されるべきディレクトリの一覧にディレクトリを追加します。`-I`によって名付けられたディレクトリは標準システムインクルードディレクトリの前に検索されます。ディレクトリが標準システムインクルードディレクトリの場合、システムディレクトリに対する既定検索順とシステムヘッダの特別な扱いが覆されないことを保証するため、この任意選択は無視されます。

7.3.2.4. 最適化

- コードを生成する時に使われる最適化のレベルを指定する一般的なスイッチの '`-O<最適化レベル>`' があります。
 - `-Os` : 生成されるコードがコード量に対して最適化されるべきことを合図します。コンパイラは生成したコードの実行性能について気にしません。
 - `-O0` : 最適化なし。GCCはデバッグが容易ですが、下で概説される増分最適化レベルでよりも遅く、より大きなコードを生成します。
 - `-O1`または`-O` : これは速度と量の両方に対して最適化します。殆どの文はC/C++コードでと同じ順に実行され、殆どの変更は生成されたコードで見つけることができます。これはデバッグに対してかなり適合されたコードを作り、これが既定です。
 - `-O2` : コード量を大幅に増加するかもしれないいくつかの最適化を除いてGCCでの最大最適化を有効にします。これは命令の計画化を許可し、そしてこれはこれからの恩恵を受けるだろうCPU基本設計に対して、データの危険要因と依存性のためにCPU失速周期を最小にするために調整されることを命令に許します。この任意選択全体は小さくて高速ですが、デバッグし辛いコードを作ります。
 - `-O3` : コード量を大幅に増加するかもしれないが、`-O2`と`-O1`の最適化レベルに比べて性能が増加するいくつかの特別な性能最適化を有効にします。これは実行関数インライン化を含みます。
- 他の最適化任意選択
 - `-ffunction-sections`と`-fdata-sections` : 目的対象が任意の区分(section)を支援する場合に出力ファイルで各関数とデータの項目をその区分に配置します。関数の名前またはデータ項目の名前は出力ファイルでの区分の名前を決めます。
 そのように行うことからかなりの恩恵がある時にだけこれらの任意選択を使ってください。これらの任意選択指定時、アセンブラとリンクはより大きなオブジェクトと実行可能なファイルを作成し、またより遅くなるでしょう。
 - `-funroll-loops` : 反復回数が既知の時に繰り返しの展開を実行します。コード量が関係しない場合、'`-O3`'スイッチに加えて'`-funroll-loops`'スイッチを用いることによってGCCが繰り返しの展開することでいくつかの追加の性能が得られるかもしれません。

7.3.2.5. デバッグ

- `-g` レベル (デバッグ レベル)
 - `-g1` : デバッグの計画をしないプログラムの部分で逆追跡するのに十分な最小の情報を生成します。これは関数と外部変数の記述を含みますが、局所変数についての情報と行番号が全くありません。
 - `-g2` : これは既定デバッグ レベルです。
 - `-g3` : これはプログラムに存在する全てのマクロ定義のような追加の情報を含みます。いくつかのデバッグは`-g3`使用時にマクロ展開を支援します。

7.3.2.6. 警告

- `-Wall` : 全ての警告を見せます。
- `-Werror` : 警告を異常として見せます。
- `-fsyntax-only` : 構文誤りについてコードを調べますが、それ以上何もしません。
- `-pedantic` : ISO C制限によって要求された全ての警告を発行します。禁じられた拡張を使う全てのプログラムとISO Cに従わないいくつかの他のプログラムを却下します。(稀に少数が必要とされるISO Cの版番号を指定する`-ansi`または`-std`の任意選択を必要とするにも関わらず、)有効なISO Cプログラムはこの任意選択有りまたはなしで正しくコンパイルされるべきです。けれども、この任意選択なしで、或るGNU拡張と伝統的なC機能もまた支援されます。この任意選択でそれらは却下されます。
- `-pedantic-errors` : 拘子定規な(上記制限違反の)警告が異常として生成されます。
- `-w` : 全ての警告メッセージを禁止します。

7.3.2.7. その他

- `-v` : 冗長任意選択。これはコンパイルの段階を走行するように実行された命令を(標準異常出力で)出力します。また、コンパイラ駆動部プログラムの版番号と適切な前処理部(プリプロセッサ)とコンパイラも出力します。
- `-ansi` : ANSIプログラムを支援します。これは(Cコードをコンパイルする時の)ISO C90と不適合なGCCの或る機能をOFFにします。Cコンパイラに対してはインライン キーワード'だけでなくC++様式の//注釈の認証も禁止します。`-ansi`任意選択は非ISOプログラムを不必要に却下させません。それについては`-ansi`に加えて`-pedantic`が必要とされます。

7.3.3. リンカ任意選択

7.3.3.1. 全般

- `-Wl,任意選択` : リンカに任意選択として任意選択を渡します。任意選択がカンマ(',')を含む場合、それはカンマで複数の任意選択に分割します。任意選択に引数を渡すのにこの構文を使うことができます。例えば、`'-Wl,-Map,output.map'`はリンカに`'-Map output.map'`を渡します。
- `-Wl, -nostartfiles` : リンク時に標準システム始動ファイルを使いません。`-nostdlib`または`-nodefaultlibs`が使われない限り、標準的に標準システムライブラリが使われます。
- `-Wl,-nodefault` : リンク時に標準システムライブラリを使いません。あなたが指定するライブラリだけがリンカへ渡されます。`-static-libgcc`や`-share d-libgcc`のようなシステムライブラリのリンクを指定する任意選択は無視されます。`-nostartfiles`が使われない限り、標準的に標準始動ファイルが使われます。コンパイラは`memcpy`, `memset`, `memcpy`, `memmove`に対する呼び出しを生成するかもしれません。これらの入り口は通常、`lib.c`での入り口によって解決されます。これらの入り口点はこの任意選択が指定される時にいくつかの他の機構を通して供給されるべきです。
- `-Wl,-nostdlib` : リンク時に標準始動ファイルまたは標準ライブラリを使いません。
`-nostdlib`と`-nodefaultlibs`によって迂回される標準ライブラリの1つは`libgcc.a`で、GCCの内部サブルーチンのライブラリは特定マシンの短所を克服するのに、またはいくつかの言語に対する特別な要求に使います。殆どの場合、例え他の標準ライブラリを避けたい時でも、`libgcc.a`が必要です。換言すると、`-nostdlib`や`-nodefaultlibs`を指定すると、通常、更に`-lgcc`も指定すべきです。これは内部GCCライブラリのサブルーチンに対して未解決の参照を持たないことを保証します。
- `-Wl,-s` : 実行可能形式から全てのシンボル表と再配置情報を削除。
- `-Wl,-static` : 動的リンクを支援するシステムに於いて、これは共有されるライブラリとのリンクを防ぎます。他のシステムに於いて、この任意選択は無効です。
- `-Wl,-Map` : 配置(`map`)ファイルを生成。

7.3.3.2. ライブラリ

- `-Wl,-ライブラリ` : リンク時にライブラリと名付けられたライブラリを検索。
この任意選択を書く命令での場所で違いを生じます。リンカはそれらが指定された順にライブラリとオブジェクトのファイルを検索して処理します。故に、`foo.o -lz bar.o`はファイルの`foo.o`の後、`bar.o`の前にライブラリ`z`を検索します。
リンカはライブラリに対してディレクトリの標準一覧を検索し、これは実際には`liblibrary.a`と名付けられるファイルです。その後リンカは名前によって正確に指定されていたかのようにこのファイルを使います。

- `-Wl, Ldir` : `-l`に対して検索すべきディレクトリの一覧に`dir`ディレクトリを追加します。

7.3.3.3. 最適化

- `-Wl, --gc-sections` : 未使用領域をゴミ回収(ガベージコレクション)します。
未使用入力領域のゴミ回収を許可します。この任意選択を支援しない目的対象では無視されます。(このゴミ回収を実行しない)既定の動きはコマンド行で`--no-gc-sections`を指定することによって元へ戻すことができます。`--gc-sections`ほどの入力区分がシンボルと再配置の調査によって使われるかを決めます。入りロシンボルを含んでいる区分とコマンド行で定義されないシンボルを含んでいる全ての区分は、動的オブジェクトによって参照されるシンボルを含んでいる区分として維持されます。
- `-funsafe-math-optimizations` : 安全でない数値演算最適化を許可します。
- `-ffast-math` : 高速数値演算を許可します。
- `-fpic` : 位置無関係なコードを生成します。

7.3.4. アセンブラ任意選択

- `-I` : `.include`ディレクトリ(`.include`参照)で指定したファイルに対して検索するためにディレクトリの一覧にパスを追加するにはこの任意選択を使ってください。様々なパスをインクルードする必要の度毎に`-I`を使うかもしれません。現在の作業ディレクトリが常に最初に検索され、その後、それらがコマンド行で指定されたのと同じ順(左から右へ)でディレクトリが検索されます。
- `-v` : 版番号を公表します。
- デバッグ(`-g`) : デバッグレベルを許可するのにこの任意選択を使います。

7.3.5. 前処理アセンブラ任意選択

- `-I` : `.include`ディレクトリ(`.include`参照)で指定したファイルに対して検索するためにディレクトリの一覧にパスを追加するにはこの任意選択を使ってください。様々なパスをインクルードする必要の度毎に`-I`を使うかもしれません。現在の作業ディレクトリが常に最初に検索され、その後、それらがコマンド行で指定されたのと同じ順(左から右へ)何れかの`-I`ディレクトリを検索します。
- `-v` : 版番号を公表します。
- デバッグ(`Wa,-g`) : デバッグレベルを許可するのにこの任意選択を使います。

7.3.6. 纏め部(Archiver)任意選択

- `-r` : 纏め部に対して既存ファイル置換または新ファイル挿入。

7.4. バイナリユーティリティ (Binutils)

以下のArm GNUバイナリユーティリティが利用可能です。

- `arm-none-eabi-ld` : GNUリンカ
- `arm-none-eabi-as` : GNUアセンブラ
- `arm-none-eabi-addr2line` : アドレスをファイル名と行番号に変換
- `arm-none-eabi-ar` : 纏め部に対する作成、変更、抽出用ユーティリティ
- `arm-none-eabi-c++filt` : 符号化されたC++シンボルを整理するための選別部
- `arm-none-eabi-nm` : オブジェクトファイルからシンボルを一覧
- `arm-none-eabi-objcopy` : オブジェクトファイルを複写と変換
- `arm-none-eabi-objdump` : オブジェクトファイルから情報を表示
- `arm-none-eabi-ranlib` : 纏め部の内容に対して指標を生成
- `arm-none-eabi-readelf` : 何れかのELF形式オブジェクトファイルから情報を表示
- `arm-none-eabi-size` : オブジェクトまたは纏め部のファイルの区分(section)の大きさを一覧
- `arm-none-eabi-strings` : ファイルから印刷可能な文字列を一覧
- `arm-none-eabi-strip` : シンボルを破棄

各ユーティリティについてのより多くの情報に関しては`<ユーティリティ名> --help`のヘルプ命令で構築を使ってください。

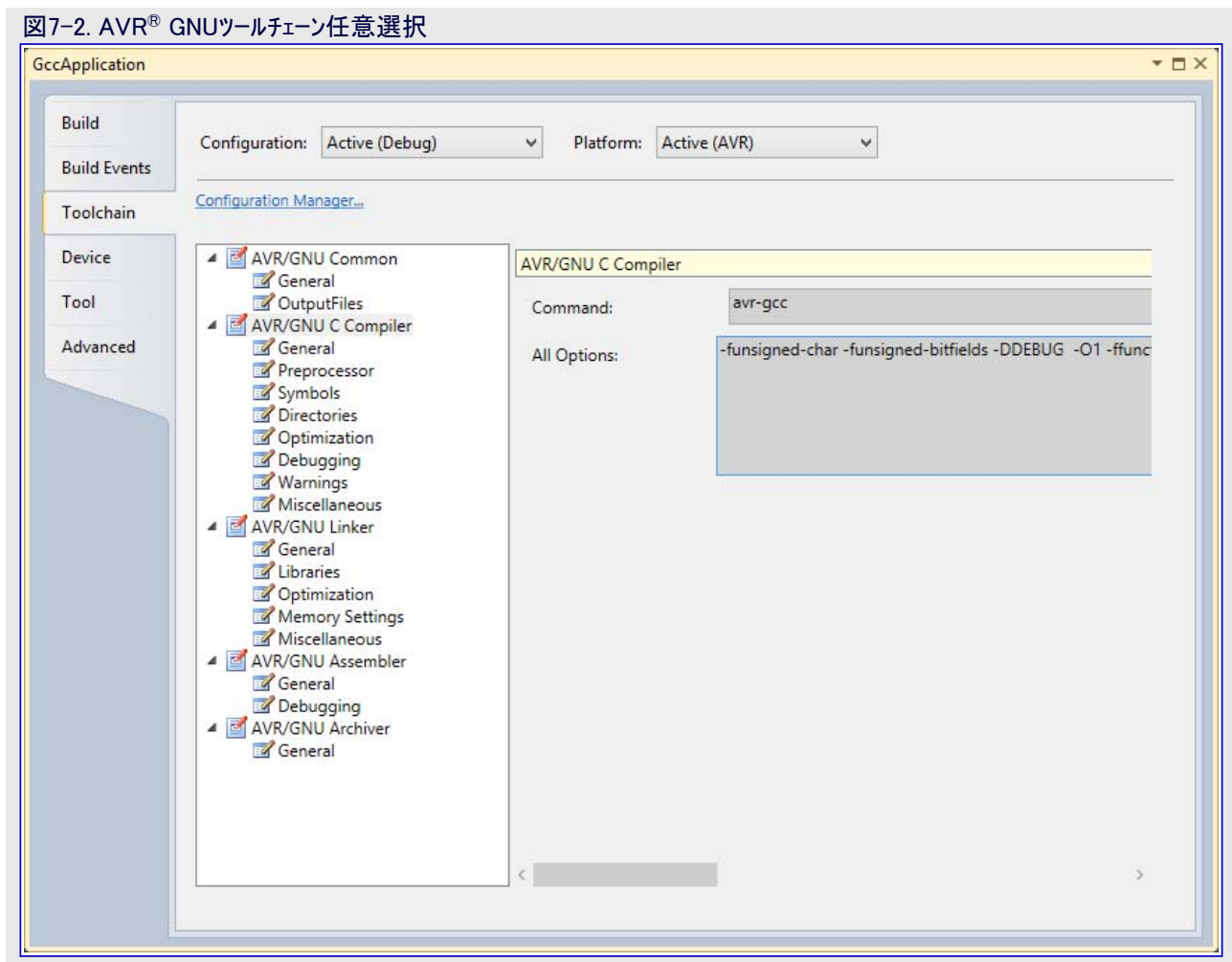
7.5. AVR[®]コンパイラとツールチェーン任意選択:GUI

AVR GNUツールチェーンでヘルプを得るには以下を行うことができます。

- Microchip Studioでの使い方と一般的なパラメータについての情報に関しては「[3.2.7. GCC[®]オブジェクト任意選択と構成設定](#)」項を調べてください。
- AVR libc実装に関するAPI参照は[ここ](#)で見つけることができます。
アルファベット順API目次は[ここ](#)で調べることができます。
- GCCについての全般情報に関しては公式の[GNU GCCウェブサイトを](#)訪ねてください。
- 代わりに、`avr32-gcc --help`を書いてそのコマンド出力でパラメータのいくつかの説明を見ることができます。





本項はMicrochip Studioの前処理部からAVR GNUツールチェーンに対して利用可能なGUI任意選択を説明します。

図7-2. AVR® GNUツールチェーン任意選択



AVR® GNU C コンパイラ任意選択

表7-6. AVR® GNU Cコンパイラ任意選択

任意選択	説明
全般任意選択	
-mcall-prologues	プロローグとエピローグの機能にサブルーチンを使用
-mno-interrupts	割り込み禁止なしでスタックポインタを変更
-funsigned-char	既定char型は符号なし
-funsigned-bitfield	既定ビット領域は符号なし
プロセッサ任意選択	
-nostdinc	システムディレクトリを検索しない。
-E	前処理(プリプロセッサ)のみ
シンボル任意選択	
ソース内のシンボルのいくつかを定義(-D)または定義解除(-U)することができます。新しいシンボル宣言は下のインターフェース鉤を用いて追加、変更、再整理をすることができます。	
<ul style="list-style-type: none"> •  新しいシンボル追加。これと以下の全アイコンはMicrochip Studioインターフェースの他の部分で同じ意味で再利用されます。 •  シンボル削除 •  シンボル編集 •  解析順に於いてシンボルを上に移動 •  解析順に於いてシンボルを下に移動 	
インクルードディレクトリ	
インクルードされるヘッダと定義指令の	全てを含み、シンボルと同じインターフェースを用いて編集することができます。
最適化任意選択	
最適化レベル(引き落としメニュー) -O0, -O1, -O2, -O3, -Os	最適化なし、速度最適化(レベル1~3)、大きさ最適化
他の最適化フラグ(手入力方式)	ここで基盤と必要条件に対して指定する最適化フラグを書くべきです。
-ffunction-sections	ゴミ回収用関数準備。関数が決して使われないなら、メモリがかき集められます。
-fpack-struct	構造体メンバを共に一括化
-fshort-enums	列挙型によって必要とされる多くのバイトとしてだけ割り当て
-mshort-calls	8Kバイトを超えるデバイスでRJMP/RCALL制限範囲命令を使用
デバッグ任意選択	
デバッグレベル(引き落としメニュー) none, -g1, -g2, -g3	ソースコードで残されるまたは挿入されるコードとヘッダを追跡してデバッグするレベルを指定
他のデバッグ任意選択(書式領域)	基本構造特有デバッグ任意選択
警告メッセージ出力任意選択	
-Wall	全ての警告
-Werror	警告を異常に格上げ
-fsyntax-only	構文だけ検査
-pedantic	GNUに対する適合性検査、非標準プログラミング実行で警告揭示
-pedantic-errors	同上、加えて警告を異常に格上げ
その他の任意選択	
他のフラグ(書式領域)	他のプロジェクト特有フラグを入力
-v	詳細
-ansi	ANSIプログラムを支援
-save-temps	一時ファイルを削除しない

AVR® GCCリンカ任意選択

表7-7. AVR® GCCリンカ任意選択

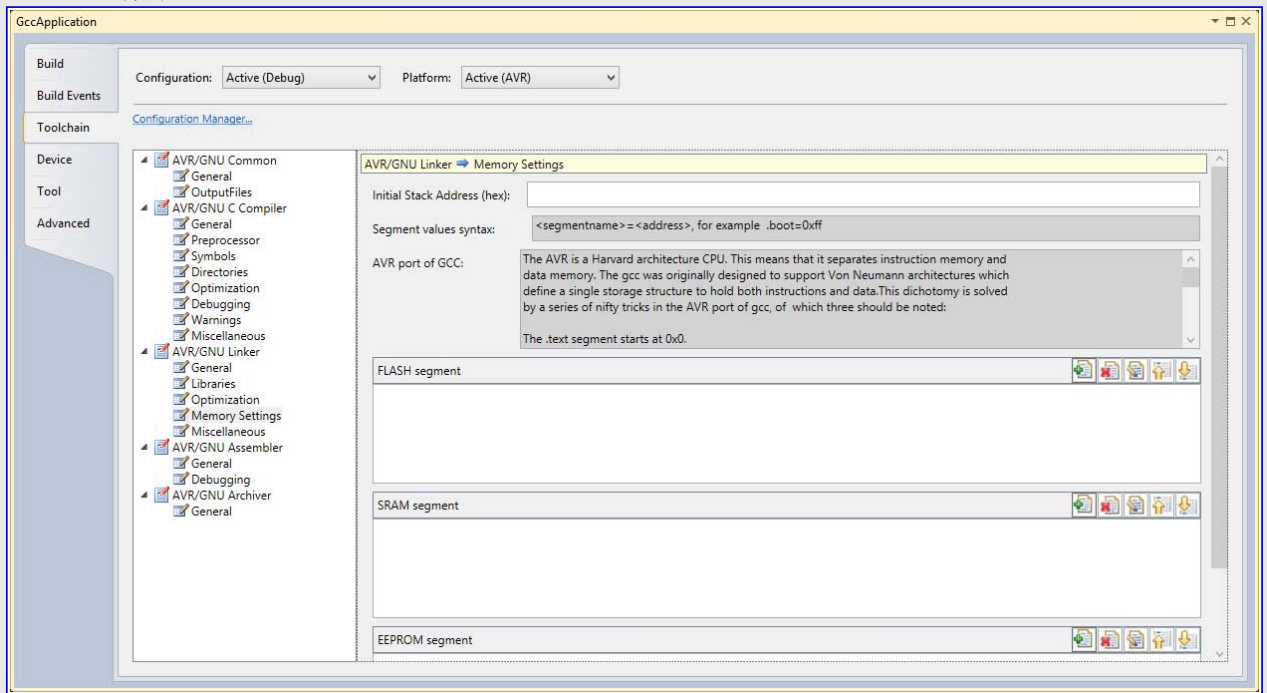
任意選択	説明
-Wl -nostartfiles	標準ファイルを使わない。
-Wl -nodefault	既定ライブラリを使わない。
-Wl -nostdlib	始動または既定ライブラリなし
-Wl -s	全てのシンボル情報を省く。
-Wl -static	静的にリンク
ライブラリ任意選択	
ライブラリ -Wl, -l (書式領域)	 の鈕を用いて、ここでライブラリ名の追加、優先付け、編集ができます。
ライブラリ検索パス -Wl, -L (書式領域)	上と同じインターフェースでリンクが動的にリンクされるライブラリを探す所のパスの追加、優先付け、編集ができます。
最適化任意選択	
-Wl, -gc-sections	未使用領域の回収
--rodata-writable	書き込み可能空間に読み込み専用データを配置
-mrelax	分岐緩和
その他の任意選択	
他のリンカフラグ (書式領域)	他のプロジェクト特有フラグを入力

メモリ設定

メモリ区分(セグメント)を構成設定することが可能なダイアログを表示してください。(区分値を指定する構文: <区分名>=<アドレス> 例えば、boot=0xFF)

アドレスは0xを前に付けた16進数として与えられなければなりません。これはフラッシュメモリに対して語アドレス、SRAMとEEPROMに対してバイトアドレスとして解釈されます。

図7-3. メモリ設定



GCCのAVR[®]ポートについての注意

AVRはハードウェア基本構造のCPUで、これは命令メモリとデータメモリを分けることを意味します。GCCは元々命令とデータの両方を保持するために単一の記憶構造を定義するノーマン基本構造を支援するように設計されました。この二分法はGCCのAVRポートで巧みな策略の連続によって解決され、その内の以下の3つに注意されるべきです。

- 0x0で始まる.text区分
- 0x800000で始まる.data区分
- 0x810000で始まる.eeprom区分

これらの特殊性はGUIによる方法で抽象化されてしまっていますが、使用者は再配置した区分を持つプロジェクトの構築時に真実を見るでしょう。

フラッシュメモリに対する再配置定義は任意選択としてavr-gcc経由でGNUリンカに渡されます。

- `-Wl,-section-start=bootloader=0x1fc00`

バイトアドレスを得るためにアドレスが2倍されることに注意してください。

.data区分に対する再配置定義は以下として渡されます。

- `-Wl,-section-start=newdatasegment=0x800`

AVR[®]アセンブラ任意選択

表7-8. AVR[®]アセンブラ任意選択

任意選択	説明
最適化任意選択	
アセンブラフラグ (書式領域)	その他のアセンブラフラグ
インクルードパス (書式領域)	ここで基本構造と基盤特有のインクルードしたファイルへのパスの追加、優先付け、編集ができます。
-v	アセンブラ出力で版番号を公表
デバッグ任意選択	
デバッグレベル(引き落としメニュー) None, -Wa -g	シンボルデバッグとソース挿入デバッグを許可

7.6. 一般的に使用される任意選択

7.6.1. コンパイラ任意選択

7.6.1.1. 全般

- `-funsigned-char` : 殆どのマシンの各々はcharが何であるべきかに対する既定を持ちます。これは既定によって多分、符号なし文字 (unsigned char)または符号付き文字(signed char)のどちらかです。この任意選択は既定char形式が符号なし (unsigned)であることを表しています。
- `-funsigned-bitfields` : これらの任意選択は宣言が符号付き(signed)または符号なし(unsigned)のどちらも使わない時に、ビット領域が符号付き(signed)または符号なし(unsigned)のどちらかを制御します。この任意選択は既定ビット領域形式が符号なし(unsigned)であることを表しています。

7.6.1.2. 前処理部(プリプロセッサ)

- `-nostdinc` : ヘッドファイルに対して標準システムディレクトリを検索しません。`-I`任意選択で指定したディレクトリ(と割り当てたなら、現在のファイルのディレクトリ)だけが検索されます。
- `-E` : 前処理段後に停止、コンパイラを全く動かしません。出力は前処理されたソースコードの形式で、それは標準出力に送られます。前処理が必要でない入力ファイルは無視されます。

7.6.1.3. シンボル

- `-D`
 - `-D 名前` : 1の定義でマクロとして名前を事前定義します。例えば、
 - `-D 名前=値` : 値の定義でマクロとして名前を事前定義します。定義の内容は`#define`指令で第3変換段階中にそれらが現れたかのように通票化されて処理されます。特に、定義は埋め込まれた新規行(改行)文字によって切り詰められます。
- `-U` : 組み込み、または`-D`任意選択で提供されたどちらかの以前のどの名前の定義も取り消します。

`-D`と`-U`の任意選択はそれらがコメント行で与えられた順に処理されます。全ての`-iマクロファイル`と`-iインクルードファイル`の任意選択は全ての`-D`と`-U`の任意選択後に処理されます。

7.6.1.4. ディレクトリ

- `-I` ディレクトリ: ヘッダ ファイルに対して検索されるべきディレクトリの一覧にディレクトリを追加します。`-I`によって名付けられたディレクトリは標準システム インクルード ディレクトリの前に検索されます。ディレクトリが標準システム インクルード ディレクトリの場合、システム ディレクトリに対する既定検索順とシステム ヘッダの特別な扱いが覆されないのを保証するために、この任意選択は無視されます。

7.6.1.5. 最適化

- コードを生成する時に使われる最適化のレベルを指定する一般的なスイッチの '`-O<最適化レベル>`' があります。
 - `-Os`: 生成されるコードがコード量に対して最適化されるべきことを合図します。コンパイラは生成したコードの実行性能について気にしません。
 - `-O0`: 最適化なし。GCCはデバッグが容易ですが、下で概説される増分最適化レベルでよりも遅く、より大きなコードを生成します。
 - `-O1`または`-O`: これは速度と量の両方に対して最適化します。殆どの文はC/C++コードでと同じ順に実行され、殆どの変更は生成されたコードで見つけることができます。これはデバッグに対してかなり適合されたコードを作ります。これが既定です。
 - `-O2`: コード量を大幅に増加するかもしれないいくつかの最適化を除いてGCCでの最大最適化を有効にします。これは命令の計画化を許可し、そしてこれはこれからの恩恵を受けるだろうCPU基本設計に対して、データの危険要因と依存性のためにCPU失速周期を最小にするために調整されることを命令に許します。この任意選択全体は小さくて高速ですが、デバッグし辛いコードを作ります。
 - `-O3`: コード量を大幅に増加するかもしれないが、`-O2`と`-O1`の最適化レベルに比べて性能が増加するいくつかの特別な性能最適化を有効にします。これは実行関数インライン化を含みます。
- 他の最適化任意選択
 - `--function-sections`と`--fdata-sections`: 目的対象が任意の区分(section)を支援する場合に出力ファイルで各関数とデータの項目をその区分に配置します。関数の名前またはデータの名前は出力ファイルでの区分の名前を決めます。
 そのように行うことからかなりの恩恵がある時にだけこれらの任意選択を使ってください。これらの任意選択指定時、アセンブラとリンクはより大きなオブジェクトと実行可能なファイルを作成し、またより遅くなるでしょう。
 - `--funroll-loops`: 反復回数が既知の時に繰り返しの展開を実行します。コード量が関係しない場合、'`-O3`' スwitchに加えて '`--funroll-loops`' スwitchを用いることによってGCCが繰り返しの展開することでいくつかの追加の性能が得られるかもしれません。

7.6.1.6. デバッグ

- `-g` レベル (デバッグ レベル)
 - `-g1`: デバッグの計画をしないプログラムの部分で逆追跡をするのに十分な最小の情報を生成します。これは関数と外部変数の記述を含みますが、局所変数についての情報と行番号が全くありません。
 - `-g2`: これは既定デバッグ レベルです。
 - `-g3`: これはプログラムに存在する全てのマクロ定義のような追加の情報を含みます。いくつかのデバッグは`-g3`使用時にマクロ展開を支援します。

7.6.1.7. 警告

- `-Wall`: 全ての警告を見せます。
- `-Werror`: 警告を異常として見せます。
- `-fsyntax-only`: 構文誤りについてコードを調べますが、それ以上何もしません。
- `-pedantic`: ISO C制限によって要求された全ての警告を発行し、禁じられた拡張を使う全てのプログラムとISO Cに従わないいくつかの他のプログラムを却下します。(稀に少数が必要とされるISO Cの版番号を指定する`-ansi`または`-std`の任意選択を必要とするにも関わらず、)有効なISO Cプログラムはこの任意選択有りまたはなしで正しくコンパイルされるべきです。けれども、この任意選択なしで、或るGNU拡張と伝統的なC機能もまた支援されます。この任意選択でそれらは却下されません。
- `-pedantic-errors`: 拘子定規な(上記制限違反の)警告が異常として生成されます。
- `-w`: 全ての警告メッセージを禁止します。

7.6.1.8. その他

- `-v`: 冗長任意選択。これはコンパイルの段階を走行するように実行された命令を(標準異常出力で)出力します。また、コンパイラ駆動部プログラムの版番号と適切な前処理部(プリプロセッサ)とコンパイラも出力します。
- `-ansi`: ANSIプログラムを支援します。これは(Cコードをコンパイルする時の)ISO C90と不適合なGCCの或る機能をOFFにします。コンパイラに対してはインライン キーワードだけでなくC++様式の//注釈の認証も禁止します。`-ansi`任意選択は非ISOプログラムを不必要に却下させません。それについては`-ansi`に加えて`-pedantic`が必要とされます。

7.6.2. リンカ任意選択

7.6.2.1. 全般

- `-Wl,任意選択` : リンカに任意選択として任意選択を渡します。任意選択がカンマ(‘,’)を含む場合、それはカンマで複数の任意選択に分割します。任意選択に引数を渡すのにこの構文を使うことができます。例えば、`’-Wl,-Map,output.map’`はリンカに`’-Map output.map’`を渡します。
- `-Wl,-nostartfiles` : リンク時に標準システム始動ファイルを使いません。`-nostdlib`または`-nodefaultlibs`が使われない限り、標準的に標準システムライブラリが使われます。
- `-Wl,-nodefault` : リンク時に標準システムライブラリを使いません。あなたが指定するライブラリだけがリンカへ渡されます。`-static-libgcc`や`-shared-libgcc`のようなシステムライブラリのリンクを指定する任意選択は無視されます。`-nostartfiles`が使われない限り、標準的に標準始動ファイルが使われます。コンパイラは`memcmp`, `memset`, `memcpy`, `memmove`に対する呼び出しを生成するかもしれませんが。これらの入り口は通常、`lib.c`での入り口によって解決されます。これらの入り口点はこの任意選択が指定される時にいくつかの他の機構を通して供給されるべきです。
- `-Wl,-nostdlib` : リンク時に標準始動ファイルまたは標準ライブラリを使いません。
`-nostdlib`と`-nodefaultlibs`によって迂回される標準ライブラリの1つは`libgcc.a`で、GCCの内部サブルーチンのライブラリは特定マシンの短所を克服するのに、またはいくつかの言語に対する特別な要求に使います。殆どの場合、例え他の標準ライブラリを避けたい時でも、`libgcc.a`が必要です。換言すると、`-nostdlib`や`-nodefaultlibs`を指定すると、通常、更に`-lgcc`も指定すべきです。これは内部GCCライブラリのサブルーチンに対して未解決の参照を持たないことを保証します。
- `-Wl,-s` : 実行可能形式から全てのシンボル表と再配置情報を削除。
- `-Wl,-static` : 動的リンクを支援するシステムに於いて、これは共有されるライブラリとのリンクを防ぎます。他のシステムに於いて、この任意選択は無効です。

7.6.2.2. ライブラリ

- `-Wl,-ライブラリ` : リンク時にライブラリと名付けられたライブラリを検索。
この任意選択を書く命令での場所で違いを生じます。リンカはそれらが指定された順にライブラリとオブジェクトのファイルを検索して処理します。故に、`foo.o -lz bar.o`はファイルの`foo.o`の後、`bar.o`の前にライブラリを検索します。
リンカはライブラリに対してディレクトリの標準一覧を検索し、これは実際には`liblibrary.a`と名付けられるファイルです。その後、リンカは名前によって正確に指定されていたかのようにこのファイルを使います。
- `-Wl, Ldir : -l` : `-l`に対して検索すべきディレクトリの一覧に`dir`ディレクトリを追加します。

7.6.2.3. 最適化

- `-Wl, --gc-sections` : 未使用領域をゴミ回収(ガベージコレクション)します。
未使用入力領域のゴミ回収を許可します。この任意選択を支援しない目的対象では無視されます。(このゴミ回収を実行しない)既定の動きはコマンド行で`--no-gc-sections`を指定することによって元へ戻すことができます。`--gc-sections`はどの入力区分がシンボルと再配置の調査によって使われるかを決めます。入り口シンボルを含んでいる領域とコマンド行で定義されないシンボルを含んでいる全ての区分は、動的オブジェクトによって参照されるシンボルを含んでいる区分として維持されます。
- `--rodata-writable` : 書き込み可能なデータ領域に読み込み専用データを配置します。

7.6.3. アセンブラ任意選択

- `-I` : `.include`ディレクトリ(`.include`参照)で指定したファイルに対して検索するためにディレクトリの一覧にパスを追加するにはこの任意選択を使ってください。様々なパスをインクルードする必要の度毎に`-I`を使うかもしれません。現在の作業ディレクトリが常に最初に検索されます。その後、それらがコマンド行で指定されたのと同じ順(左から右へ)何れかの`’-I’`ディレクトリを検索します。
- `-v` : 版番号を公表します。

7.7. 8ビット特有AVR® GCCコマンド行任意選択

7.7.1. AVR® コンパイラ

7.7.1.1. 全般

- `-mcall-prologues` : プロローグ/エピローグの機能は適切なサブルーチンを呼ぶように拡充されます。コード量がより小さくなるでしょう。
- `-mno-interrupts` : 割り込み禁止なしでスタックポインタを変更します。生成したコードはハードウェア割り込みに適合しません。コード量がより小さくなるでしょう。
- `-mno-tablejump` : 表参照分岐命令を生成しません(GCC 4.5.1 `coz`から削除された`-fno-jump-tables`と同じです)。
- `-msize` : `asm`ファイルで命令量を出力します(AVR-GCC `coz`から削除された命令長を出力する`-dp`スイッチ使用と同じです)。

7.7.1.2. 最適化

- `-fpack-struct` : 指定値なしでは、全ての構造体のメンバを穴なしで共に詰め込みます。(小さな2のべき乗でなければならない)値が指定されると、最大整列を表すこの値に従って構造体メンバを詰め込みます(即ち、これよりも大きな既定整列必要条件を持つオブジェクトは次の選別位置で潜在的に整列されずに出力するでしょう)。
- `-fshort-enums` : 宣言した可能な値の範囲に対して必要なのと同じバイト数だけを列挙(enum)型に割り当てます。特に、列挙型は十分な場所を持つ最小の整数型に相当します。
- `-mshort-calls` : 8Kバイトを超えるデバイスで(制限された範囲の)RJMP/RCALL命令を使います。

7.7.1.3. その他

- `-save-temps` : 一時ファイルを削除しません。通常の”一時的な”中間ファイルを永久的に保管し、ソースファイルに基づいてそれらに名前を付けて現在のディレクトリにそれらを置きます。故に、`-c -save-temps`での`foo.c`コンパイルは`foo.o`だけでなく、`foo.i`と`foo.s`も生成します。これは例えばコンパイラが今や普通に統合された前処理部(プリプロセッサ)を使ったとしても、前処理された`foo.i`出力ファイルを作成します。

7.7.2. AVR® リンカ

7.7.2.1. 最適化

- `-mrelax` : 分岐緩和。リンカ緩和はリンカに’`-relax`’任意選択を渡すことによってリンカ内で許可されます。リンカに対する前処理部としてGCCを使うことは、’`-O2`’または’`-O3`’使用時、または明示的に’`-mrelax`’任意選択を使う時に、この任意選択が自動的にリンカへ渡されます。この任意選択が使われると、GCCは`lda.w`, `call`などのような疑似命令を出力します。リンカはその後に、入力ファイルが緩和可能として札付けされた場合、疑似命令を最終的なシンボルアドレスに関して可能な最善の命令に変換します。

7.8. 32ビット特有AVR® GCCコマンド行任意選択

7.8.1. 最適化

- `-mfast-float` : この切り替えは最適化されたAVR 32ビット浮動小数点ライブラリ関数のいくつかのIEEE非適合版を使って高速にします。この切り替えは’`-ffast-math`’任意選択が使われる場合に既定によって許可されます。
- `-funsafe-math-optimizations` : (a)引数と結果が有効であると仮定し、(b)IEEEとANSIの規格に違反するかもしれない浮動小数点演算に対する私的化を許します。リンク時に使われると、既定FPU制御語または他の同様な最適化を変更するライブラリまたは始動ファイルをインクルードするかもしれません。この任意選択はそれが算術関数に対するIEEEまたはISOの規則/仕様の正確な実装に依存するプログラムに対して不正な出力に終わり得るため、どの’`-O`’任意選択によってもONにされません。けれども、それはそれらの仕様の保証を必要としないプログラムに対してより速いコードを生じるかもしれません。’`-fno-signed-zeros`’、’`-fno-trapping-math`’、’`-fassociative-math`’、’`-freciprocal-math`’を許可します。
- `-ffast-math` : この任意選択は定義されるべき`_FAST_MATH_`前処理部(プリプロセッサ)マクロをもたらします。この任意選択はそれが算術関数に対するIEEEまたはISOの規則/仕様の正確な実装に依存するプログラムに対して不正な出力に終わり得るため、どの’`-O`’任意選択によってもONにされません。けれども、それはそれらの仕様の保証を必要としないプログラムに対してより速いコードを生じるかもしれません。これは’`-fno-math-errno`’、’`-funsafe-math-optimizations`’、’`-ffinite-math-only`’、’`-fno-rounding-math`’、’`-fno-signaling-nans`’、’`-fcx-limited-range`’を設定します。
- `-fpic` : 目的対象マシンに対して支援されるなら、共用ライブラリで使われるのに適した位置無関係コード(PIC:Position-Independent Code)を生成します。このようなコードは全域変位表(GOT:Global Offset Table)を通して全ての定数アドレスにアクセスします。動的ローダはプログラムが開始する時にGOT登録を解決します(動的ローダはGCCの一部ではなく、オペレーティングシステムの一部です)。リンクされた実行可能物に対するGOTの大きさがマシン仕様の最大量を超える場合、’`-fpic`’が動かないことを示すリンカからの異常メッセージを受け取ります。この場合、代わりに’`-fPIC`’で再コンパイルしてください。(これらの最大はSPARCで8K、m68kとRS/6000で32Kです。386はそのような制限を持ちません。)位置無関係コードは特別な支援を必要とし、従って、或るマシンでだけ動きます。386について、System Vに対してはPICを支援しますが、Sun 386iに対してはしません。IBM RS/6000用に生成されたコードは常に位置無関係です。このフラグが設定されると、`_pic`と`_PIC`のマクロが1に定義されます。
- `-mno-init-got` : 位置無関係コード(PIC)をコンパイルする時にそれを使う前に全域変位表(GOT)レジスタを初期化しません。
- `-masm-addr-pseudos` : この任意選択は既定によって許可され、GCCに直接関数呼び出しとシンボルアドレス取得の各々に対して`call`と`lda.w`の疑似命令を出力させます。’`-mno-asm-addr-pseudos`’のスイッチを指定することによってOFFにすることができます。これらの疑似命令を使う利点は、リンカ緩和が許可された場合にリンク時にリンカがこれらの命令を最適化することができることです。’`-mrelax`’任意選択は緩和可能なオブジェクトコードを生成すべきことをアセンブラに合図するためにGCCに渡すことができます。
- `-mforce-double-align` : 倍長語メモリアクセスに対して倍長語整列を強制します。

- `-mimm-in-const-pool` : レジスタ内への単一移動命令に適さない即値を移動することをGCCが必要とする時に、以下の2つの可能な選択を持ちます。それは現在の命令の近くの何処かのコード内に定数を置き(定数プール)、その後値を取得するために単一取得命令を使うことができます。またはメモリ設定を使うことなく、値を直接取得するために2つの即値命令を使うことができます。コードメモリからの取得が2つの単一1周期即値命令よりも速いなら、それらの即値を定数プールに置くことが速度に対して最も最適です。これは命令キャッシュを実装するMCU基本構造に対して度々真実で、一方内部フラッシュメモリから実行するコードを持つ基本構造は多分コードメモリから値を取得するのに数周期が必要でしょう。既定によってGCCは命令キャッシュを持つAVR 32ビット製品に対して定数プールを、フラッシュメモリに基づくMCUに対して2つの即値命令を使います。これは`'-mimm-in-const-pool'`任意選択か、または逆の`'-mno-imm-in-const-pool'`任意選択を使うことによって上書きすることができます。
- `-muse-rodata-sections` : 既定によってGCCはコード(.text)区分に読み込み専用データを出力します。コードメモリが遅い場合、利用可能ならば、別のより速いメモリに読み込み専用データを置くことで性能に関してもっと最適になるかもしれません。これは読み込み専用データを.rodata区分に置くことをGCCにさせる`'-muse-rodata-section'`スイッチを指定することによって行うことができます。その後リンクファイルは配置されるべき.rodata区分の内容の場所を指定することができます。フラッシュメモリからコードを走らせるシステムについて、けれども、これは読み出し専用データはフラッシュメモリに置かれ、その後始動で別のメモリ上に複製されなければならないことを意味し、そしてそれはこの仕組みで余分なメモリ使用が必要とされることを意味します。

7.8.2. デバッグ

- `-pg` : プログラムのgprofの分析に適したプロファイル情報を書くために追加のコードを生成します。それについてのデータを望むソースファイルをコンパイルする時にこの任意選択を使わなければならない、またリンク時にもそれを使わなければならない。
- `-p` : プログラムのprofの分析に適したプロファイル情報を書くために追加のコードを生成します。それについてのデータを望むソースファイルをコンパイルする時にこの任意選択を使わなければならない、またリンク時にもそれを使わなければならない。

7.8.3. AVR32 リンク

7.8.3.1. 最適化

- `-mfast-float` : 高速浮動小数点ライブラリを許可します。`'-funsafe-math-optimizations'`スイッチが指定される場合は既定によって許可されます。
- `-funsafe-math-optimizations` : (a)引数と結果が有効であると仮定し、(b)IEEEとANSIの規格に違反するかもしれない浮動小数点演算に対する私的化を許します。リンク時に使われると、既定FPU制御語または他の同様な最適化を変更するライブラリまたは始動ファイルをインクルードするかもしれません。この任意選択はそれが算術関数に対するIEEEまたはISOの規則/仕様の正確な実装に依存するプログラムに対して不正な出力に終わり得るため、どの`'-O'`任意選択によってもONにされません。けれども、それはそれらの仕様の保証を必要としないプログラムに対してより速いコードを生じるかもしれません。`'-fno-signed-zeros'`、`'-fno-trapping-math'`、`'-fassociative-math'`、`'-freciprocal-math'`を許可します。既定は`'-fno-unsafe-math-optimizations'`です。
- `-ffast-math` : この任意選択は定義されるべき`_FAST_MATH_`前処理部(プリプロセッサ)マクロをもたらします。この任意選択はそれが算術関数に対するIEEEまたはISOの規則/仕様の正確な実装に依存するプログラムに対して不正な出力に終わり得るため、どの`'-O'`任意選択によってもONにされません。けれども、それはそれらの仕様の保証を必要としないプログラムに対してより速いコードを生じるかもしれません。これは`'-fno-math-errno'`、`'-funsafe-math-optimizations'`、`'-ffinite-math-only'`、`'-fno-rounding-math'`、`'-fno-signaling-nans'`、`'-fcx-limited-range'`を設定します。
- `-fpic` : 目的対象マシンに対して支援されるなら、共用ライブラリで使われるのに適した位置無関係コード(PIC:Position-Independent Code)を生成します。このようなコードは全域変位表(GOT:Global Offset Table)を通して全ての定数アドレスにアクセスします。動的ローダはプログラムが開始する時にGOT登録を解決します(動的ローダはGCCの一部ではなく、オペレーティングシステムの一部です)。リンクされた実行可能物に対するGOTの大きさがマシン仕様の最大量を超える場合、`'-fpic'`が動かないことを示すリンクからの異常メッセージを受け取ります。この場合、代わりに`'-fPIC'`で再コンパイルしてください。(これらの最大はSPARCで8K、m68kとRS/6000で32Kです。386はそのような制限を持ちません。)位置無関係コードは特別な支援を必要とし、従って、或るマシンでだけ動きます。386について、System Vに対してはPICを支援しますが、Sun 386iに対してはしません。IBM RS/6000用に生成されたコードは常に位置無関係です。このフラグが設定されると、`_pic_`と`_PIC_`のマクロが1に定義されます。
- `-Wl,--direct-data` : 最適化時に直接データ参照を許します。`ld.a.w`を即値移動命令に変換することをリンクに許す(即ち、リンク緩和にする)には、リンクに`'-direct-data'`任意選択が与えられなければならない。

7.8.3.2. その他

- `-Xlinker[任意選択]` : リンカへの任意選択として任意選択を渡します。GCCが認証方法を知らないシステム特有リンカ任意選択を供給するのにこれを使うことができます。独立した引数を取る任意選択に渡したい場合、任意選択に対しての1回と引数に対しての1回で、2回`-Xlinker`を使わなければなりません。例えば、`-assert definitions`を渡すには`'-X linker -assert -Xlinker definitions'`を書かなければなりません。`-Xlinker "-assert definitions"`書き込みは、これがリンカが予期するものではない単一引数として文字列全体を渡すため、動きません。GNUリンカ使用时、通常、独立した引数としてよりも`option=value`構文を使ってリンカ任意選択に引数を渡すのがもっと便利です。例えば、`'-Xlinker -Map -Xlinker output.map'`よりもむしろ`'-Xlinker -Map=output.map'`を指定することができます。他のリンカはコマンド行任意選択に対してこの構文を支援しないかもしれません。

7.9. バイナリ ユーティリティ (Binutils)

以下のAVR 32ビットGNUバイナリ ユーティリティが利用可能です。

- `avr32-ld` : GNUリンカ
- `avr32-as` : GNUアセンブラ
- `avr32-addr2line` : アドレスをファイル名と行番号に変換
- `avr32-ar` : 纏め部に対する作成、変更、抽出用ユーティリティ
- `avr32-c++filt` : 符号化されたC++シンボルを整理するための選別部
- `avr32-nm` : オブジェクト ファイルからシンボルを一覧
- `avr32-objcopy` : オブジェクト ファイルを複写と変換
- `avr32-objdump` : オブジェクト ファイルから情報を表示
- `avr32-ranlib` : 纏め部の内容に対して指標を生成
- `avr32-readelf` : 何れかのELF形式オブジェクト ファイルから情報を表示
- `avr32-size` : オブジェクトまたは纏め部のファイルの区分(section)の大きさを一覧
- `avr32-strings` : ファイルから印刷可能な文字列を一覧
- `avr32-strip` : シンボルを破棄

各ユーティリティについてのより多くの情報に関しては`avr32-〈ユーティリティ名〉 --help`のヘルプ命令で構築を使ってください。

- GNUアセンブラ(GAS)、GNUリンカ、他のバイナリ ユーティリティについての全般情報に関しては公式の[GNU Binutilsウェブ サイト](#)を訪ねてください。

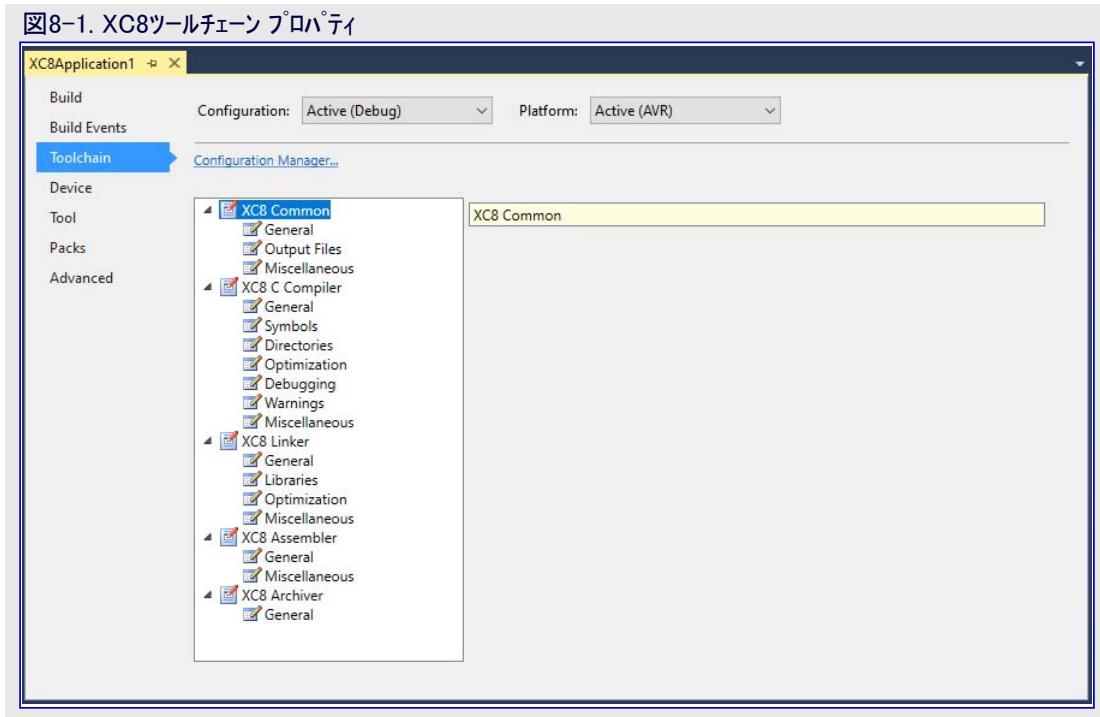
8. XC8ツールチェーン

8.1. 序説

Microchip XC8 コンパイラはCプログラミング言語用の自立型最適化ISO C99クロス コンパイラです。Microchip Studio XC8統合は8ビットAVR[®]マイクロ コントローラを支援します。XC8 コンパイラ コマンド行ドライバ、xc8-ccはCコード生成、アセンブリ、リンクの段階を含み、コンパイルの全ての面を実行するために呼び出されます。Microchip Studio XC8統合は全ての内部応用の複雑さを隠して全ての構築段階に対して一貫したインターフェースを提供します。

8.2. XC8コンパイラとツールチェーンの任意選択: GUIツールチェーン任意選択

本項はMicrochip StudioでXC8コンパイラに対して利用可能なGUI任意選択を説明します。







XC8共通任意選択

表8-1. XC8共通任意選択

任意選択	説明
全般任意選択	
-mrelax	長い形式の呼び出しと分岐の命令の最適化を制御
出力ファイル任意選択	
.hex	hexファイルを生成
.eep	eepファイルを生成
.usersignatures	使用者識票ファイルを生成
.lss	lssファイルを生成
その他の任意選択	
他の共通フラグ (書式領域)	他のプロジェクト特有フラグを入力

XC8 コンパイル任意選択

表8-2. XC8 コンパイル任意選択

任意選択	説明
全般任意選択	
-mcall-prologues	プロローグとエピローグの機能にサブルーチンを使用
-mno-interrupts	割り込み禁止なしでスタックポインタを変更
-funsigned-char	既定char型は符号なし
-funsigned-bitfield	既定ビット領域は符号なし
-nostdinc	システムディレクトリを検索しない。
-mext=cci	CCIを使用
シンボル任意選択	
ソース内のシンボルを定義(-D)または定義解除(-U)することができます。新しいシンボル宣言は下のインターフェース鉤を用いて追加、変更、再整理をすることができます。	
<ul style="list-style-type: none"> •  新しいシンボル追加。これと以下の全アイコンはMicrochip Studioインターフェースの他の部分で同じ意味で再利用されます。 •  シンボル削除 •  シンボル編集 •  解析順に於いてシンボルを上に移動 •  解析順に於いてシンボルを下に移動 	
インクルード デリクトリ	
インクルードされるヘッダと定義指令の	全てを含み、シンボルと同じインターフェースを用いて編集することができます。
最適化任意選択	
最適化レベル(引き落としメニュー) -O0, -O1, -O2, -O3, -Os, -Og	最適化なし、速度最適化(レベル1~3)、大きさ最適化、良好なデバッグ体験用最適化
-ffunction-sections	ゴミ回収用関数準備。関数が決して使われないなら、メモリがかき集められます。
-fpack-struct	構造体メンバを共に一括化
-fshort-enums	列挙型によって必要とされるのと同数のバイトだけを割り当て
デバッグ任意選択	
デバッグレベル(引き落としメニュー) none, -g1, -g2, -g3	ソースコードで残されるまたは挿入されるコードとヘッダを追跡してデバッグするレベルを指定
警告メッセージ出力任意選択	
-Wall	全ての警告を表示
-Werror	疑わしい構成に対する警告の代わりに異常を生成
-pedantic	厳密なANSI Cによって要求された警告、禁止拡張を使う全てのプログラムを却下
-w	全ての警告メッセージを抑制
その他の任意選択	
他のフラグ(書式領域)	他のプロジェクト特有フラグを入力
-v	詳細

XC8リンカ任意選択

表8-3. XC8リンカ任意選択

任意選択	説明
全般任意選択	
<code>-ndefaultlibs</code>	標準Cライブラリをリンクしない。
<code>-Wl, -Map</code>	マップ ファイルを生成
<code>-Wl, -u, vfprintf</code>	<code>vfprintf</code> ライブラリを使用
ライブラリ任意選択	
ライブラリ -l (書式領域)	 の鉤を用いて、ここでライブラリ名の追加、優先付け、編集ができます。
ライブラリ検索パス -L (書式領域)	上と同じインターフェースでリンカが動的にリンクされるライブラリを探す所のパスの追加、優先付け、編集ができます。
最適化任意選択	
<code>-Wl, -gc-sections</code>	未使用領域ゴミ回収
その他の任意選択	
他のリンカフラグ (書式領域)	他のプロジェクト特有フラグを入力

XC8アセンブラ任意選択

表8-4. XC8アセンブラ任意選択

任意選択	説明
その他の任意選択	
他のアセンブラフラグ (書式領域)	他のプロジェクト特有フラグを入力

XC8纏め部任意選択

表8-5. XC8纏め部任意選択

任意選択	説明
全般任意選択	
纏め部フラグ (書式領域)	他のプロジェクト特有フラグを入力

8.3. XC8ツールチェーン任意選択

8.3.1. XC8共通任意選択

8.3.1.1. 全般

- **Target Device** (目的対象デバイス)
コア情報のようなデバイス特有任意選択(`-mcpu`)を示します。このフラグはコンパイル、リンク、アセンブラの段階中に渡されます。
- **Default Include Paths** (既定インクルードパス)
構築処理によって使われるコンパイラとDFPの全ての既定インクルードパスを一覧にします。
- **Relax Branches** (分岐緩和)
`-mrelax`
この任意選択はリンク時にコード生成部によって常により短く/より速い相対の呼び出しや分岐を出力される長い形式の呼び出しや分岐の命令の最適化を制御します。

8.3.1.2. 出力ファイル

- **Generate hex file (.hex)** (hexファイル生成)
この任意選択はelfファイルの成功裏の生成後のhexファイルの生成を制御します。
- **Generate eep file (.eep)** (eepファイル生成)
この任意選択はelfファイルの成功裏の生成後のeepファイルの生成を制御します。
- **Generate usersignatures file (.usersignatures)** (usersignaturesファイル生成)
この任意選択はelfファイルの成功裏の生成後のusersignaturesファイルの生成を制御します。
- **Generate lss file (.lss)** (lssファイル生成)
この任意選択はelfファイルの成功裏の生成後のlssファイルの生成を制御します。

8.3.1.3. その他

- **Other Common Flags** (その他の共通フラグ)

コンパイル、リンク、アセンブラ段階中に使用者がインクルード/追加を望むかもしれないその他のプロジェクト特有フラグを入力するため。

8.3.2. コンパイラ任意選択

8.3.2.1. 全般

- **-mcall-prologues** : 関数が入口でレジスタを保存する方法とそれらのレジスタが出口で復元される方法を変更。この任意選択が指定されない場合、各関数によって保存されることが必要なレジスタはそれらの関数内のコードによって保存と復元が行われます。**-mcall-prologues**任意選択が使われる場合、保存コードは関数の適切な場所と呼ばれるサブルーチンとして抜き出されます。この任意選択の使用はコードの大きさを減らしますが、コードの実行時間を増やし得ます。
- **-mno-interrupts** : スタックポインタが変更される時に割り込みが禁止されるべきかを制御。殆どのデバイスに対して、ステータスレジスタ(SREG)の状態は一時レジスタに保存され、スタックポインタが調整される前に割り込みが禁止されます。その後スタックポインタが変更されてしまった後でステータスレジスタが復元されます。プログラムが割り込みを使わないなら、スタック調整に対してこの方法で保護される必要はありません。この任意選択の使用はスタックポインタを調整するコード周りで割り込みを禁止して潜在的に再許可コードを省略し、従ってコードの大きさと実行時間を減らします。
- **-funsigned-char** : **unsigned**型を持つことを無修飾のcharオブジェクトに強制。既定で、**-mext=cci**任意選択が使われない限り、無修飾のchar型は**signed char**と等価で、この場合は**unsigned char**と等価です。**-funsigned-char**はこの型を**explicit**にします。コンパイラによって無修飾のchar型に割り当てられた型に頼るよりむしろそれらが定義された時にcharオブジェクトの符号属性を明示的に表明することを考慮してください。
- **-funsigned-bitfields** : これらの任意選択は宣言が**signed**または**unsigned**のどちらも使われない時にビット領域が**signed**または**unsigned**のどちらかを制御します。これらの任意選択は既定ビット領域が**unsigned**であることを示します。
- **-mext=cci** : コンパイル中に使われる言語拡張を制御します。CCI拡張の許可はコンパイラに対して全てのソースコードとコンパイラ任意選択を共通Cインターフェース(CCI:Common C Interface)で調べることをコンパイラに要求します。このインターフェースを持つコンパイラのコードは全てのMPLAB XCコンパイラに渡ってより簡単に移植することができます。CCIに準拠しないコードや任意選択はコンパイラ警告によって合図されます。

8.3.2.2. シンボル

- **-D**
 - **-D 名前** : 1の定義でマクロとして**名前**を事前定義します。
 - **-D 名前=値** : 値の定義でマクロとして**名前**を事前定義します。定義の内容は**#define**指令で第3段階への変換中にそれらが現れたかのように通票化されて処理されます。特に、定義は埋め込まれた新規行(改行)文字によって切り詰められます。
- **-U** : 組み込み、または**-D**任意選択で提供されたどちらかの以前のどの**名前**の定義も取り消します。

-Dと**-U**の任意選択はそれらがコメント行で与えられた順に処理されます。全ての**-iマクロファイル**と**-iインクルードファイル**の任意選択は全ての**-D**と**-U**の任意選択後に処理されます。

8.3.2.3. デイレトリ

- **-I デイレトリ** : ヘッダファイルに対して検索されるべきディレクトリの一覧にディレクトリを追加します。**-I**によって名付けられたディレクトリは標準システムインクルードディレクトリの前に検索されます。**ディレクトリ**が標準システムインクルードディレクトリの場合、システムディレクトリに対する既定検索順とシステムヘッダの特別な扱いが覆されないことを保証するため、この任意選択は無視されます。

8.3.2.4. 最適化

- コードを生成する時に使われる最適化のレベルを指定する一般的なスイッチの '**-O<最適化レベル>**' があります。
 - **-Os** : この任意選択は空指向最適化を要求します。この任意選択は代表的にコードの大きさを増さない支援される全ての最適化を要求します。コードの大きさを減らすように設計された更なる最適化も実行しますが、これは手続き型抽象化最適化のようにプログラム実行を遅くするかもしれません。このレベルは認可済みコンパイラに対してだけ利用可能です。
 - **-O0** : 初歩的な最適化だけ実行。これは**-O**任意選択が指定されない場合の既定最適化レベルです。選んだ最適化レベルで、コンパイラの目的はコンパイルの負担を減らしてデバッグで期待される結果を生じさせることです。文はこの最適化レベルでコンパイルする時に無関係です。文間の中断点(ブレークポイント)でプログラムを停止する場合、その後新しい値を何れかの変数に割り当てる、または関数内の他のどれかの文にプログラムカウンタを変更してソースコードから意図する結果を正確に得ることができます。コンパイラはレジスタで宣言された変数だけを割り当てます。
 - **-O1**または**-O** : コードの大きさと実行時間を減らしますが、未だ妥当なデバッグ能力を許します。このレベルは認可済みだけでなく未認可のコンパイラに対しても利用可能です。

- **-O2** : このレベルに於いてコンパイラは空と速度の二律背反を伴わないほぼ全ての支援される最適化を実行します。このレベルは認可済みだけでなく未認可のコンパイラに対しても利用可能です。
- **-O3** : この任意選択は実行時間を減らす全ての支援される最適化を要求しますが、これはプログラムの大きさを増すかもしれません。このレベルは認可済みコンパイラに対してだけ利用可能です。
- **-Og** : この任意選択はデバッグと厳密にインターフェースする最適化を禁止し、高速コンパイルと良好なデバッグ体験を維持すると同時に妥当な最適化のレベルを提供します。
- 他の最適化任意選択
 - **--function-sections**と**-fdata-sections** : 目的対象が任意の区分(section)を支援する場合に出力ファイルで各関数またはデータの項目をその区分に配置。関数の名前またはデータ項目の名前は出力ファイルでの区分の名前を決めます。
 そのように行うことからかなりの恩恵がある時にだけこれらの任意選択を使ってください。これらの任意選択指定時、アセンブラとリンカはより大きなオブジェクトと実行可能のファイルを作成し、またより遅くなるでしょう。
 - **-fshort-enums** : 宣言された範囲の可能な値に必要なバイト数だけの列挙型を割り当て。具体的に、列挙型は最小の整数型と等価で、これは十分な場所を持ちます。

8.3.2.5. デバッグ

- **-g** レベル (デバッグ レベル)
 - **-g1** : デバッグの計画をしないプログラムの部分で逆追跡をするのに十分な最小の情報を生成します。これは関数と外部変数の記述を含みますが、局所変数についての情報と行番号がありません。
 - **-g2** : これは既定デバッグ レベルです。
 - **-g3** : これはプログラムに存在する全てのマクロ定義のような追加の情報を含みます。いくつかのデバッグは**-g3**使用時にマクロ展開を支援します。

8.3.2.6. 警告

- **-Wall** : 例えマクロと組み合わせても、或る使用者が疑問があると考え易く回避可能な構造についての全ての警告を許可。
- **-Werror** : 警告を異常として表示。
- **-pedantic** : プログラムが禁止された拡張を使わないことと、プログラムがISO Cに従わない時に警告が発行されることを保証。
- **-w** : 全ての警告メッセージを禁止。

8.3.2.7. その他

- **他のコンパイラ フラグ** : コンパイル段階で使用者がインクルード/追加を望むかもしれない他のプロジェクト特有コンパイラ フラグの入力のため。
- **-v** : 冗長なコンパイルを指定。この任意選択使用時、内部コンパイラ応用の名前とパスはそれらが実行される時に表示され、各応用が渡すコマンド行引数が後続します。あなたのドライバ任意選択が期待するように処理されることを確かめる、または内部応用がどの警告または異常を発行したかを見るためにこの任意選択を使うかもしれません。

8.3.3. リンカ任意選択

8.3.3.1. 全般

- **-ndefaultlibs** : この任意選択は標準システム ライブラリがリンク時に使われるのを防ぎます。指定したライブラリだけがリンカに渡されます。
- **-Wl,-Map** : 配置(**map**)ファイルを生成。
- **-Wl,-u,vfprintf** : **vprint**ライブラリを使用。

8.3.3.2. ライブラリ

- **-lライブラリ** : **-lライブラリ**任意選択はリンク時に未解決シンボルのために**ライブラリ**と名付けられたライブラリを走査します。リンカは**library.a**の名前を持つライブラリに対してディレクトリの標準一覧を検索します。コマンドでこの任意選択を書く場所で違いを生じます。リンカはそれらが指定された順にライブラリとオブジェクトのファイル処理します。従って、**foo.o -lz bar.o**はファイルの**foo.o**の後、**bar.o**の前にライブラリzを検索します。
 検索されるディレクトリはいくつかの標準システム ディレクトリとそれに加えて**-L**で指定するどれをも含みます。
 通常、この方法で見つかるファイルはライブラリ ファイル(メンバーがオブジェクト ファイルである纏め(アーカイブ)ファイル)です。リンカはメンバーに対してそれを通して走査することによって纏めファイル処理し、これは参照されたけれども未だ定義されていないシンボルを定義します。しかし、その見つかったファイルが普通のオブジェクト ファイルの場合、通常のようにリンクされます。**-l**任意選択(例えば、**-lmylib**)使用とファイル名(例えば、**mylib.a**)使用間の違いは**-L**任意選択によって指定された時にコンパイラがいくつかのディレクトリで**-l**を使って指定したライブラリを検索することだけです。
- **Ldir** : **-l**に対して検索すべきディレクトリの一覧に**dir**ディレクトリを追加します。

8.3.3.3. 最適化

- `-Wl, --gc-sections` : 未使用領域をゴミ回収(ガベージコレクション)。
未使用入力領域のゴミ回収を許可。この任意選択を支援しない目的対象では無視されます。
(このゴミ回収を実行しない)既定の動きはコマンド行で '`--no-gc-sections`' を指定することによって元へ戻すことができます。 '`--gc-sections`' はどの入力区分がシンボルと再配置の調査によって使われるかを決めます。入り口シンボルを含んでいる区分とコマンド行で定義されないシンボルを含んでいる全ての区分は、動的オブジェクトによって参照されるシンボルを含んでいる区分として維持されます。

8.3.3.4. その他

- **その他のリンカフラグ** : リンカ段階で使用者がインクルード/追加を望むかもしれないその他のプロジェクト特有リンカフラグを入力するため。

8.3.4. アセンブラ任意選択

8.3.4.1. 全般

- 既定アセンブラフラグ
 - `-Wa, -x assembler-with-cpp` : `-Wa`, 任意選択はアセンブラに直接任意選択引数を渡します。 `-Wa, -x assembler-with-cpp` 言語任意選択はアセンブリソースファイルがアセンブルされる前に前処理されることを保証し、従って `#include` とアセンブリコードを持つC形式注釈のような前処理部疑似命令の使用を許します。既定で、`.S` または `.sx` 拡張子を使わないアセンブリファイルは前処理されません。
 - `-c` : `-c` 任意選択はアセンブラ実行後にコンパイルを停止するのに使われ、出力として、`.o` 拡張子を持つ再配置可能な中間オブジェクトファイルのままにします。

8.3.4.2. デバッグ

ソースコードに残されたまたは挿入されたコードとヘッダの追跡とデバッグの水準を指定。

- 既定デバッグフラグ
 - `-Wa, -g` : コードの追跡とデバッグの水準を指定。

8.3.4.3. その他

- **その他のアセンブラフラグ** : アセンブラ段階で使用者がインクルード/追加を望むかもしれないその他のプロジェクト特有アセンブラフラグを入力するため。

8.3.5. 纏め部任意選択

8.3.5.1. 全般

- `-r` : 既存または挿入した新しいファイルを纏め部に再配置。
- `--target` : 目的対象デバイスを指定。

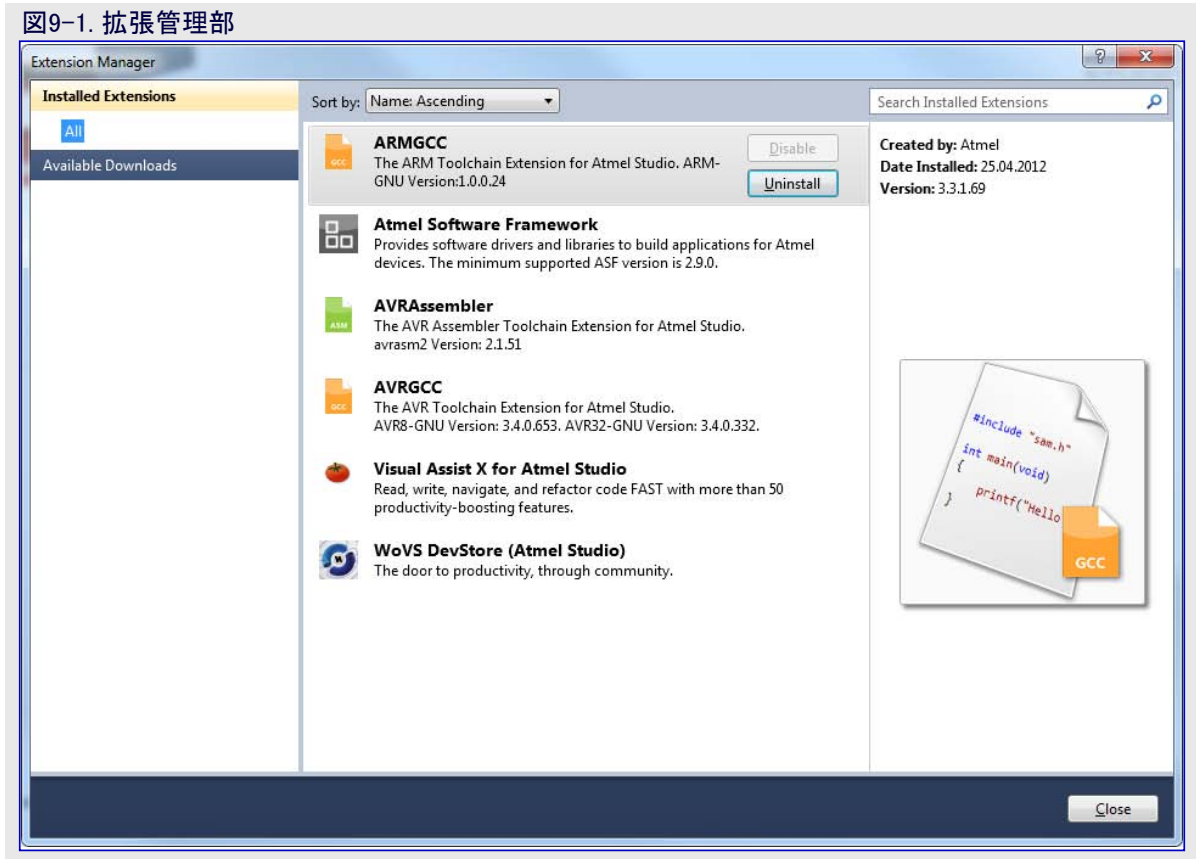
9. Microchip Studioの拡張

Microchip StudioはMicrochip Studio拡張を追加、削除、許可、禁止を許す拡張管理部(Extension Manager)と名付けられたツールを含みます。拡張管理部を開くにはTools(ツール)メニューでExtension Maneger(拡張管理部)をクリックしてください。

拡張開発者は進行中に拡張の以前の版をアンインストールし、開発中の衝突を防ぐために潜在的に衝突する拡張をアンインストールまたは禁止することが勧告されます。

9.1. 拡張管理部UI

Extension Manager(拡張管理部)ウィンドウは3つの枠に分けられます。左枠はオンライン陳列室からのインストールした拡張と新しい拡張を群によって選ぶことを許します。



拡張は中央枠に表示されます。一覧の上のコンボ枠から名前または著者によって一覧を並び替えることができます。

中央枠で拡張を選択すると、それについての情報が右枠に現れます。現在、使用者によってインストールされた拡張はアンインストールまたは禁止することができます。Microchip Studioと共に配給された拡張は変更することができません。

Extension Maneger(拡張管理部)ウィンドウは検索枠を含みます。左枠での選択に応じて、インストールした拡張、オンライン陳列室、利用可能な更新を検索することができます。

オンライン陳列室拡張管理部はMicrochip Studio陳列室から拡張をインストールすることができます。これらの拡張は一括、雛形、またはMicrochip Studioに機能を追加する他の部分品かもしれません。

拡張管理部で始めるには「[9.3. Microchip Studioでの新規拡張インストール](#)」を調べてください。

拡張形式

拡張管理部はプロジェクト雛形、項目雛形、工具箱項目、管理された拡張枠組み(MEF:Managed Extension Framework)、VS一括を含むかもしれないVSIX一括形式での拡張を支援します。拡張管理部はMSIに基づく拡張をダウンロードしてインストールすることもできますが、それらを許可または禁止することができません。Microchip Studio陳列室はVSIXとMSIの両拡張を含みます。

依存関係の扱い

依存性を持つ拡張をインストールしようとする場合、インストーラはそれらの依存物が既にインストールされていることを確認します。それらがインストールされていない場合、拡張管理部は拡張がインストールされ得るのに先立ってインストールされなければならない依存物の一覧を示します。

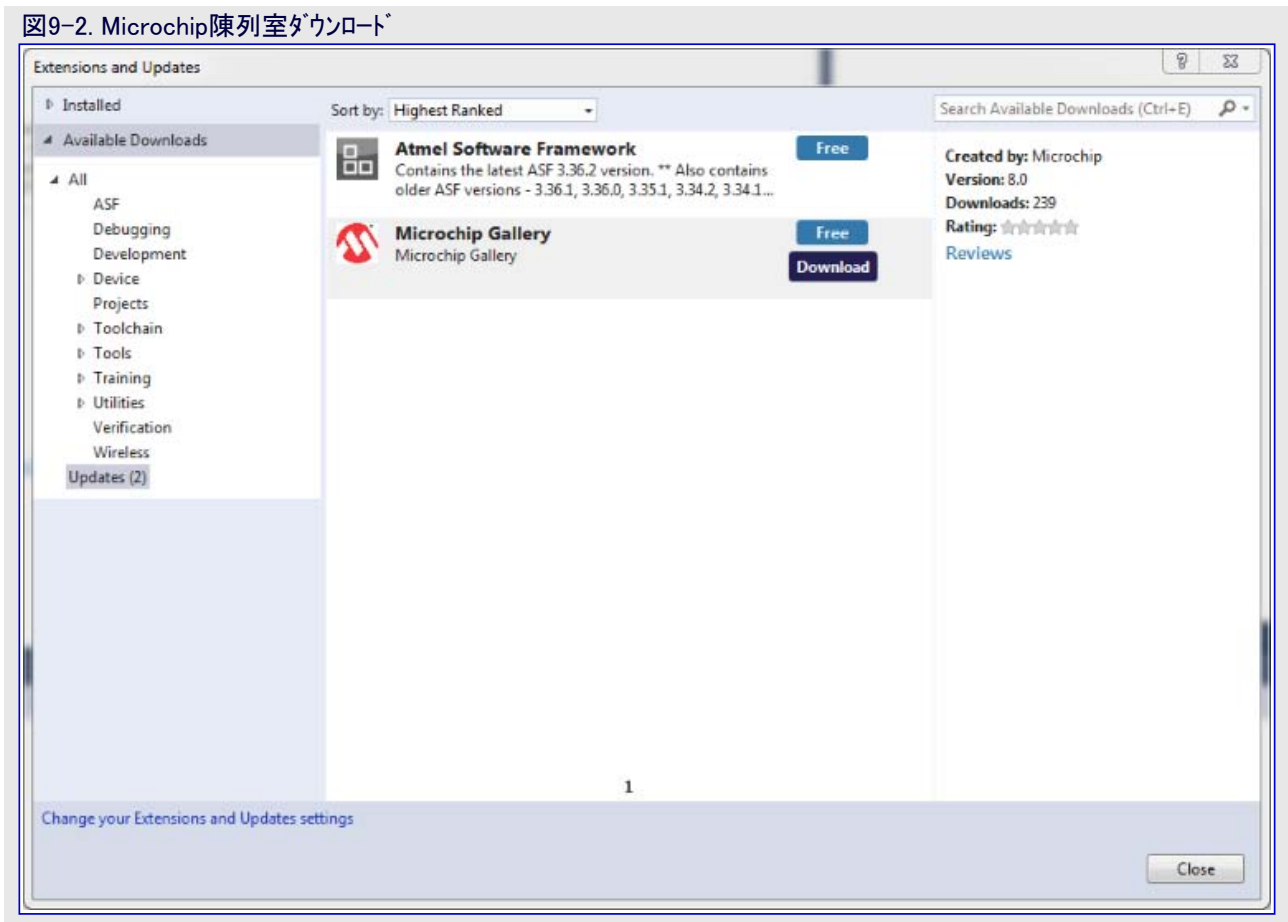
拡張管理部を使わないインストール

.vsixファイルで一括にされた拡張はMicrochip Studio展示室以外の場所で入手可能かもしれません。拡張管理部はそれらのファイルを検出することができません。けれども、それをダブル クリックしてその後に構成設定指示に従うことによって、.vsixファイルをインストールすることができます。拡張がインストールされると、それを許可、禁止、削除するために拡張管理部を使うことができます。

9.2. Microchip拡張陳列室での登録

拡張をダウンロードするためにはMicrochip拡張陳列室での登録が必要とされます。

図9-2. Microchip陳列室ダウンロード



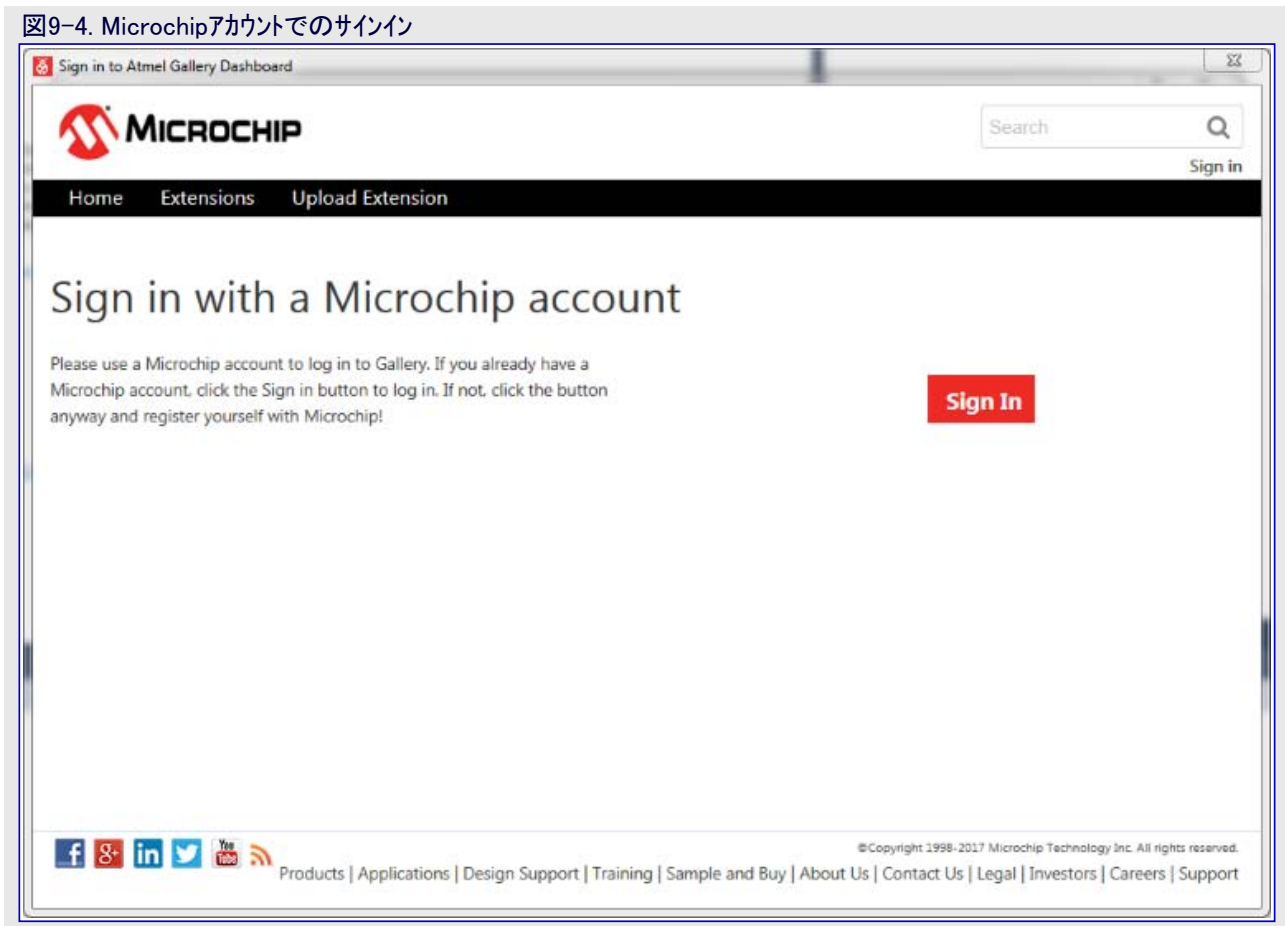
一度、ダウンロードをクリックすると、Microchip陳列室のサインイン頁に連れて行かれます。

ダイアログの右上でSign in(サインイン)をクリックすると、Microchipのアカウントにサインインするように問われます。

図9-3. Microchip陳列室に対するサインイン頁

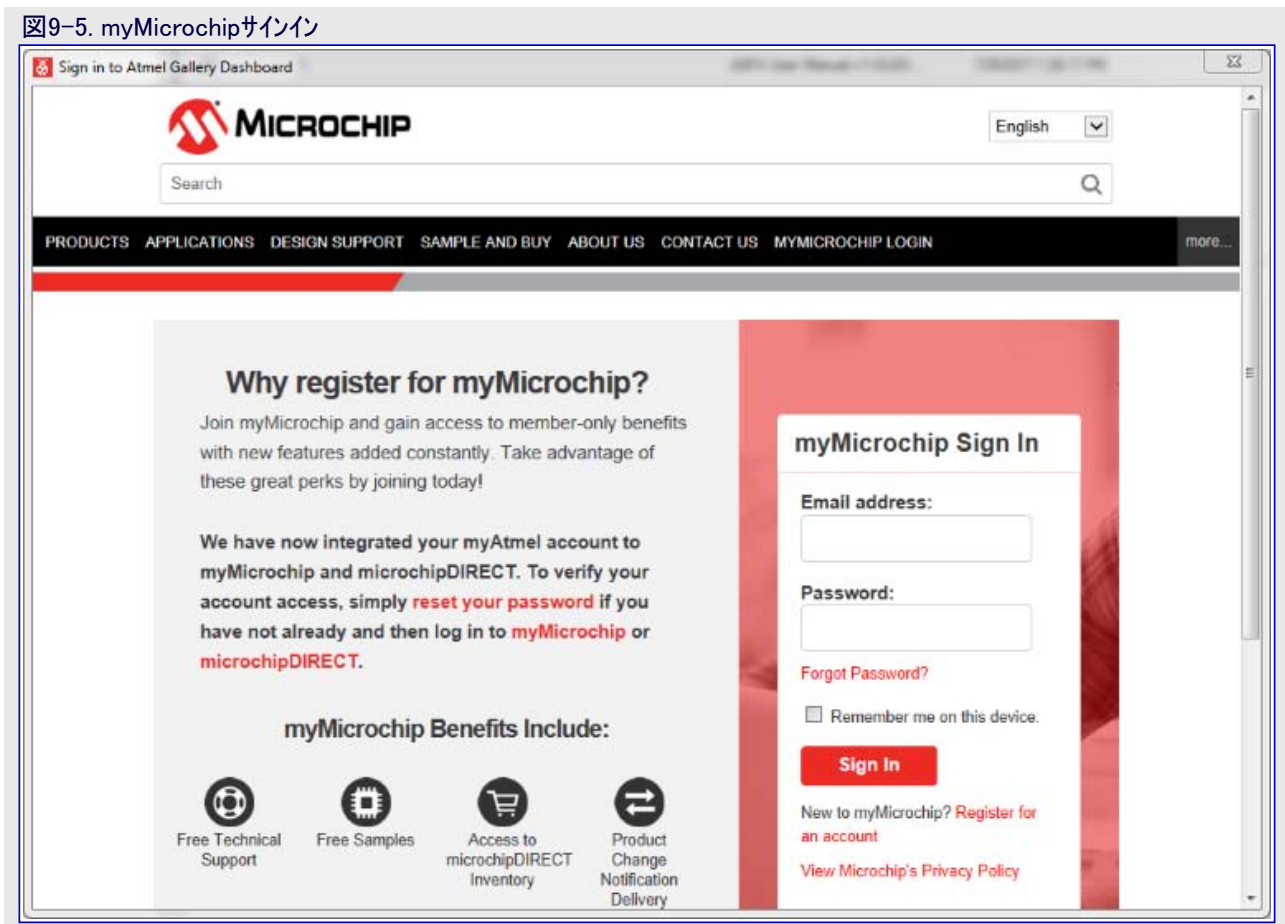


図9-4. Microchipアカウントでのサインイン



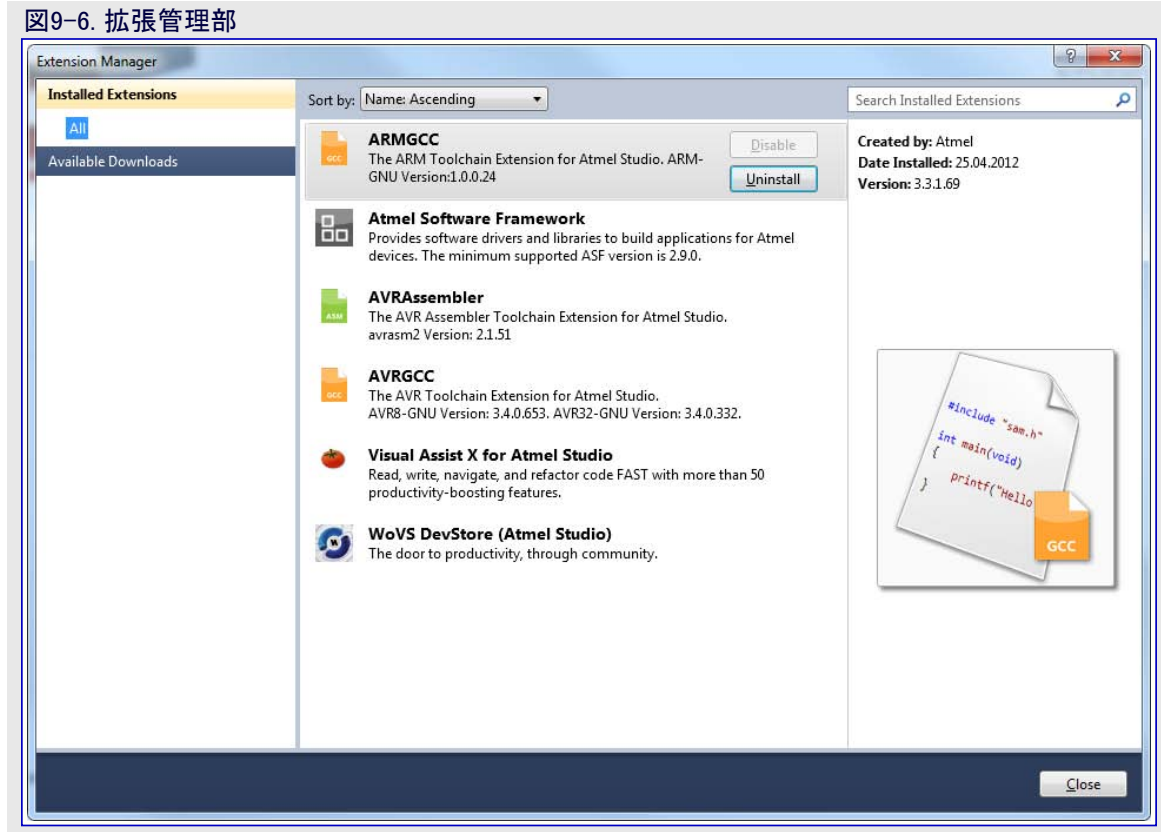
Sign in(サインイン)でのクリックがあなたをmyMicrochipサインイン頁へ連れて行きます。

図9-5. myMicrochipサインイン



9.3. Microchip Studioで新しい拡張のインストール

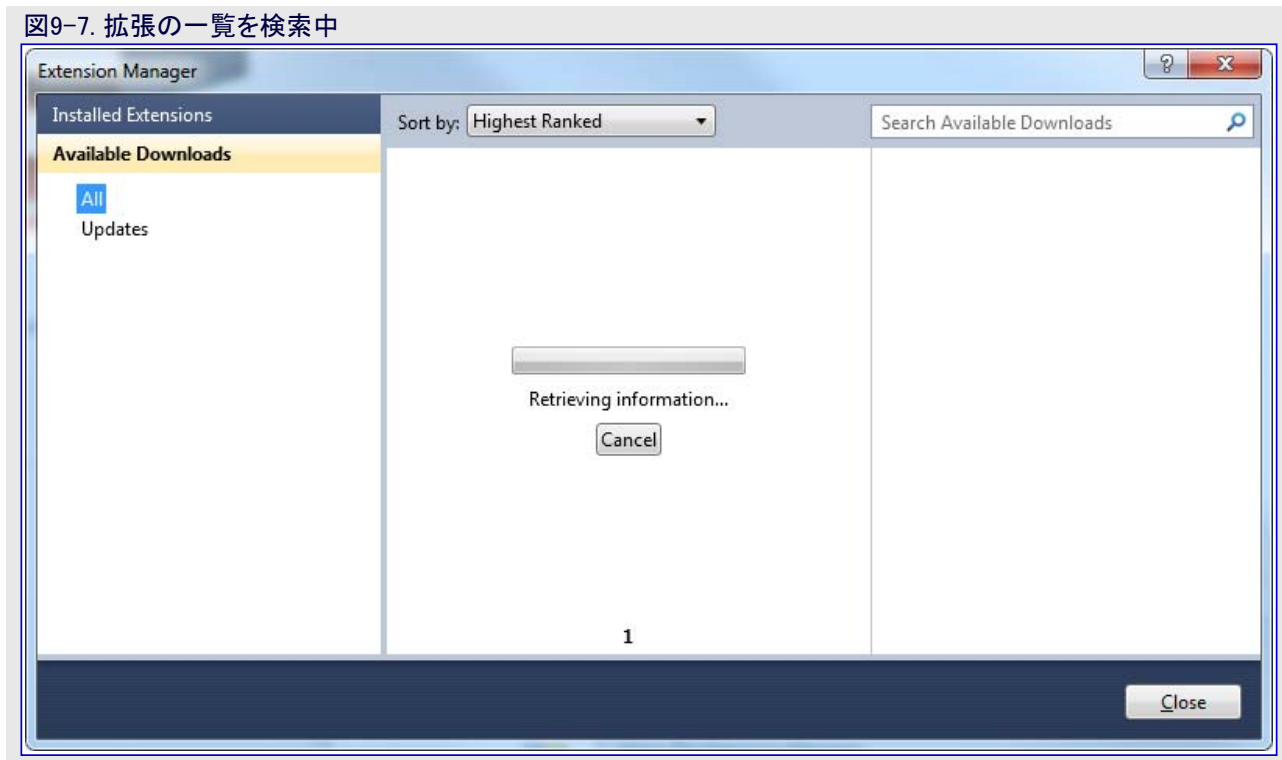
段階1



Extension Manager (拡張管理部)ウィンドウを開くことはインストールされた拡張を示します。

新しい拡張を見つけてインストールするには、左枠でAvailable Downloads(利用可能なダウンロード)タブをクリックしてください。

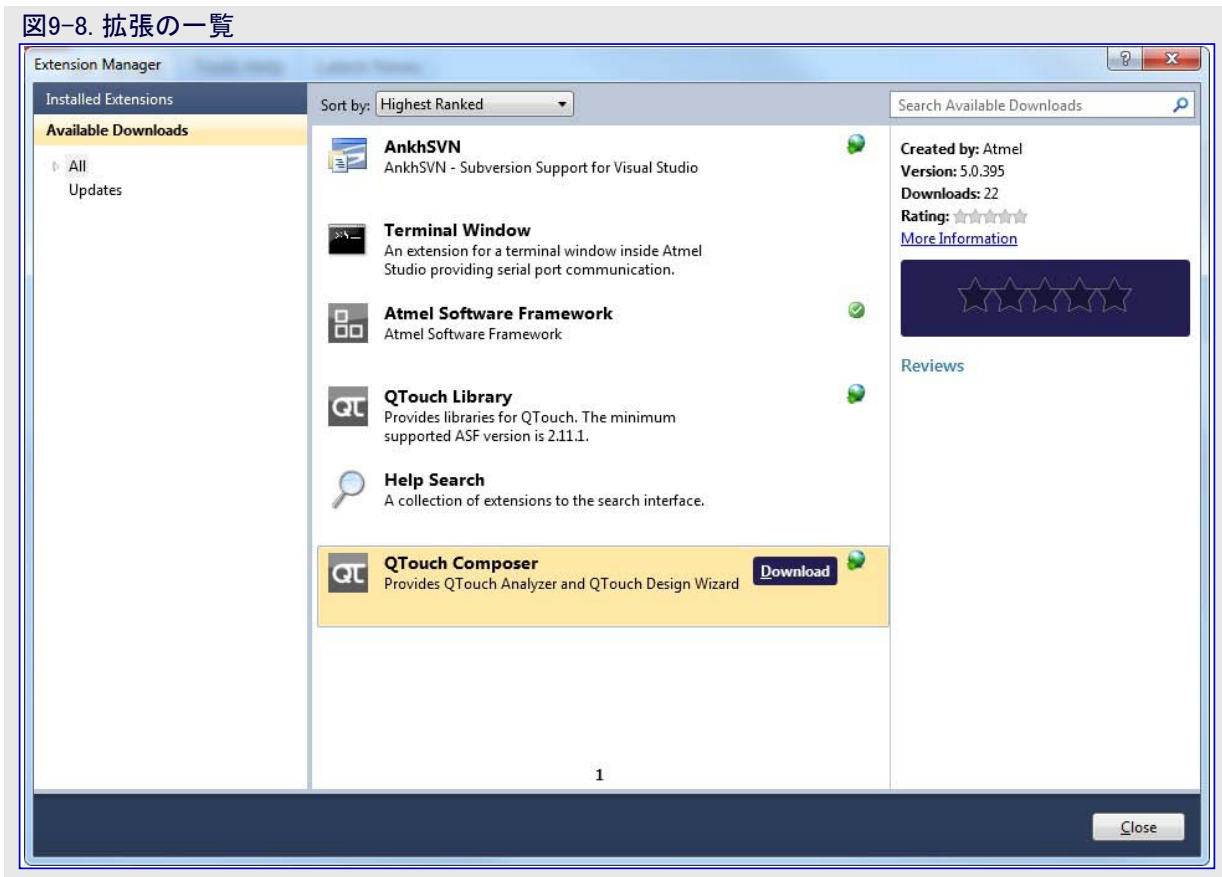
段階2



利用可能な拡張一覧検索には暫くかかります。

段階3

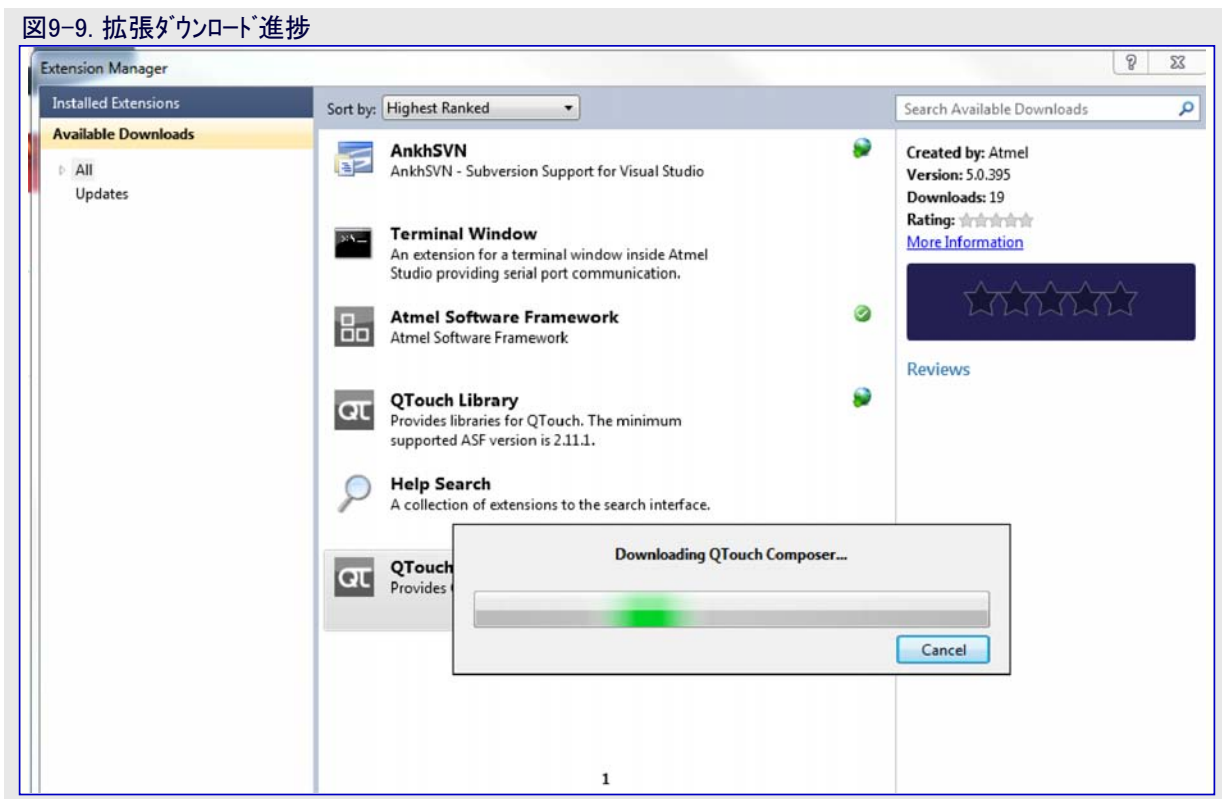
図9-8. 拡張の一覧



緑のチェック印は既にインストールされた拡張を示します。

QTouch Composer(QTouch構成器)を選択してDownload(ダウンロード)を押してください。以前に拡張陳列室での使用者として登録されていない場合、この時点で「9.2. Microchip拡張陳列室での登録」に連れていかれます。

図9-9. 拡張ダウンロード進捗



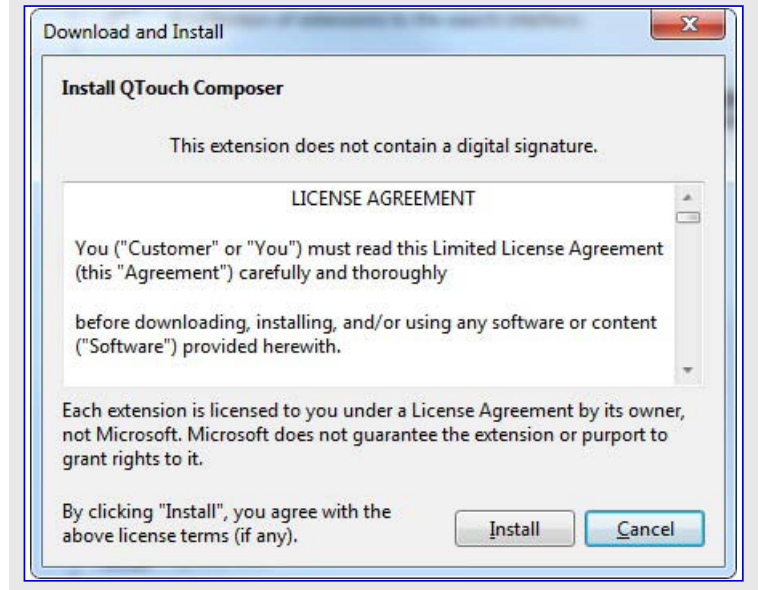
Microchip Studioの状態バーで示されるようにダウンロードが始まります。拡張が独立型インストーラとして配給される場合、そのファイルを保存する場所を問われるでしょう。ダウンロードは大きなファイルに対して数分かかり得ます。ダウンロード中に進捗バーを持つダイアログが表示されます。ダウンロードが大きな拡張に対して長くかかり得ることに注意してください。ダウンロードを中断するにはCancel(取り消し)を押してください。

段階4

新しい拡張をインストールする時の殆どに於いて読むために許諾契約が現れるでしょう。

それを慎重に読み、例えば、拡張の安全性が破られた場合、相互に互換性のないインストールからの結果としての起こり得る誤動作と二次的な損害に於いて拡張の著作者の殆どが責任を取らないので、必要な拡張だけをインストールしてください。

図9-10. 拡張許諾



段階5

一旦拡張がダウンロードされると、下側の状態バーにメッセージが現れます。

図9-11. 拡張管理部再始動警告

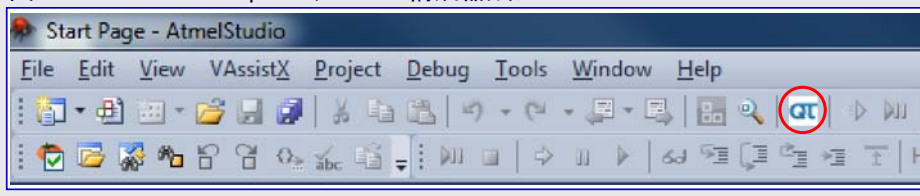


直ちにIDEを再始動するにはRestart Now(直ぐ再始動)鈕をクリックしてください。別に後で再始動する予定ならClose(閉じる)鈕をクリックしてください。

未保存プロジェクトがある場合、再始動に前に行った変更を保存するように要求されます。

段階6

図9-12. QTouch Composer(QTouch構成器)鈕



Microchip Studio再始動後、QTouch Composer(QTouch構成器)に対して新しい鈕が追加されます。

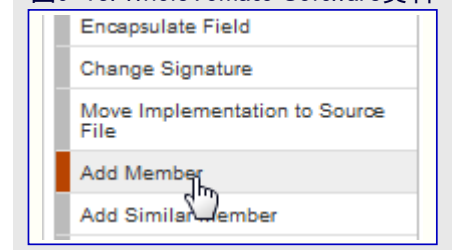
9.4. Visual Assist

Microchip Studioは予めインストールされた拡張のWholeTomato SoftwareからのVisual Assistと共にやって来ます。

Visual Assistの資料は様々な供給元から入手可能です。

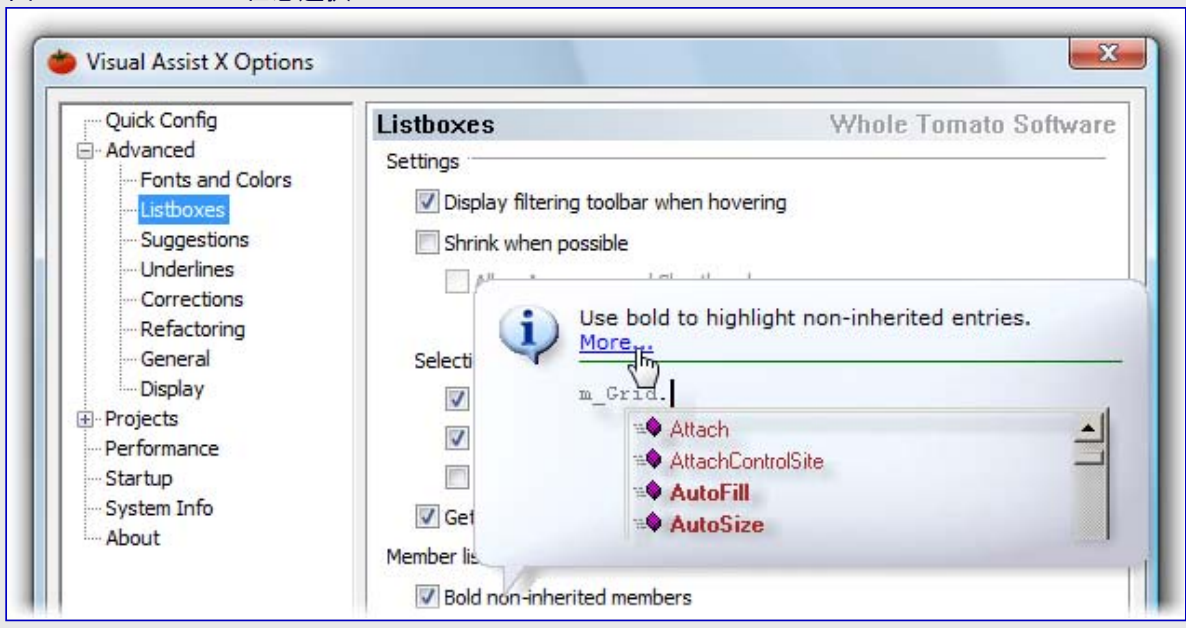
- www.wholetomato.comへ行ってください。機能によって資料を検索するには左手カーソルのメニューをクリックしてください。

図9-13. WholeTomato Software資料

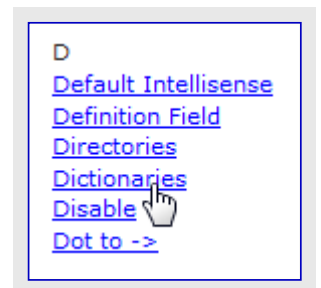


- Visual Assist任意選択ダイアログ内のハイパーリンクを用いて関連資料へ直接飛んでください。

図9-14. Visual Assist任意選択



- 用語集で用語をクリックしてください。



9.5. Percepio Tracealyzer

Percepio Tracealyzerについてより多くの情報を得るにはpercepio.com/atmel/をご覧ください。

9.6. QTouch®構成器とライブラリの概要

QTouch Composer(構成器)とライブラリは応用に対して容易で継ぎ目のない容量性接触機能の開発を許し、これはMicrochip Studioでコードを編集して接触設計を調整するのに必要とされるツールと結び付けることによって設計手順を簡単化します。以前はQTouch Studioと呼ばれたQTouch構成器は拡張としてMicrochip Studio 6に完全に統合されます。QTouchライブラリはMicrochipStudioに対するソフトウェア枠組み拡張で、様々なMicrochipデバイスでの接触機能追加を許します。

9.6.1. インストール

1. Microchip Studioを開始してください。
2. Tools(ツール)⇒Extension Manager(拡張管理部)⇒Online Gallery(オンライン陳列室)へ行ってください。
3. QTouch Library(QTouchライブラリ)を選択して”Download”をクリックし、その後にそれをインストールしてください。
4. QTouch Composer(QTouch構成部)を選択して”Download”をクリックし、その後にそれをインストールしてください。
5. Extension Manager(拡張管理部)ウィンドウで”Rstart Now(直ぐ再始動)” 卸をクリックしてください。
6. Microchip Studio再始動後、Tools(ツール)⇒Extension Manager(拡張管理部)に行ってください。QTouch Library(QTouchライブラリ)とQTouch Composer(QTouch構成部)が一覧にされて状態が許可されていることを調べてください。

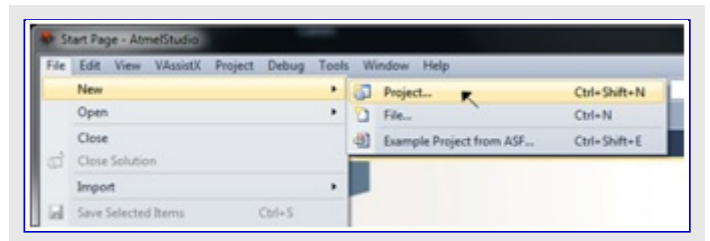
9.6.2. QTouch®プロジェクト構築器の概要

YouTubeでのMicrochip Studioに於けるQTouch®構成器での開始に際して



QTouch Project構築器はデバイスと接触感知部の選択から完全な接触プロジェクトを自動的に生成するまでの全ての段階を通して案内します。

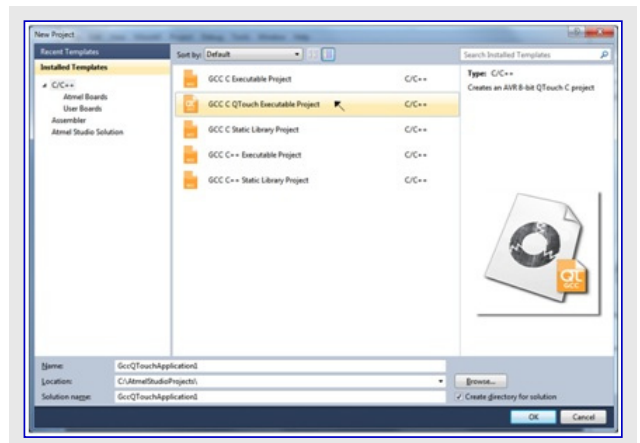
1. Microchip Studioを開始してください。
2. **Files**(ファイル)メニューを開いて**New**(新規)⇒**Project**(プロジェクト)でクリックしてください。



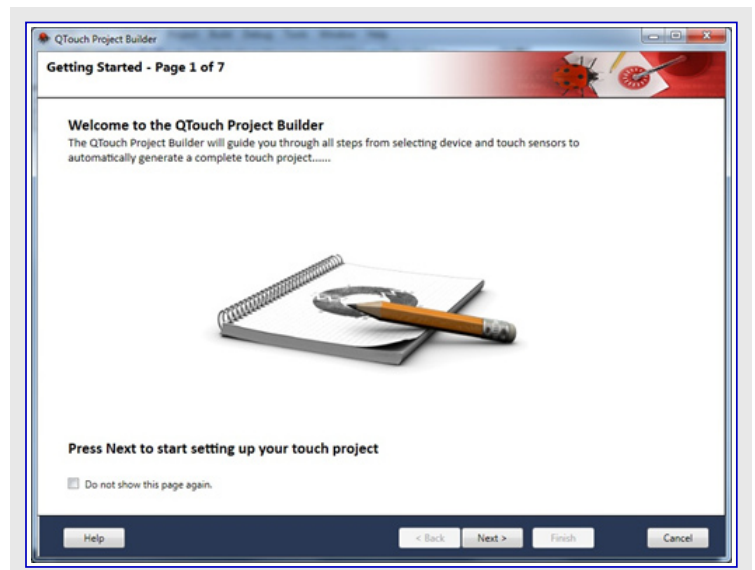
3. **New Project**(新しいプロジェクト)ダイアログが開かれます。New Projectダイアログで”GCC C QTouch Executable Project(GCC C QTouch実行可能プロジェクト)”を選択してください。

New Projectダイアログで以下の詳細を入力してOK鈕をクリックしてください。

- プロジェクトの名前
- プロジェクトと解決策の場所
- 解決策の名前



4. プロジェクト作成で必要とされる段階を通して案内する“QTouch Project Builder(QTouchプロジェクト構築部)”ダイアログが開かれます。



9.6.3. QTouch®分析器の概要

QTouch Analyzer(分析器)はQTouchキットから各種表示部に送られる接触データを読んで解釈します。この分析器はKit View(キット表示部)、Kit/Sensor Properties(キット/感知部特性)、Sensor Data(感知部データ)、Trace View(追跡表示部)、Power View(電力表示部)、Graph View(グラフ表示部)に分けられます。接触キットが接続されてMicrochip Studioが開かれると、QTouch Analyzer(QTouch分析器)ウィンドウが開いて接続情報が更新されます。

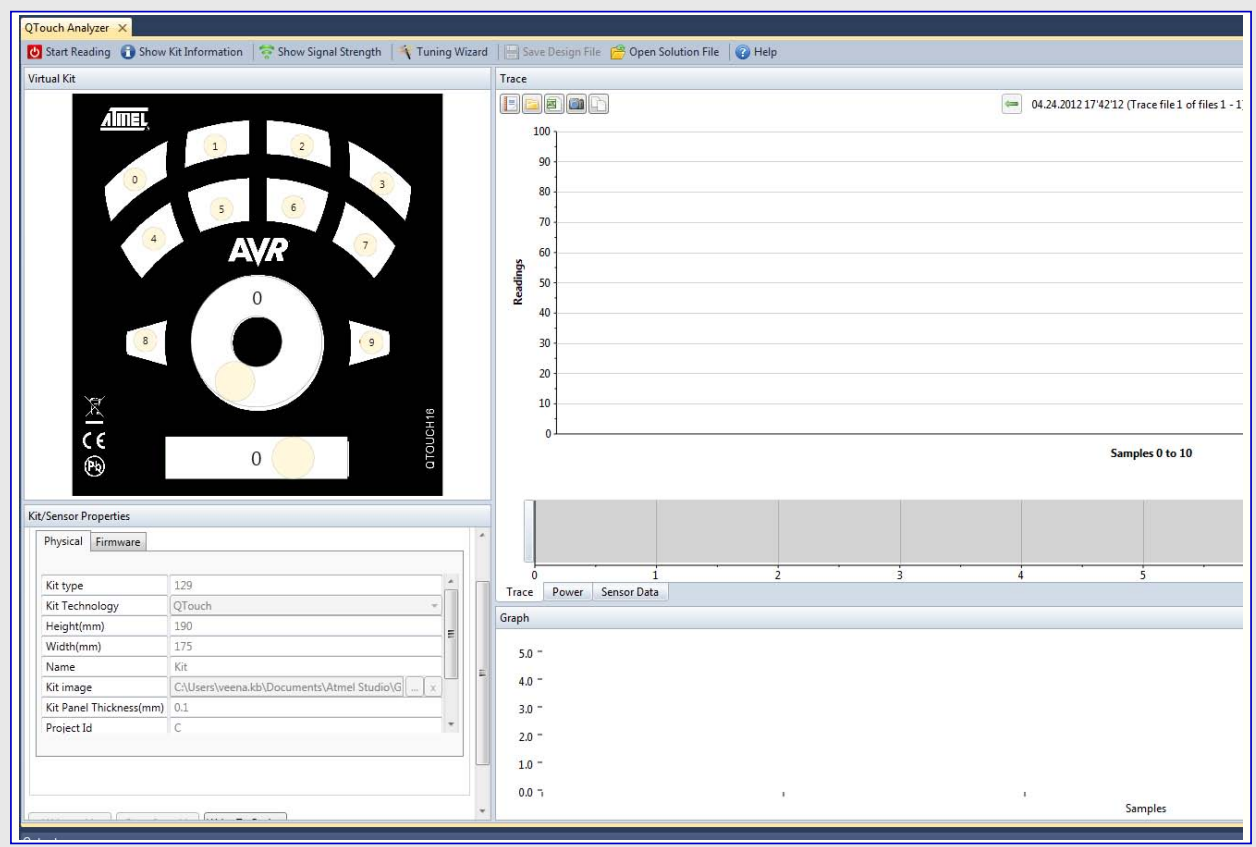
Virtual Kit(仮想キット)表示部は釦押下、輪と摺動子の使用のような接触事象を示します。この画像は接続した接触キットから読まれたデータに基づいて更新されます。

Kit/Sensor Data表示部はキット/感知部の構成設定任意選択を見て編集することを許します。

Sensor Data表示部は現在接続したキットの接触データ情報を提供します。

Graph表示部は1つまたはより多くの選んだ接触データをグラフで表示します。グラフは最も最近の接触データを表示します。見るデータ群は表示部の右側のデータ群一覧から選ぶことができます。データ群はSignals(信号)、Deltas(差)、References(基準)、Wheel/Slider positions(輪/摺動子位置)を表すタブ付けされたページで表示されます。各データ群選択は、その項目を選択するのに一覧内でその項目をクリックする通常の見出し規定に従います。連続する項目の範囲を選択するには最初に先頭の項目をクリックしてその後にShiftキーを押し続けて範囲の最後の項目を選択してください。複数項目は一覧内で次の項目を選択するのに先立ってCtrlキーを押し続けることによって一度に1つずつを選択することもできます。最後の場合(方法)は項目が連続する範囲である必要がありません。Ctrl選択法の使用は複数項目の選択から個別項目の選択解除も許します。

追跡はグラフ内の接触データを持つ1つ以上の一連のデータを含みます。追跡は(押下で読み込みを開始と停止をする)1つの信号読み込み作業の全ての履歴データを保持し、データは独立したファイルに保存され、後で再び開くことができます。



9.7. スクリプト手書き拡張

Microchip StudioはIDEによって自動的に実行することができるいくつかの引っかけスクリプトを提供します。これらの拡張は主にPythonで書かれ、例えば、中断点が当たった時、式が評価された時、またはプログラムがリセットされつつある時に実行します。

9.7.1. スクリプトのデバッグ

スクリプト デバッグ インターフェイスはPython®ファイルで定義されるべく名付けられた関数に基づき且つかかり依存する機能です。その後この機能は対応する事象がMicrochip Studioの内側で起きている時に呼ばれます。

注意: 通常作業中の初期化で使われる時間が低く保たれるように、異常検査はPython環境内に送出される関数に対して最小に保たれます。これはこのインターフェイスを通してMicrochip Studioを破壊するものが色々あることを意味します。

Pythonファイルを設定するには、プロジェクトのDebugフォルダ内、EFLアフィルの傍、またはプロジェクトファイルがある場所の1つ上のフォルダに、debughooks.pyと名付けられたファイルを配置してください。全てのプロジェクトに対してスクリプトを設定するためにMicrochip Studioのインストール ディレクトリの内側にこのファイルを配置することも可能です。

注: Pythonファイルはプロジェクトが開始される時に読み込まれてコンパイルされ、故にデバッグ作業中のPythonファイル変更は次のデバッグ作業が開始されるまで有効ではありません。Pythonファイルは.NETとPython 2.7ランタイムへの完全なアクセスを持つIronPythonコンテキストで走行します。ランタイムのより多くの情報についてはironpython.net/documentation/dotnet/をご覧ください。

Microchip Studioが読み込みを試みる関数はそれらの関数識票と共に下で示されます。

```
def should_process_breakpoint(studio_interface, breakpoint_address, breakpoint_id, obj):
    """
    中断点がMicrochip Studioをデバッグ動作へ移行させるべきかを定めるために呼ばれます。
    この関数がFalseを返すなら、Microchip Studioは中断点で中断しません。
    """
    return True

def has_processed_breakpoint(studio_interface, breakpoint_address, breakpoint_id, obj):
    """
    この関数はMicrochip Studioが中断点で中断している場合に呼ばれます。
    GUIは現在停止動作です。
    """
    pass

def on_reset(studio_interface, reset_address):
    """
    この関数は目的対象がリセットされる時に呼ばれます。
    リセットが行く場所のアドレスは'reset_address'です。
    """
    pass

def on_eval_expr(studio_interface, expression):
    """
    この関数はAtmel Sudioで評価される各式に対して呼ばれます。

    これは目的対象からのデータを示す監視ウィンドウと他のウィンドウを含みます。
    それを評価するために'式'文字列を渡すか、または式を上書きするために評価されるべき別の式を返します。
    この上書きはMicrochip StudioのGUIでは見ることができません。
    """
    return expression
```

注: Microchip Studioはスクリプトが見つかって正しく読み込まれた場合にそれらの全ての関数が利用可能だと期待します。例えば`should_process_breakpoint`が未定義の場合、未定義関数の戻り値はそれ自身が未定義のため、中断点は無作法なふるまいを始めるかもしれません。

上で示されるコードに於いて、Microchip Studio内に戻る主インターフェースは`studio_interface`オブジェクトです。このオブジェクトはメッセージを示して目的対象相互作用を行うためのいくつかの関数を含みます。

`studio_interface`オブジェクト内の`Print`関数はMicrochip Studio内側の出力ウィンドウでテキストを示すのに使われます。この関数は出力するための文字列と出力ウィンドウ内のタブの名前の2つの引数を取ります。下の例は“Expressions(式)”タブに評価した全ての式を出力します。

```
def on_eval_expr(studio_interface, expression):
    studio_interface.Print("Evaluating {}".format(expression), "Expressions")
    return expression
```

注: `Print`を通して送られるテキストの重要度は`INFO`に設定され、これはその出力がMicrochip Studioによって遮蔽されるかもしれないことを意味します。閾値をより低くするには`Tools(ツール)⇒Tools(ツール)`へ行って、`Status Management(状態管理)`を選んで、`Display Threshold(表示閾値)`を`INFO(情報)`に設定してください。

`studio_interface`オブジェクト内の`ExecStmt`関数はデバッグで文を実行するのに使われます。これは例えば変数を設定するのに使うことができます。より多くの情報については[MSDNデバッグExecuteStatement Method\(文実行法\)](http://MSDNデバッグExecuteStatement Method(文実行法))をご覧ください。

`WriteMemory`と`ReadMemory`は目的対象上のメモリの読み込みと書き込みのための対称的な関数です。スクリプトとMicrochip Studio間でデータを渡すのに`System.Array[System.Byte]`オブジェクトを使うことが重要です。

```
import System

def should_process_breakpoint(studio_interface, breakpoint_address, breakpoint_id, obj):
    vals = System.Array[System.Byte]([1, 2, 3, 4, 5, 6, 7, 8, 9])

    studio_interface.WriteMemory(data=vals, adr=0, type="eeprom")

    ret = studio_interface.ReadMemory(adr=0, type="eeprom", count=9)

    studio_interface.Print("ret == vals => {!r}".format(ret == vals), "Python")
    return True
```

`CalcNumericValue`は`CalcValue`呼び出しに対する略記形式です。これはシンボルの数値を、または関数がシンボルの値取得に失敗した場合には提供された既定値を返します。

```
def should_process_breakpoint(studio_interface, breakpoint_address, breakpoint_id, obj):
    a = studio_interface.CalcNumericValue("a", 0)
    if a == 0:
        studio_interface.Print("a was 0 or default", "Value scripts")
    else:
        studio_interface.Print("a = {}".format(a), "Value scripts")
    return True
```

`CalcValue`関数は目的対象のコードが走行する場所の範囲でシンボルについての情報を取得するのに使われます。この呼び出しの戻り値はシンボルのアドレス、シンボル情報、値を含む情報の一覧です。この一覧で送られたオブジェクトはシンボルについて既知の全ての情報を含みますが、殆どの有用な領域は評価されたシンボルの値を含む最後の要素です。

```
def should_process_breakpoint(studio_interface, breakpoint_address, breakpoint_id, obj):
    a = studio_interface.CalcValue("a")
    # aは現在変数aについての全ての情報を含みます。
    # 以下のメンバを持つ一覧です。
    # a = [
    #     <Atmel.VsIde.AvrStudio.Services.TargetService.TCF.Services.ValueInfo>,
    #     <Atmel.VsIde.AvrStudio.Services.TargetService.TCF.Services.SymbolInfo>,
    #     <Atmel.VsIde.AvrStudio.Services.TargetService.TCF.Services.ExpressionInfo>,
    #     '1' ] ← これは文字列としてのシンボルの値で、ここでそれは1の値を持っていました。

    studio_interface.Print("Value of a = {}".format(a[3]), "Value Scripts")
    return True
```

注: `CalcValue`呼び出しによって返された各種オブジェクトは内部か、またはMicrochip Studio SDKで資料化されたかの、どちらかのオブジェクトを含みます。出力(エクスポート)された領域を見るのにPythonの`dir()`指令を使ってください。

10. メニューと設定

10.1. 既存のメニューとツールバーの独自化

どのメニューやツールバーでも命令の追加や削除、または順序の変更やそれらの命令の群化を行うことができます。統合開発環境(IDE)内でツールバーの追加と、配置、位置、既存ツールバーの内容の変更も行うことができます。

メニューまたはツールバーに命令を追加

1. Tools(ツール)メニューでCustomize(独自化)をクリックしてください。
2. Customize(独自化)ダイアログ枠内のCommands(命令)タブのChoose a menu or toolbar to rearrange(再配置するメニューかツールバーを選択)下で変更したいメニューかツールバーを選択し、その後にAdd command(命令追加)をクリックしてください。
3. Add command(命令追加)ダイアログ枠内のCategories(区分)一覧で区分名を選択し、その後にCommands(命令)一覧で追加したい命令を選んでください。
4. OKをクリックしてください。
5. Close(閉じる)をクリックしてください。

メニューまたはツールバーから命令を削除

1. Tools(ツール)メニューでCustomize(独自化)をクリックしてください。
2. Customize(独自化)ダイアログ枠内のCommands(命令)タブのChoose a menu or tool bar to rearrange(再配置するメニューかツールバーを選択)下で変更したいメニューかツールバーを選んで選択してください。
3. 削除したい命令を選択し、その後にDelete(削除)をクリックしてください。
4. Close(閉じる)をクリックしてください。

メニューまたはツールバーで命令を分割

1. Tools(ツール)メニューでCustomize(独自化)をクリックしてください。
2. Customize(独自化)ダイアログ枠内のCommands(命令)タブのChoose a menu or tool bar to rearrange(再配置するメニューかツールバーを選択)下で変更したいメニューかツールバーを選択してください。
3. それの上の命令から分割したい命令を選択してください。
4. Modify Selection(選択変更)一覧でBegin a Group(群の始め)をクリックしてください。
5. 命令の一覧で選択した命令の上に分離バーが現れます。
6. OKをクリックしてください。
7. Close(閉じる)をクリックしてください。

その命令はそれの前に分離子を持つメニューまたはツールバーで現れます。

新しいメニューを追加

1. Tools(ツール)メニューでCustomize(独自化)をクリックしてください。
2. Customize(独自化)ダイアログ枠内のCommands(命令)タブでAdd New Menu(新しいメニューを追加)をクリックしてください。New Menu(新しいメニュー)と名付けられたメニューが現れます。
3. Modify Selection(選択変更)一覧で新しいメニュー用の名前を入力してください。
4. OKをクリックしてください。
5. Close(閉じる)をクリックしてください。

メニューの順番を変更

1. Tools(ツール)メニューでCustomize(独自化)をクリックしてください。
2. Customize(独自化)ダイアログ枠内のCommands(命令)タブのChoose a menu or tool bar to rearrange(再配置するメニューかツールバーを選択)下で移動したいメニューかツールバーを選択してください。
3. 命令を移動するためにMove Up(上移動)かMove Down(下移動)を選んで一覧で新しいメニュー用の名前を入力してください。
4. OKをクリックしてください。
5. Close(閉じる)をクリックしてください。

ツールバー作成

1. Tools(ツール)メニューでCustomize(独自化)をクリックしてください。
2. Customize(独自化)ダイアログ枠内のToolbars(ツールバー)タブでNew(新規)をクリックしてください。
3. New Toolbar(新しいツールバー)ダイアログ枠でこのツールバー用の名前を入力してください。
4. ツールバーに命令を追加するためにこの話題で先に記述された手順を使ってください。

ツールバー配置変更

主接合領域内でそれらをドラッグする、またはそれらを他の接合領域へ移動するためにCustomize(独自化)ダイアログ枠を使うことによってツールバーを整理することができます。

主接合領域でツールバーを整理

1. それを望む場所へ移動するためにその左端によってツールバーをドラッグしてください。
2. ツールバーの周辺が自動的に再配置されます。
3. ツールバーの接合位置を変更するため、
4. Tools(ツール)メニューでCustomize(独自化)をクリックしてください。
5. Customize(独自化)ダイアログ枠内のToolbars(ツールバー)タブのModify Selection(選択変更)一覧で接合位置を選んでください。
6. Close(閉じる)をクリックしてください。

主メニューとショートカットメニューのリセット

命令の位置や命令アイコンを変更する場合、それらを元の構成設定にリセットすることができます。

メニューまたはツールバーをリセット

1. Tools(ツール)メニューでCustomize(独自化)をクリックしてください。
2. Customize(独自化)ダイアログ枠内のCommands(命令)タブのChoose a menu or tool bar to rearrange(再配置するメニューかツールバーを選択)下でリセットしたいメニューかツールバーを選択してください。
3. Reset all(全てリセット)をクリックしてください。

選んだメニュー、ツールバー、または状況メニューはそれの元の構成設定に戻ります。

10.2. 設定のリセット

Import and Export settings(設定のインポートとエクスポート)ウィザードを用いて統合開発環境(IDE: Integrated Development Environment)を前の状態にリセットすることができます。

設定をリセット

1. Tools(ツール)メニューでImport and Export Settings(設定のインポートとエクスポート)をクリックしてください。
2. Welcome to the Import and Export Settings Wizard(ようこそ設定のインポートとエクスポートのウィザードへ)ページでReset all settings(全設定をリセット)をクリックし、その後にNext(次へ)をクリックしてください。
3. 現在の設定組み合わせを保存したい場合、Yes, save my current settings(はい、私の現在の設定を保存します。)をクリックしてファイル名を指定し、その後にNext(次へ)をクリックしてください。
--または--
現在の設定組み合わせを削除したい場合、No, just reset settings, overwriting my current settings(いいえ、単に設定をリセットし、私の現在の設定を上書きします。)を選んで、その後にNext(次へ)をクリックしてください。この任意選択は既定設定を削除せず、それは次にこのウィザードを使う時に未だ利用可能です。
4. Which collection of settings do you want to reset to(リセットしたい設定の集合はどれですか?)で一覧から設定の集合を選んでください。
5. Finish(終了)をクリックしてください。

Reset Complete(リセット完了)ページはリセット中に何れかの問題に出会ったことを警告します。

10.3. 任意選択ダイアログ枠

Options(任意選択)ダイアログ枠はあなたの要求に対して統合開発環境(IDE)の構成設定を許します。例えば、あなたのプロジェクト用の既定保存位置を設置、ウィンドウの既定の外観と動きを変更、一般的に使われる命令に対するショートカットを作成することができます。あなたの開発言語と基盤に対して指定する任意選択もあります。Tools(ツール)メニューからOptions(任意選択)にアクセスすることができます。

注: ダイアログ枠で利用可能な任意選択とあなたが見るメニュー命令の名前と位置はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。設定を変更するにはTools(ツール)メニューでImport and Export Settings(設定のインポートとエクスポート)を選んでください。

任意選択ダイアログ枠の配置

Options(任意選択)ダイアログ枠は左側の誘導枠と右側の表示領域の2つの部分に分けられます。誘導枠内の樹状制御部はEnvironment(環境)、Text Editor(テキスト エディタ)、Projects and Solutions(プロジェクトと解決策)、Source Control(ソース制御)のようなフォルダ節点を含みます。それが含む任意選択のページを一覧するにはどれかのフォルダ節点を展開してください。特定のページに対する節点を選ぶと、その任意選択が表示領域に現れます。

IDE機能に対する任意選択はその機能がメモリに読み込まれるまで誘導枠に現れません。従って、新しい作業を始める時と最後に終わる時で同じ任意選択が表示されないかもしれません。プロジェクトを作成、または特定の应用を使う命令を走らせると、Options(任意選択)ダイアログ枠に関連する任意選択が追加されます。これらの追加された任意選択はその後もIDE機能がメモリに留まる限り、利用可能に留まります。

注: いくつかの設定集合はOptionsダイアログ枠の誘導枠内に現れるページ数が範囲です。Show all settings(全ての設定を表示)を選ぶことによって可能な全てのページを見るように選ぶことができます。

任意選択の適用方法

Options(任意選択)ダイアログ枠でのOKクリックは全てのページの全ての設定を保存します。何れかのページのCancel(取り消し)でのクリックは、他のOptions(任意選択)ページで行われたばかりの何れをも含み、全ての変更要求を取り消します。Fonts and Colors(フォントと色)、Environment(環境)、Options(任意選択)ダイアログで行われるそれらのように任意選択設定に対するいくつかの変更はMicrochip Studioを閉じて再び開いた後にだけ有効になります。

10.3.1. 環境任意選択

Options(任意選択)ダイアログ枠の環境フォルダ内のページは統合開発環境(IDE)の或る要素の表示と動きの方法を設定させます。Tools(ツール)メニューでOptions(任意選択)をクリックし、その後にEnvironment(環境)をクリックすることによってEnvironment(環境)ページにアクセスすることができます。

10.3.1.1. 環境全般設定

Items shown in Window menu (ウインドウ メニューで示される項目)

Windows(ウインドウ)メニューのウインドウ一覧で現れるウインドウ数を変更します。1~24の間で数を入力してください。既定による数は10です。

Items shown in recently used lists (最近使われた一覧で示される項目)

Files(ファイル)メニューで現れる最も最近に使われたプロジェクトとファイルの数を変更します。1~24の間で数を入力してください。既定による数は10です。これは最近使ったプロジェクトとファイルを取得する簡単な方法です。

Automatically adjust visual experience based on client performance (クライアントPC性能に基づく視覚体験自動調整)

Microchip Studioが視覚体験に対する調整を自動的に設定するか、または使用者が調整を明示的に設定するかのどちらかに指定します。この調整は色の表示をグラデーションから均一な色へ変更するかもしれませんが、またはメニューやポップアップ ウインドウでアニメーションの使用を制限するかもしれません。

Enable rich client experience (贅沢なクライアントPC体験を許可)

グラデーションとアニメーションを含むMicrochip Studioの完全な視覚体験を許可します。遠隔デスクトップ接続または古いグラフィック アダプタを使っている時は、これらの機能はそれらの場合で貧弱な性能を持ち得るため、この任意選択を解除してください。この任意選択はクライアント任意選択に基づいて視覚体験自動調整を解除した時にだけ利用可能です。

Use hardware graphics acceleration if available (利用可能なら、ハードウェア グラフィック加速を使用)

それが利用可能ならばソフトウェア加速よりもむしろハードウェア グラフィック加速を使います。

Show status bar (状態バー表示)

状態バーを表示します。状態バーはIDEウインドウの底に置かれ、進行中の操作の進捗についての情報を表示します。

Close button affects active tool window only (閉じる鈕は活性なツール ウインドウにだけ影響)

Close(閉じる)鈕がクリックされた時に接合設定に於いてツール ウインドウの全てではなくフォーカスを持つツール ウインドウだけが閉じられることを指定します。既定によってこの任意選択が選ばれます。

Auto Hide button affects active tool window only (自動隠し鈕は活性なツール ウインドウにだけ影響)

Auto Hide(自動隠し)鈕がクリックされた時に接合設定に於いてツール ウインドウの全てではなくフォーカスを持つツール ウインドウだけが自動的に隠されることを指定します。既定によってこの任意選択は選ばれません。

Restore File Associations (ファイル関連付け回復)

Microchip Studioと連携する代表的なファイル形式を登録します。登録はWindowsにWindowsエクスプローラで正しいアイコンを表示させ、これらのファイル形式を開くための正しい応用としてMicrochip Studioを認証させます。

この任意選択は同じコンピュータでインストールされた異なる版のMicrochip Studioを持ち、その後で1つの版をアンインストールした場合に有用で有り得ます。アンインストール後、Microchip Studioファイル用のアイコンはWindowsエクスプローラでもはや現れません。また、Windowsはこれらのファイルを編集するための既定応用としてもはやMicrochip Studioを認証しません。この任意選択はそれらの関係を回復します。

10.3.1.2. アドイン/マクロ保護

10.3.1.2.1. Add-in Security Settings (アドイン保護設定)

自動活性化から悪意のあるアドインを防ぐことによって保護を強化するため、Microchip StudioはAdd-in/Macros Security(アドイン/マクロ保護)と名付けられたTools Options(ツール任意選択)ページでの設定を提供します。

また、この任意選択ページはMicrochip Studioが.Addin登録ファイルに対して検索するフォルダを指定することを許します。これは.Addin登録ファイルを読むことができる場所を制限することを許すことによって保護を強化し、不注意に使われることから悪意のある.Addinファイルを防ぐ手助けをします。

アドイン保護に関連するOptions(任意選択)のダイアログ枠⇒Environment(環境)⇒Add-in/Macros Security(アドイン/マクロ保護)での設定は次のとおりです。

- アドイン構成部品を読み込むことを許します。既定によってチェックされます。チェックされると、アドインはMicrochip Studioに読み込まれることを許されます。チェックされないと、アドインはMicrochip Studioで読み込むことを禁止されます。
- URLからアドイン構成部品を読み込むことを許します。既定によってチェックされません。チェックされると、アドインは外部のウェブ サイトから読み込まれることを許されます。チェックされないと、遠隔アドインはMicrochip Studioで読み込むことを禁止されます。アドインがいくつかの理由のために読み込むことができない場合、それをウェブ から読み込むことはできません。この設定はアドインDLLの読み込みだけを制御します。**.Addin**登録ファイルは常にローカル システムに置かれなければなりません。

10.3.1.3. AutoRecover (自動復元)

ファイルが自動的にバックアップされるかどうかを指定するにはOptions(任意選択)ダイアログ枠のこのページを使ってください。このページは統合開発環境(IDE)が予期せぬ終了をする時に変更されたファイルが復元されるかどうかを指定することも許します。Tools(ツール)メニューを選んでOptions(オプション)を選び、その後にEnvironment folder(環境フォルダ)を選んでAutoRecover(自動復元)ページを選ぶことによってこのダイアログ枠にアクセスすることができます。一覧でこのページが現れない場合、Optionsダイアログ枠でShow all settings(全ての設定を表示)を選んでください。

Save AutoRecover information every <n> minutes (n分毎に自動復元情報を保存)

エディタに於いてファイルがどの位の頻度で自動的に保存されるかを設定するにはこの任意選択を使ってください。以前に保存されたファイルについては、ファイルの複製が¥...¥My Documents¥Atmel Studio¥7.0¥Backup Files¥<プロジェクト名>に保存されます。ファイルが新しくて手動で保存されていない場合は、そのファイルは無作為に生成されたファイル名を用いて自動保存されます。

Keep AutoRecover information for <n> days (n日間自動復元情報を維持)

Microchip Studioがどれ位長く自動復元用に作成されたファイルを維持するかを指定するにはこの任意選択を使ってください。

10.3.1.4. Find and Replace (検索と置換)

メッセージ枠と、検索と置換の操作の他の側面を制御するにはOptions(任意選択)ダイアログ枠のこのページを使ってください。Tools(ツール)メニューからOptions(任意選択)をクリックし、Environment(環境)を展開し、その後にFind and Replace(検索と置換)をクリックすることによってこのダイアログ枠にアクセスすることができます。一覧でこのページが現れない場合、Optionsダイアログ枠でShow all settings(全ての設定を表示)を選んでください。

Display informational messages (情報メッセージを表示)

Always show this message(このメッセージを常に表示)任意選択を持つ全ての検索と置換情報メッセージを表示するにはこの任意選択を選んでください。例えば、"Find reached the starting point of the search(検索開始点に到達)"メッセージ表示が選ばれていなかった場合、この任意選択を選ぶことはこの情報メッセージをFind and Replace(検索と置換)を使う時に再び現れるようにさせます。

Find and Replace(検索と置換)に対してどの情報メッセージも見たくない場合、この任意選択を解除してください。

全部ではなく一部で解除されたAlways show this message(このメッセージを常に表示)任意選択がある時に、検索と置換情報メッセージのDisplay informational messages(情報メッセージ表示)チェック枠が記入されるために現れますが、選択されていません。全ての任意選択の検索と置換のメッセージを復元するにはこの任意選択を解除し、その後に再びそれを選んでください。

注: この任意選択はAlways show this message任意選択を表示しないどの検索と置換の情報メッセージにも影響を及ぼしません。

Display warning messages (警告メッセージ表示)

Always show this message(このメッセージを常に表示)任意選択を持つ全ての警告的な検索と置換のメッセージを表示するにはこの任意選択を選んでください。例えば、編集のために現在開かれていないファイルで置換をしようとする時に現れるReplace All(全部置換)警告メッセージを表示しないように選んだ場合、この任意選択の選択はReplace Allを試みる時にこの警告メッセージを再び現れるようにさせます。

Find and Replace(検索と置換)に対してどの警告的なメッセージも見たくない場合、この任意選択を解除してください。

全部ではなく一部で解除されたAlways show this message(このメッセージを常に表示)任意選択がある時に、検索と置換警告メッセージのDisplay warning messages(警告メッセージ表示)チェック枠が記入されるために現れますが、選択されていません。全ての任意選択の検索と置換のメッセージを復元するにはこの任意選択を解除し、その後に再びそれを選んでください。

注: この任意選択はAlways show this message任意選択を表示しないどの検索と置換の情報メッセージにも影響を及ぼしません。

Automatically populate Find What with text from the editor (エディタからのテキストでFind What領域自動投入)

Edit(編集)メニューからFind and Replace(検索と置換)ウィンドウの何れかの表示部を選ぶ時にFind what(何を検索)領域内に現在のエディタの挿入点のどちらかの側のテキストを張り付けるにはこの任意選択を選んでください。Find what文字列として以前の検索から最後の検索様式を使うにはこの任意選択を解除してください。

Hide Find and Replace window after a match is located for Quick Find or Quick Replace (Quick Find(迅速検索)またはQuick Replace(迅速置換)に対して一致場所が見つかった後に検索と置換のウィンドウを隠す)

Quick Find(迅速検索)に対して最初の一致が見つかった時にFind and Replace(検索と置換)ウィンドウを自動的に閉じるにはこの任意選択を選んでください。次の一致へ行くにはEdit.FindNextに対するショートカット キー、通常はF3、または再びFind and Replaceウィンドウを使ってください。

10.3.1.5. Fonts and Colors (フォントと色)

Options(任意選択)ダイアログ枠の**Fonts and Colors**(フォントと色)ページは統合開発環境(IDE)に於ける様々なユーザーインターフェース要素に対して独自のフォントと配色を確立させます。**Tools**(ツール)メニューで**Options**(任意選択)をクリックし、その後に**Environment**(環境)フォルダで**Fonts and Colors**(フォントと色)ページを選ぶことによってこのダイアログ枠にアクセスすることができます。一覧でこのページが現れない場合、**Options**ダイアログ枠で**Show all settings**(全ての設定を表示)を選んでください。

注: あなたが見るダイアログ枠とメニュー命令はあなたの有効な設定や版に応じてヘルプで記述されるそれらと違うかもしれません。設定を変更するには**Tools**(ツール)メニューで**Import and Export Settings**(設定のインポートとエクスポート)を選んでください。

配色変更はそれらを行う作業の間は効力を発揮しません。Microchip Studioの別の実体を開いて変更が適合する意図する状況を提示することによって色変更を評価することができます。

Show settings for (～用設定を表示)

フォントと配色を変更することができるユーザーインターフェース要素の全てを一覧にします。この一覧から項目を選択後、**Display**(表示)項目で選ばれた項目に対して色設定を独自設定することができます。

Text Editor (テキスト エディタ)

テキスト エディタ用のフォントの種類、大きさ、色の表示設定変更は既定テキスト エディタでテキストの見かけに影響を及ぼします。IDEの外側のテキスト エディタで開かれた文書はこれらの設定によって影響を及ぼされません。

Printer (プリンタ)

プリンタ用のフォントの種類、大きさ、色の表示設定変更は印刷される文書に於ける見かけに影響を及ぼします。

注: 必要に応じて、テキスト エディタで表示されるのに使われるものと違う印刷用既定フォントを選ぶことができます。これは1バイトと2バイトの両方の文字を含むコードを印刷する時に有効で有り得ます。

Statement Completion (文完成)

エディタ内の文完成ポップアップに現れるテキスト用のフォントの種類と大きさを変更します。

Editor Tool tip (エディタ助言)

エディタで表示される助言(ToolTips)に現れるテキスト用のフォントの種類と大きさを変更します。

Environment Font (環境フォント)

Show Settings for(～用設定を表示)で独立した任意選択を未だ持たない全てのIDEユーザーインターフェース要素に対してフォントの種類と大きさを変更します。例えば、この任意選択は開始ページに適用しますが、**Output**(出力)ウィンドウに影響を及ぼしません。

[All Text Tool Windows] (全テキスト ツール ウィンドウ)

IDEで出力枠を持つツール ウィンドウ内でテキストの見かけに影響を及ぼすこの項目に対してフォントの種類、大きさ、色の表示設定を変更します。例えば、**Output**(出力)ウィンドウ、**Command**(命令)ウィンドウ、**Immediate**(即値)ウィンドウなど。

注: **[All Text Tool Windows]**(全テキスト ツール ウィンドウ)項目のテキストに対する変更はそれらを行う作業中に効力を発揮しません。Microchip Studioの別の実体を開くことによってそのような変更を評価することができます。

Use Defaults/Use (既定/使用を使用)

Show settings for(～用設定を表示)で選ばれた一覧項目のフォントと色の値をリセットします。選択に対して他の表示体系が利用可能な時に**Use**(使用)鉤が現れます。例えば、プリンタに対しては2つの体系から選ぶことができます。

Font (bold type indicates fixed-width fonts) (フォント(太字は固定幅を示す))

システムにインストールされた全てのフォントを一覧にします。引き落としメニューが先に現れる時は**Show settings for**(～用設定を表示)領域で選ばれた要素に対する現在のフォントが強調表示されます。固定フォント - エディタでの整列がより容易 - 太字で出現

Size (大きさ)

強調表示されたフォントに対する利用可能なポイントでの大きさを一覧にします。フォントの大きさ変更は**Show settings for**(～用設定を表示)選択に対する全ての**Display**(表示)項目に影響を及ぼします。

Display items (表示項目)

前面と背面の色を変更することができる項目を一覧にします。

注: **Plain Text**(平文)が既定表示項目です。そのため、**Plain Text**に割り当てられる特性は他の表示項目に割り当てられた特性によって上書きされます。例えば、**Plain Text**に青色を**Identifier**(識別子)に緑色を割り当てた場合、全ての**Identifier**は緑で出現します。この例では**Identifier**特性が**Plain Text**特性を上書きします。

表示項目のいくつかは以下を含みます。(表示項目、次行からその説明)

Plain Text (平文)

エディタ内の文

Selected Text (選択した文)

エディタがフォーカスを持つ時に現在の選択に含まれる文

Inactive Selected Text (不活性な選択した文)

エディタがフォーカスを失った時に現在の選択に含まれる文

Indicator Margin (指標部余白)

中断点と葉のアイコンが表示されるコード エディタの左側の余白

Line Numbers (行番号)

コードの各行の傍に現れる任意選択の番号

Visible White Space (空白表示)

空白、タブ、次行送り(ワードラップ)の指標部

Bookmark (葉)

葉を持つ行。葉はIndicator Margin(指標部余白)が禁止されている場合にだけ見えます。

Brace Matching (Highlight) (大括弧一致 (強調表示))

その強調表示は一致する大括弧に対して代表的に太字形式です。

Brace Matching (Rectangle) (大括弧一致 (矩形))

その強調表示は代表的に背面での灰色矩形です。

Breakpoint (Enabled) (中断点 (許可))

単純な中断点を含む文または行に対する強調表示色を指定します。この任意選択は文レベル中断点の有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用可能です。

Breakpoint (Error) (中断点 (異常))

異常状態の中断点を含む文または行に対する強調表示色を指定します。文レベル中断点の有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用できます。

Breakpoint (Warning) (中断点 (警告))

警告状態の中断点を含む文または行に対する強調表示色を指定します。文レベル中断点の有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用できます。

Breakpoint - Advanced (Disabled) (中断点 - 高度 (禁止))

禁止された条件付きまたは的中回数付き中断点を含む文または行に対する強調表示色を指定します。文レベル中断点の有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用できます。

Breakpoint - Advanced (Enabled) (中断点 - 高度 (許可))

許可された条件付きまたは的中回数付き中断点を含む文または行に対する強調表示色を指定します。文レベル中断点の有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用できます。

Breakpoint - Advanced (Error) (中断点 - 高度 (異常))

異常状態の条件付きまたは的中回数付き中断点を含む文または行に対する強調表示色を指定します。文レベル中断点の有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用できます。

Breakpoint - Advanced (Warning) (中断点 - 高度 (警告))

警告状態の条件付きまたは的中回数付き中断点を含む文または行に対する強調表示色を指定します。文レベル中断点の有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用できます。

Code Snippet Dependent Field (コード断片依存領域)

現在の編集可能領域が変更される時に更新される領域

Code Snippet Field Editable (コード断片領域編集可)

コード断片が有効な時の領域

Collapsible Text (折り畳み可能テキスト)

コード エディタ内で表示をON/OFF交互切換することができるテキストまたはコードの塊

Comment (注釈)

コードの注釈

Compiler Error (コンパイラ異常)

コンパイル異常を示すエディタ内の青い波線

Coverage Not Touched Area (網羅範囲は接触領域外)

単位部検査によって網羅されなかったコード

Coverage Partially Touched Area (網羅範囲は部分的に接触領域)

単位部検査によって部分的に網羅されたコード。単位部検査によって完全に網羅された網羅範囲接触領域コード

Current list location (現在の一覧位置)

Output(出力)ウィンドウや**Find Results**(検索結果)ウィンドウのような一覧ツール ウィンドウで誘導された現在行

Current Statement (現在の文)

デバッグ時に現在の段階位置を示すソース文または行に対する強調表示色を指定します。

Debugger Data Changed (デバッグ データ変化)

Registers(レジスタ)と**Memory**(メモリ)のウィンドウの内側で変更されたデータを表示するのに使われるテキストの色

Definition Window Background (定義ウィンドウ背面)

Code Definition(コード定義)ウィンドウの背面色

Definition Window Current Match (定義ウィンドウ現在の一致)

Code Definition(コード定義)ウィンドウの現在の定義

Disassembly File Name (逆アセンブリ ファイル名)

Disassembly(逆アセンブリ)ウィンドウの内側で中断するファイル名を表示するのに使われるテキストの色

Disassembly Source (逆アセンブリ供給元)

Disassembly(逆アセンブリ)ウィンドウの内側でソース行を表示するのに使われるテキストの色

Disassembly Symbol (逆アセンブリ シンボル)

Disassembly(逆アセンブリ)ウィンドウの内側でシンボル名を表示するのに使われるテキストの色

Disassembly Text (逆アセンブリ テキスト)

Disassembly(逆アセンブリ)ウィンドウの内側でオペコードとデータを表示するのに使われるテキストの色

Excluded Code (除外コード)

#ifのような条件付き前処理部(プリプロセッサ)指令によってコンパイルされないコード

Identifier (識別子)

クラス名、メソッド名、変数名のようなコードに於ける識別子

Keyword (キーワード)

予約された与えられた言語のためのキーワード。例えば、**class**と**namespace**

Memory Address (メモリ アドレス)

Memory(メモリ)ウィンドウの内側でアドレス列を表示するのに使われるテキストの色

Memory Changed (変更されたメモリ)

Memory(メモリ)ウィンドウの内側で変更されたデータを表示するのに使われるテキストの色

Memory Data (メモリ データ)

Memory(メモリ)ウィンドウの内側でデータを表示するのに使われるテキストの色

Memory Unreadable (読み込み不可メモリ)

Memory(メモリ)ウィンドウ内で読み込み不可メモリ領域を表示するのに使われるテキストの色

Number (数)

実際の数値を表すコードでの数

Operators (演算子)

+, -, !=のようなもの

Other Error (他の異常)

他の異常波線によって網羅されない他の異常形式。現在、これは<guimenuitem>Edit and Continue</guimenuitem>での無作法な編集を含みます。

Preprocessor Keyword (前処理部キーワード)

#includeのように前処理部(プリプロセッサ)によって使われるキーワード

Read-Only Region (読み込み専用領域)

編集することができないコード。例えば、**Code Definition**(コード定義)表示部ウィンドウで表示されるコード、または編集と継続の間で編集することができないコード。

Register Data (レジスタ データ)

Register(レジスタ)ウィンドウの内側でデータを表示するのに使われるテキストの色

Register NAT (レジスタNAT)

Register(レジスタ)ウィンドウの内側で未認証のデータとオブジェクトを表示するのに使われるテキストの色

Stale Code (古いコード)

更新を待っている取って代わられるコード。或る場合で、編集と継続はコード変更を直ちに適用することができませんが、デバッグを続けるために後でそれらを適用します。これは現在実行している関数を呼び出さなければならない関数を編集する場合、または呼び出しスタックで待っている関数に対して64バイトを超える新しい変数を追加する場合に起きます。これが起こると、デバッグは”Stale Code Warning(古いコード警告)”ダイアログ枠を表示し、問題の関数が終わって再び呼ばれるまで、取って代わられるコードが実行を続けます。その時にコード変更を適用するために編集して継続してください。

String (文字列)

文字どおりの文字列

Syntax Error (構文誤り)

解析異常

Task List Shortcut (タスク一覧ショートカット)

タスク一覧ショートカットが行に追加されて指標余白が禁止される場合、その行は強調表示されます。

Tracepoint (Enabled) (追跡点 (許可))

単純な追跡点を含む文または行に対して強調表示色を指定します。この任意選択は文レベル追跡点が有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用できます。

Tracepoint (Error) (追跡点 (異常))

異常状態の追跡点を含む文または行に対して強調表示色を指定します。この任意選択は文レベル追跡点の有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用できます。

Tracepoint (Warning) (追跡点 (警告))

警告状態の追跡点を含む文または行に対して強調表示色を指定します。この任意選択は文レベル追跡点の有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用できます。

Tracepoint – Advanced (Disabled) (追跡点 – 高度 (禁止))

禁止された条件付きまたは的中回数付き追跡点を含む文または行に対して強調表示色を指定します。この任意選択は文レベル追跡点の有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用できます。

Tracepoint – Advanced (Enabled) (追跡点 – 高度 (許可))

許可された条件付きまたは的中回数付き追跡点を含む文または行に対して強調表示色を指定します。この任意選択は文レベル追跡点の有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用できます。

Tracepoint – Advanced (Error) (追跡点 – 高度 (異常))

異常状態の条件付きまたは的中回数付き追跡点を含む文または行に対して強調表示色を指定します。この任意選択は文レベル追跡点の有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用できます。

Tracepoint – Advanced (Warning) (追跡点 – 高度 (警告))

警告状態の条件付きまたは的中回数付き追跡点を含む文または行に対して強調表示色を指定します。この任意選択は文レベル追跡点の有効か、またはOptions(任意選択)ダイアログ枠のGeneral(全般)やDebugging(デバッグ)でHighlight entire source line for breakpoints or current statement(中断点または現在の文に対してソース行全体を強調表示)任意選択が選ばれる場合にだけ適用できます。

Track Changes after save (保存後の変更経緯)

ファイルが開かれたけれどもディスクに保存されてから変更されたコードの行

Track Changes before save (保存前の変更経緯)

ファイルが開かれたけれどもディスクに保存されていない時から変更されたコードの行

User Types (使用者型)

使用者によって定義される型

User Types (Delegates) (使用者型 (代表))

代表に使われる型の色

User Types (Enums) (使用者型 (列挙))

列挙に使われる型の色

User Types (Interfaces) (使用者型 (インターフェース))

インターフェースに使われる型の色

User Types (Value types) (使用者型 (値型))

Cでの構造体のような値型に対する型の色

Warning (警告)

コンパイルの警告

Warning Lines (警告行)

静的な警告行分析に使われるパス

XML Attributer (XML属性)

属性名

XML Attribute Quotes (XML属性引用)

XML属性に対する引用文字

XML Attribute Value (XML属性値)

XML属性の内容

XML Cdata Section (XML CDATA項)

<![CDATA[...]]>の内容

XML Comment (XML注釈)

<!-- -->の内容

XML Delimiter (XML分離子)

<, <?, <!, <!--, -->, >, <![,]], [,]を含むXML構文分離子

XML Doc Attribute (XML文書属性)

“1”が色付けされた<param name="1">のようなXML文書属性の値

XML Doc Comment (XML文書注釈)

XML文書注釈内に囲まれた注釈

XML Doc Tag (XML文書の札)

/// <summary>のようなXML文書注釈内の札

XML Keyword (XMLキーワード)

CDATA, IDREF, NDATAのようなDTDキーワード

XML Name Element (XML名前要素)

名前と処理命令目的対象名

XML Processing Instruction (XML処理命令)

目的対象名を含まない処理命令の内容

XML Text Plain (XMLテキスト面)

テキスト要素内容

XSLT Keyword (XSLTキーワード)

XSLT要素名

Item foreground (項目前面)

表示項目で選ばれた項目の前面に対して選ぶことができる利用可能な色を一覧にします。いくつかの項目が関連され、従って一貫した表示体系を維持するため、テキストの前面色変更は**Compiler Error**(コンパイル異常)、**Keyword**(キーワード)、**Operator**(演算子)のような要素に対する既定も変更します。自動項目は**Plain Text**(平文)のような他の表示項目から前面色を継承することができます。この任意選択使用は、継承される表示項目の色を変えると、関連する表示項目も自動的に変わります。例えば、**Compiler Error**(コンパイル異常)に対して**Automatic**(自動)値を選び、後で**Plain Text**(平文)の色を赤に変えた場合、**Compiler Error**(コンパイル異常)も自動的に赤色を継承します。AVR Studio 5初回開始時に項目に対して現れる色を初期値にしてください。**Use Defaults**(初期値を使用)釦のクリックはこの色のリセットします。**Custom Display the Color**(色を独自表示)ダイアログ枠は**Display items**(表示項目)一覧で選んだ項目に対する独自の色を設定することを許します。

注: 独自の色を定義する能力はコンピュータ表示に対する色設定によって制限されるかもしれません。例えば、コンピュータが256色に設定され、**Color**(色)ダイアログ枠から独自の色を選ぶ場合、IDEは最も近い利用可能な基本色を既定とし、**Color preview**(色事前表示)枠で色塊を表示します。

Item background (項目背面)

Display items(表示項目)で選ばれた項目に対する背面色を選ぶことができる色パレットを提供します。

いくつかの項目が関連され、従って一貫した表示体系を維持するため、テキストの背面色変更は**Compiler Error**(コンパイル異常)、**Keyword**(キーワード)、**Operator**(演算子)のような要素に対する既定も変更します。

自動項目は**Plain Text**(平文)のような他の表示項目から背面色を継承することができます。

この任意選択使用は、継承される表示項目の色を変えると、関連する表示項目も自動的に変わります。例えば、**Compiler Error**(コンパイル異常)に対して**Automatic**(自動)値を選び、後で**Plain Text**(平文)の色を赤に変えた場合、**Compiler Error**(コンパイル異常)も自動的に赤色を継承します。

Use Defaults(初期値を使用)鈕のクリックはこの色をリセットします。**Custom Display the Color**(色を独自表示)ダイアログ枠は**Display items**(表示項目)一覧で選んだ項目に対する独自の色を設定することを許します。選んだ**Display items**(表示項目)のテキストを太字で表示するには**Bold Select this**(これを太字選択)任意選択を用いてください。太字はエディタでの識別が容易です。**Sample Display**(見本表示)は**Show settings for**(~に対する設定表示)と選ばれた**Display items**(表示項目)に対するフォントの種類、大きさ、配色の見本を表示します。異なる形式任意選択で実験をする時の結果を事前表示するのにこの枠を使うことができます。

10.3.1.6. Language and International Settings (言語と国際の設定)


International Settings(国際設定)ページはマシンで複数の統合開発環境(IDE)の言語版を持つ時に既定言語を変更することを許します。

Tools(ツール)メニューから**Options**(任意選択)を選び、その後に**Environment**(環境)フォルダから**International Settings**(国際設定)を選ぶことによってこのダイアログ枠にアクセスすることができます。一覧でこのページが現れない場合、**Options**(任意選択)ダイアログ枠で**Show all settings**(全ての設定を表示)を選んでください。

このページで行うどの変更も既定IDEにだけ適用され、環境が再始動されるまで実施されません。

Language (言語)

インストールされた製品言語版に対して利用可能な言語を一覧にします。マシンにインストールされた複数の言語版を持たない限り、この任意選択は利用不可です。複数言語の製品または混合言語インストールの製品が環境を共有する場合、言語選択はMicrosoft Windowsと同じように変更されます。

 **注意** インストールされた複数言語を持つシステムでは構築ツールがこの設定によって影響を及ぼされません。これらのツールは最後にインストールされた言語用の版を使い、構築ツールが付随DLL様式を使わないため、以前にインストールされた言語用のツールは上書きされます。

9.3.1.7. Keyboard Settings (キーボード設定)

現在適用された機構でのショートカットキーの組み合わせは、あなたが行ったかもしれないどの独自化だけでなくあなたが選んだ設定に依存します。Visual Studioは7つの他のキーボード配置体系も含み、各々は既定によって割り当てられたショートカットキーに於いて様々なUI要素に関して他と違います。

全域有効範囲の一部であるショートカットキー組み合わせを持つ命令は統合開発環境(IDE)の現在の状況に依存する他の有効範囲での命令と取り換えることができます。例えば、ファイルを編集している場合、テキストエディタ有効範囲の一部である命令は同じキー組み合わせで開始する全域有効範囲内の命令より優先権を持ちます。例えば、いくつかの全域命令が**Ctrl+K**で開始するキーの組み合わせを持ち、テキストエディタも**Ctrl+K**で開始するキーの組み合わせを持ついくつかの命令を持つ場合、コードを編集すると、テキストエディタのキー組み合わせが動き、全域キー組み合わせは無視されます。

注: ダイアログ枠で利用可能な任意選択とあなたが見るメニュー命令の名前と位置はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。このヘルプページは全般的な開発設定を念頭に置いて書かれました。設定を変更するには**Tools**(ツール)メニューから**Import and Export Settings**(インポートとエクスポートの設定)を選んでください。より多くの情報については設定での動きをご覧ください。

命令に割り当てるショートカットキーを決定

それが割り当てられたショートカットキーの組み合わせを持つかどうかを決定するのに手動で命令を検索することができます。

命令に対してショートカットキーの組み合わせを決定

1. **Tools**(ツール)メニューで**Options**(任意選択)をクリックしてください。
2. **Environment**(環境)フォルダを展開して**Keyboard**(キーボード)を選んでください。
注: **Keyboard**(キーボード)ページが見えない場合、**Options**(任意選択)ダイアログ枠の左下に位置する**Show all settings**(全ての設定を表示)をチェックしてください。
3. 一覧で正しい命令を選んでください。
例えば、**View.SolutionExplorer**(解決策エクスプローラ表示)
4. 命令に対してショートカットキーの組み合わせが存在している場合、その組み合わせは選んだ命令引き落とし一覧に対する**Shortcut(s)**(ショートカット)で現れます。

独自ショートカット キー作成

どれかの命令に対して新しいショートカット キーを作成したり、存在する組み合わせを持つ命令に対するショートカット キー組み合わせを変更することができます。

新しいショートカット キー組み合わせを作成

1. Tools(ツール)メニューでOptions(任意選択)をクリックしてください。
2. Environment(環境)フォルダを展開してKeyboard(キーボード)を選んでください。
注: Keyboard(キーボード)ページが見えない場合、Options(任意選択)ダイアログ枠の左下に位置するShow all settings(全ての設定を表示)をチェックしてください。
 Show commands containing(命令内容表示)枠で空白なしで命令の名前を入力してください。
 例えば、solutionexplorer(解決策エクスプローラ)
3. 一覧でショートカット キー組み合わせに割り当てたい命令を選んでください。
4. Use new shortcut(新しいショートカットを使用)引き落とし一覧でショートカットを使いたい機能領域を選んでください。例えば、ショートカットを状況で動くようにしたいなら、Global(全域)を選ぶことができます。別のエディタで(Globalとして)同じショートカットが配置されていない限り、それを使うことができます。さもないと、エディタはショートカットを無効にします。
注: 以下のキーはGlobalで命令に割り当てることができません。Print Screen/System Request、Scroll Lock、Pause/Break、Tab、Caps Lock、Insert、Home、End、Page Up、Page Down、Windowsロゴキー、応用キー、矢印キーのどれも、Enter、Num Lock、Delete、数字キー部のClear、Ctrl+Alt+Delete
5. Press shortcut key(s)(ショートカット キー押下)枠にカーソルを置いて、その後に命令への使用を意図するキーの組み合わせを入力するのにキーボードを使ってください。
注: ショートカットは文字との組み合わせでShift、Alt、Ctrlを含むことができます。配置体系に於いてキー組み合わせが既に別の命令に割り当てられているかを見るために、Shortcut currently used by(~によって使われ現在のショートカット)枠を調べることを確実にしてください。組み合わせが既に使われている場合、別の組み合わせを入力する前にキー組み合わせを削除するためにBack Spaceを押してください。
6. Assign(割り当て)をクリックしてください。
注: Assign(割り当て)鉤を用いることによって行った変更はCancel(取り消し)鉤をクリックした場合に取り消されません。

エクスポートとインポートのショートカット キー

他のものがデータをインポートできるように情報をファイルにエクスポート(出力)することによって現在のキーボード配置体系でショートカット キー組み合わせを共用することができます。

ショートカット キーだけをエクスポート

1. Tools(ツール)メニューでImport and Export Settings(設定のインポートとエクスポート)ウィザードを選んでください。
2. Export select environment settings(選択環境設定をエクスポート)を選び、その後にNext(次へ)をクリックしてください。
3. What settings do you want to export?(エクスポートしたい設定は何?)下で既定によって選ばれた全ての区分を解除してください。
4. Options(任意選択)を展開し、その後にEnvironment(環境)を展開してください。
5. Keyboard(キーボード)を選び、その後にNext(次へ)をクリックしてください。
6. What do you want to name your settings file?(設定ファイル名は?)に対し名前を入力し、その後にFinish(終了)をクリックしてください。

ショートカット キーだけをインポート

1. Tools(ツール)メニューでImport and Export Settings(設定のインポートとエクスポート)ウィザードを選んでください。
2. Import select environment settings(選択環境設定をインポート)を選び、その後にNext(次へ)をクリックしてください。
3. 単に新しい設定をインポートして現在の設定に上書きするのでNo(いいえ)をクリックし、その後にNext(次へ)をクリックしてください。
4. My Settings(私の設定)下でインポートしたいショートカット キーを含む設定ファイルを選ぶか、または正しい設定ファイルの位置を突き止めるためBrowse(検索)をクリックしてください。
5. Next(次へ)をクリックしてください。
6. Which settings do you want to import?(インポートしたい設定はどれ?)下で全ての区分を解除してください。
7. Options(任意選択)を展開し、その後にEnvironment(環境)を展開してください。
8. Keyboard(キーボード)を選び、その後にFinish(終了)をクリックしてください。

10.3.1.8. 開始ページ - Microchip Studio開始時に表示される既定UIを変更

1. Tools(ツール)メニューでOptions(任意選択)を選んでください。
2. Environment(環境)フォルダを展開してStartup(始動)を選んでください。
3. At start-up(始動で)引き落とし一覧から任意選択の1つを選んでください。
4. OKをクリックしてください。

この変更は次回Microchip Studio開始後に効果を生じます。

Microchip Studio開始時に、もしあれば何の内容かまたは使用者インターフェース(UI)を指定するのにこのページを使ってください。このページにアクセスするにはTools(ツール)メニューでOptions(任意選択)をクリックし、Environment(環境)フォルダを展開し、その後にStartup(始動)をクリックしてください。Options(任意選択)ダイアログ枠の一覧でこのページが現れない場合、Show all settings(全ての設定を表示)を選んでください。

注: ダイアログ枠で利用可能な任意選択とあなたが見るメニュー命令の名前と位置はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。このヘルプ ページは全般的な開発設定を念頭に置いて書かれました。設定を変更するにはTools(ツール)メニューからImport and Export Settings(設定のインポートとエクスポート)をクリックしてください。

At start-up (始動で)

Microchip Studioを開始する度に見たいものを指定することができます。

Open Home page (ホーム ページを開く)

Options(任意選択)ダイアログ枠⇒Environment(環境)⇒Web Browser(ウェブ ブラウザ)⇒Home page(ホームページ)任意選択によって指定される既定ウェブ ページを表示します。

Load last loaded solution (最後に読み込んだ解決策を読み込み)

最後に保存された解決策をその以前の状態で読み込みます。それが最後に閉じられた時にその解決策で開かれていたどのファイルもMicrochip Studio開始時に開かれて表示されます。Microchip Studioを抜ける時に解決策が全く読み込まれていない場合、戻るときにも解決策は全く読み込まれません。

Show Open Project dialog box (プロジェクトを開くダイアログ枠を表示)

Microchip Studio開始時にOpen Project(プロジェクトを開く)ダイアログ枠を表示します。このダイアログ枠はOptions(任意選択)ダイアログ枠⇒Environment(環境)⇒Projects and Solutions(プロジェクトと解決策)のMicrochip Studio Projects location(Microchip Studioプロジェクト位置)領域で設定されるフォルダを使います。

Show New Project dialog box (新しいプロジェクト ダイアログ枠を表示)

Microchip Studio開始時にNew Project(新しいプロジェクト)ダイアログ枠を表示します。

Show empty environment (空の環境を表示)

Microchip Studio開始時に空の統合開発環境(IDE)を表示します。

Show Start Page (開始ページを表示)

Microchip Studio開始時に現在適用される設定で関連付けされた開始ページを表示します。

Start Page news channel (開始ページのニュース チャンネル)

開始ページのMicrochip Studio News項の内容を表示するのに使われるRSSフィードを指定します。

Download content every n minutes (n分毎に内容をダウンロード)

開始ページ用の新しいRSSフィード内容と製品見出しをどの位の頻度でIDEが調べるかを指定します。この設定が選ばれない場合、新しいRSSフィード内容と製品見出しは開始ページにダウンロードされません。

Customize Start Page (開始ページ独自化)

インストールされた独自開始ページがある場合、どの開始ページが読み込まれるかを指定することができます。Customize Start Page(開始ページ独自化)ドロップ ダウン リストは既定Microchip Studio開始ページを読み込むための入り口(Default Start Page(既定開始ページ))とシステム上の各独自開始ページ用の入り口を含みます。

使用者開始ページディレクトリ内のどの.XAMLファイルも独自開始ページを考慮されます。より多くの情報については独自開始ページをご覧ください。

10.3.1.9. Import and Export Settings (設定のインポートとエクスポート)

サーバー上に保存されるチーム設定ファイルを使うかどうかを指定するだけでなく、保存する設定ファイルに対する特性を設定するのにもOptions(任意選択)ダイアログ枠のこのページを使ってください。Tools(ツール)メニューからOptions(任意選択)を選んでEnvironment(環境)フォルダからImport and Export Settings(設定のインポートとエクスポート)ページを選ぶことによってこのダイアログ枠にアクセスすることができます。



助言: 一覧でこのページが現れない場合、Options(任意選択)ダイアログ枠でShow all settings(全ての設定を表示)を選んでください。

注: ダイアログ枠で利用可能な任意選択とあなたが見るメニュー命令の名前と位置はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。このヘルプ ページは全般的な開発設定を念頭に置いて書かれました。設定を変更するにはTools(ツール)メニューからImport and Export Settings(設定のインポートとエクスポート)を選んでください。

自動の読み込みと保存の設定

Automatically save my settings to this file: (私の設定をこのファイルに自動保存)

現在使っている.vssettingsファイルの場所と名前を表示します。IDEを閉じる時にウィンドウ移動や任意選択変更のようにあなたが行ったどの変更もこの現在のファイルに保存されます。次にIDEを開始する時にこの設定が読まれて設定されます。

チーム設定

Use team settings file: (チーム設定ファイルを使用)

選択時、**Browse**(検索) 釦を使うことによって共有する **.vssettings** ファイルへ誘導することを許します。この設定ファイルはより新しい版が利用可能な場合を Microchip Studio が検出する度毎に自動的に再適用されます。

注: チーム設定ファイルの場所は UNC パスまたは局所パスとして指定されなければなりません。URL と他の規約は支援されないパスです。

10.3.1.10. Task List (タスク一覧)

この任意選択は **Task List** (タスク一覧) 覚書を生成する注釈通票の追加、削除、変更を許します。これらの設定を表示するには **Tools** (ツール) メニューから **Options** (任意選択) を選び、**Environment** (環境) フォルダを展開し、**Task List** (タスク一覧) を選んでください。

Confirm deletion of tasks (タスクの削除を確認)

選択時、使用者タスクが **Task List** (タスク一覧) から削除される時に必ずメッセージ 枠が表示され、削除の確認を許します。

注: **Task Comment** (タスク注釈) を削除するには注釈を見つけるためにリンクを使い、その後にあなたのコードからそれを取り去ってください。

Hide full file paths (完全なファイルパスを隠す)

選択時、**Task List** (タスク一覧) の **File** (ファイル) 列はそれらの完全なパスではなく、編集されるべきファイル名だけを表示します。

通票 (トークン)

Token List (通票一覧) からの通票で始まるテキストのあなたのコードに注釈を挿入する時に、**Task List** (タスク一覧) は編集のためにファイルが開かれる時は必ず新しい入り口としてあなたの注釈を表示します。コード内の注釈行へ直接飛ぶのにこのタスク一覧入り口をクリックすることができます。

Token List (通票一覧)

通票の一覧を表示、独自通票の追加と削除を許します。注釈通票は大文字/小文字を区別します。

注: **Token List** (通票一覧) で現れるように望む通票を正確に入力しなかった場合、注釈タスクが通票一覧で表示されません。

Priority (優先権)

選んだ通票を使うタスクの優先権を設定します。この通票で始めるタスク注釈は **Token List** (通票一覧) で指示された優先権を自動的に割り当てられます。

Name (名前)

通票文字列を入力してください。これは **Add** (追加) 釦を許可します。**Add** (追加) でこの文字列が **Token List** (通票一覧) に含まれ、この名前で始まる注釈が **Task List** (タスク一覧) に表示されます。

Add (追加)

新しい名前を入力する時に許可されます。**Name** (名前) と **Priority** (優先権) の領域で入力された値を用いて新しい通票を追加するにはクリックしてください。

Delete (削除)

Token List (通票一覧) から選ばれた通票を削除するにはクリックしてください。既定注釈通票を削除することはできません。

Change (変更)

Name (名前) と **Priority** (優先権) の領域で入力された値を用いて既存の通票に変更を行うにはクリックしてください。

注: 既定注釈通票を改名または削除することはできませんが、その優先権レベルを変更することができます。

10.3.1.11. ウェブ ブラウザ任意選択

内部ウェブ ブラウザとインターネット エクスプローラの両方に対する任意選択を設定します。このダイアログ 枠にアクセスするには **Tools** (ツール) メニューで **Options** (任意選択) をクリックし、**Environment** (環境) フォルダを展開し、**Web Browser** (ウェブ ブラウザ) を選びます。

注: あなたが見るダイアログ 枠とメニュー命令はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。設定を変更するには **Tools** (ツール) メニューから **Import and Export Settings** (設定のインポートとエクスポート) を選んでください。より多くの情報については設定での動きをご覧ください。



注意: ウェブ から或るファイルや構成部品を開くことはコンピュータ上でコードを実行することで有り得ます。

Home page (ホームページ)

統合開発環境ウェブ ページを開く時に表示されるページを設定します。

Search page (検索ページ)

内部ウェブ ブラウザに対して **Search** (検索) ページの指示を許します。この場所は統合開発環境 (IDE) の外側で始められるインターネット エクスプローラの実体によって使われる検索ページと異なるようにすることができます。

View Source in (～にソースを表示)

内部ウェブ ブラウザからページ上の **View Source** (ソースを表示) を選ぶ時にウェブ ページを開くのに使われるエディタを設定します。

Source editor (ソース エディタ)

コードとテキストのエディタでソースを見るために選びます。

HTML editor (HTMLエディタ)

HTML設計部でソースを見るために選びます。設計表示部、またはテキストに基づく標準ソース表示部の2つの表示部の1つでウェブ ページを編集するにはこの選択を使ってください。

External editor (外部エディタ)

別のエディタでソースを見るために選びます。選んだ何れかのエディタ、例えば、**Notepad.exe**のパスを指定してください。

Internet Explorer Options (インターネット エクスプローラ任意選択)

Internet Properties(インターネット プロパティ)ダイアログ枠でインターネット エクスプローラに対する任意選択を変更するにはクリックしてください。このダイアログ枠で行った変更は内部ウェブ ブラウザとMicrochip Studio IDEの外側で(例えば、STARTメニューから)始められたインターネット エクスプローラの実体の両方に影響を及ぼします。

10.3.1.12. 独自開始ページ

Microchip Studioの開始ページはMicrochip Studioツール ウィンドウで走行するWindows提案基礎(WPF:Windows Presentation Foundation)拡張可能応用タグ付け言語(XAML:Extensible Application Markup Language)ページです。開始ページ ツール ウィンドウはMicrochip Studio内部命令を走らせることができます。Microchip Studioが開始すると、現在の既定開始ページを開きます。第三者の開始ページをインストールした場合、**Options**(任意選択)ダイアログ枠を用いることによってそのページを既定として設定することができます。

独自開始ページのインストールと適用

拡張管理部(Extension Manager)のオンライン陳列室部を使うことによって独自開始ページをインストールすることができます。独自開始ページを含む.vsixファイルの位置を突き止めて開くことによってウェブ サイトまたはローカルのイントラネットのページから直接、または開始ページを複製してそれらをコンピュータのDocuments¥Atmel Studio¥7.0¥StartPages¥フォルダに貼り付けることによってインストールすることもできます。

Options(任意選択)ダイアログ枠でそれを選ぶことによって独自開発ページを適用することができます。拡張管理部によってインストールした開始ページは拡張名[InstalledExtension]としてCustomize Start Page(独自化した開始ページ)一覧に現れます。¥StartPagesフォルダに投下された開始ページは以下の例で示されるように、一覧登録で部分的なファイル パスを含みます。

Documents¥Atmel Studio¥7.0¥StartPages¥StartPage.xaml

独自開始ページを適用

1. **Tools**(ツール)メニューで**Options**(任意選択)をクリックしてください。
2. **Options**(任意選択)ダイアログ枠の左側で**Environment**(環境)節点を展開し、その後に**Startup**(始動)をクリックしてください。
3. **Customize Start Page**(開始ページ独自化)一覧で望む開始ページを選んでください。
4. この一覧は使用者開始ページ フォルダとインストールされた開始ページ型の拡張の全ての.xamlファイルを含みます。
5. **OK**をクリックしてください。

10.3.2. プロジェクト任意選択**10.3.2.1. General Settings (全般設定)**

Microchip Studioプロジェクト フォルダの既定パスを設定し、開発して構築されるプロジェクトとして**Output**(出力)ウィンドウ、**Task List**(タスク一覧)、**Solution Explorer**(解決策エクスプローラ)の既定の動きを決めます。このダイアログ枠にアクセスするには**Tools**(ツール)メニューで**Options**(任意選択)をクリックし、**Projects and Solutions**(プロジェクトと解決策)を展開し、**General**(全般)をクリックしてください。

注: ダイアログ枠で利用可能な任意選択とあなたが見るメニュー命令の名前と位置はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。このヘルプ ページは全般的な開発設定を念頭に置いて書かれました。設定を見るまたは変更するには**Tools**(ツール)メニューで**Import and Export Settings**(設定のインポートとエクスポート)を選んでください。

Projects location (プロジェクト位置)

新しいプロジェクトと解決策のフォルダとディレクトリが作成される既定位置を設定します。いくつかのダイアログ枠もフォルダ開始点用にこの任意選択で設定される位置を使います。**Open Project**(プロジェクトを開く)ダイアログ枠は**My Project**(私のプロジェクト)ショートカットに対してこの位置を使います。

User project templates location (使用者プロジェクト雛形位置)

My Templates(私の雛形)の一覧を作成するために**New Project**(新しいプロジェクト)ダイアログ枠によって使われる既定位置を設定します。

User item templates location (使用者項目雛形位置)

My Templates(私の雛形)の一覧を作成するために**Add New Item**(新しい項目を追加)ダイアログ枠によって使われる既定位置を設定します。

Always show Error List if build finishes with errors (構築異常終了で常に異常一覧を表示)

プロジェクトが構築を失敗した場合にだけ、構築完了で**Error List**(異常一覧)ウィンドウを開きます。構築処理中に起きた異常が表示されます。この任意選択が解除されると、構築が完了する時に異常が未だ起きていても、このウィンドウは開きません。この任意選択は既定によって許可されます。

Track Active Item in Solution Explorer (解決策エクスプローラで活性化項目を探知)

選択時、**Solution Explorer**(解決策エクスプローラ)が自動的に開いて活性化項目が選択されます。あなたがプロジェクトまたは解決策で違うファイルで、または設計部で違う構成部品で作業すると、選択された項目が変わります。この任意選択が解除されると、解決策エクスプローラでの選択は自動的に変更しません。この任意選択は既定によって許可されます。

Show advanced build configurations (高度な構築構成設定を表示)

選択時、**Project Property Pages**(プロジェクトプロパティページ)ダイアログ枠と**Solution Property Pages**(解決策プロパティページ)ダイアログ枠で構築構成設定任意選択が現れます。解除時、**Project Property Pages**ダイアログ枠と1つの構成設定、またはデバッグ(Debug)と公開(Release)の2つの構成設定を含むプロジェクトに対する**Solution Property Pages**ダイアログ枠で構築構成設定任意選択が現れません。プロジェクトがユーザー定義構成設定を持つ場合、構築構成設定任意選択が表示されます。

非選択時、**Build**(構築)メニュー上の**Build Solution**(解決策構築)、**Rebuild Solution**(解決策再構築)、**Clean Solution**(解決策解消)のような命令は公開(Release)構成設定で実行され、**Debug**(デバッグ)メニュー上の**Start Debugging**(デバッグ開始)、**Start Without Debugging**(デバッグなしで開始)のような命令はデバッグ(Debug)構成設定で実行されます。

Always show solution (常に解決策を表示)

選択時、解決策と解決策で動作する全ての命令が常にIDEで表示されます。解除時、全てのプロジェクトが独立型プロジェクトとして作成され、解決策が1つのプロジェクトだけを含む場合、IDEに於いて**Solution Explorer**(解決策エクスプローラ)での解決策や解決策で動作する命令を見れません。

Save new projects when created (作成時に新しいプロジェクトを保存)

選択時、**New Project**(新しいプロジェクト)ダイアログ枠でプロジェクト用の位置を指定することができます。解除時、全ての新しいプロジェクトは一時プロジェクトとして作成されます。一時プロジェクトで作業すると、ディスク位置を指定することなくプロジェクトを作成して実験することができます。

Warn user when the project location is not trusted (プロジェクト位置が信用されない時に使用者に警告)

完全に信用されない場所(例えば、UNCパスやHTTPパス上)で新しいプロジェクトを作成しようとしたり、または既存プロジェクトを開こうとした場合、メッセージが表示されます。完全に信用されない場所でプロジェクトを作成または開こうとする度毎にメッセージが表示されるかどうかを指定するにはこの任意選択を使ってください。

Show Output window when build starts (構築開始時に出力ウィンドウを表示)

解決策構築の初めに於いてIDEで自動的に**Output**(出力)ウィンドウを表示します。

Prompt for symbolic renaming when renaming files (ファイル改名時に象徴的改名のための指示を求める)

選択時、Microchip Studioがコード要素に対してプロジェクト内の全ての参照も改名するか否かを問うメッセージ枠を表示します。

10.3.2.2. Build and Run (構築して走行) 任意選択

プロジェクトまたはその解決策が構築される時に変更したファイルが自動的に保存されるかどうか、同時に構築することができる最大Visual C++プロジェクト数、**Run**(走行)での或る既定の動きを決めます。このダイアログ枠にアクセスするには**Tools**(ツール)メニューで**Options**(任意選択)をクリックし、**Projects and Solutions**(プロジェクトと解決策)をクリックし、その後に**Build and Run**(構築して走行)をクリックしてください。

Save all changes (全ての変更を保存)

F5押下または**Debug**(デバッグ)メニューで**Start**(開始)または**Build**(構築)メニューで**Build**(構築)をクリックする時に、最後に構築してから変更された解決策ファイルと全てのプロジェクトファイルへの変更を自動的に保存します。指示要求表示は全く与えられません。項目はそれらの現在の名前で保存されます。既定によってこの任意選択は許可されます。

Save changes to open documents only (開いた文書だけ変更を保存)

F5押下または**Debug**(デバッグ)メニューで**Start**(開始)または**Build**(構築)メニューで**Build**(構築)をクリックする時に、開いた全ての文書への変更を自動的に保存します。指示要求表示は全く与えられません。

Prompt to save all changes (全ての変更を保存ための指示要求表示)

選択時、**F5**押下または**Debug**(デバッグ)メニューで**Start**(開始)または**Build**(構築)メニューで**Build**(構築)をクリックする時に、解決策とプロジェクト項目への変更を保存したいかどうかを問うダイアログ枠を表示します。プロジェクトに名前と場所を割り当てることができるように**Save As**(〜として保存)ダイアログ枠が表示されます。この任意選択が選ばれない場合、変更を含むメモリイメージを使って走行しますが、その変更は保存されません。

Don't save any changes (どの変更も保存しない)

プロジェクト走行時、統合開発環境(IDE)は開いた文章内のコード版を走らせ、開いた文書への変更を保存しません。

Maximum number of parallel project builds (最大並行構築プロジェクト数)

同時に構築することができる最大プロジェクト数を指定します。構築処理を最適化するため、最大並列構築プロジェクト数は自動的にコンピュータのCPU数に設定されます。最大は32です。

Only build start-up projects and dependencies on Run (始動プロジェクトと走行での依存性だけ構築)

選択時、**F5**押下または**Debug**(デバッグ)メニューで**Start**(開始)または**Build**(構築)メニューで**Build**(構築)でのクリックは始動プロジェクトとその依存性だけを構築します。この任意選択が解除されると、**F5**押下は全てのプロジェクト、依存性、解決策ファイルを構築します。既定によってこの任意選択は解除されます。

Always build (常に構築)

メッセージ枠が表示されずに古いプロジェクト構成設定が構築されます。この任意選択は **Do not show this dialog again in the message** (メッセージでこのダイアログを再び表示しない) を選び、その後 **Yes** (はい) をクリックした時に設定されます。

Never build (決して構築しない)

メッセージ枠が表示されずに古いプロジェクト構成設定は構築されません。この任意選択は **Do not show this dialog again in the message** (メッセージでこのダイアログを再び表示しない) を選び、その後 **No** (いいえ) をクリックした時に設定されます。

Prompt to build (構築に対して指示要求表示)

プロジェクト構成設定が古い度毎にメッセージ枠を表示します。

Prompt to launch (発生に対して指示要求表示)

構築異常発生の際毎にメッセージ枠を表示します。

Do not launch (開始しない)

メッセージ枠が表示されずに応用は開始されません。この任意選択は **Do not show this dialog again in the message** (メッセージでこのダイアログを再び表示しない) を選び、その後 **No** (いいえ) をクリックした時に設定されます。

Launch old version (古い版を開始)

メッセージ枠が表示されずに応用の新しい版が開始されません。この任意選択は **Do not show this dialog again in the message** (メッセージでこのダイアログを再び表示しない) を選び、その後 **Yes** (はい) をクリックした時に設定されます。

For new solutions use the currently selected project as the start-up project (新しい解決策に対して始動プロジェクトとして現在選択されたプロジェクトを使用)

選択時、新しい解決策は始動プロジェクトとして現在選択されたプロジェクトを使います。

MSBuild project build output verbosity (MSBuildプロジェクト構築出力冗長性)

構築出力に対する冗長性レベルを設定します。より多くの情報については **MSBuild** コマンド行参考文献で **/verbosity** スイッチをご覧ください。

MSBuild project build log file verbosity (MSBuildプロジェクト構築記録ファイル冗長性)

構築記録ファイルに対する冗長性レベルを設定します。より多くの情報については **MSBuild** コマンド行参考文献で **/verbosity** スイッチをご覧ください。

10.3.3. ソース制御

インストールされた (SVN, ClearCase, Vault, Git などの) ソース制御用プラグインを持つ場合、活性化してソース保管場所と共にプラグインを使うために、この部分で引き落とし一覧からそれを選ぶべきです。

10.3.4. テキスト エディタ任意選択**10.3.4.1. General Settings (全般設定)**

このダイアログ枠は Microchip Studio のコードとテキストのエディタに対する全般設定の変更を許します。このダイアログ枠を表示するには **Tools** (ツール) メニューで **Options** (任意選択) をクリックし、その後 **General** (全般) をクリックしてください。

注: あなたが見るダイアログ枠とメニュー命令はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。設定を変更するには **Tools** (ツール) メニューから **Import and Export Settings** (設定のインポートとエクスポート) を選んでください。

Settings (設定)**Drag and drop text editing (ドラッグ&ドロップ テキスト編集)**

選択時、これは現在の文書または他の何れかの開いた文書内の別の位置へマウスでテキストを選択して引き摺る (ドラッグする) ことによってテキストを移動することを許可します。

Automatic delimiter highlighting (自動分離子強調表示)

選択時、一致する中括弧だけでなく、パラメータや項目値対を分離する分離子文字も強調表示されます。

Track changes (変更経緯)

選択時、コード エディタの選択余白は最近変更されたコードを記すために黄色の垂直線を、変更されないコードの次に緑色の垂直線を表示します。

Auto-detect UTF-8 encoding without signature (署名なしで UTF-8 符号化を自動検出)

既定によって、エディタはバイト順の記号や文字セットの札を検索することによって符号化を検出します。現在の文書でどちらも見つけられない場合、コード エディタはバイトの流れを走査することによって UTF-8 符号化自動検出を試みます。符号化の自動検出を禁止するにはこの任意選択を解除してください。

Display (表示)**Selection margin (選択余白)**

選択時、エディタのテキスト領域の左端に沿って垂直余白が表示されます。テキストの行全体を選択するのにこの余白をクリックしたり、テキストの連続する行を選択するのにクリックと引き摺り (ドラッグ) をすることができます。

Indicator margin (指示部余白)

選択時、エディタのテキスト領域の左端の外側に垂直余白が表示されます。この余白をクリックすると、テキストに関連するアイコンと情報(Tooltip)が表示されます。例えば、中断点やタスク一覧ショートカットが指示部余白に現れます。指示部余白情報は印刷されません。

Vertical scroll bar (垂直スクロールバー)

選択時、エディタの可視領域外になる構成部分を見るために上下スクロールを許す垂直スクロールバーが表示されます。垂直スクロールバーが利用できない場合、スクロールするためにPage Up、Page Down、それとカーソルキーを使うことができます。

Horizontal scroll bar (水平スクロールバー)

選択時、エディタの可視領域外になる構成部分を見るために左右スクロールを許す水平スクロールバーが表示されます。水平スクロールバーが利用できない場合、スクロールするためにカーソルキーを使うことができます。

10.3.4.2. ファイル拡張と連携

ソース ファイル拡張のツール連携を指定することができます。

10.3.4.3. General Language (言語全般) 任意選択

このダイアログ枠はコード エディタの既定の動き変更を許します。これらの設定はHTML Designer's Source(HTML設計者のソース)表示部のようなコード エディタに基づく他のエディタにも適用します。このダイアログ枠を開くにはTools(ツール)メニューからOptions(任意選択)を選んでください。Text Editor(テキスト エディタ)フォルダ内でAll Languages(全言語)副フォルダを展開してからGeneral(全般)を選んでください。

注意 このページは全ての開発言語に対する既定任意選択を設定します。このダイアログ内の任意選択をリセットすることはここでどんな選択が選ばれても全ての言語でGeneral(全般)任意選択をリセットすることを覚えて置いてください。単に1つの言語についてテキスト エディタ任意選択を変更するにはその言語用の副フォルダを展開してその任意選択ページを選んでください。

任意選択がいくつかのプログラミング言語に対するGeneral options(全般任意選択)ページで選択されているけれども他ではされていない時に灰色のチェック記号が表示されます。

注: あなたが見るダイアログ枠とメニュー命令はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。設定を変更するにはTools(ツール)メニューからImport and Export Settings(設定のインポートとエクスポート)を選んでください。

Statement Completion (文完成)**Auto list members (自動メンバー一覧)**

選択時、エディタで入力する時にインテリセンス(intellisense)によって利用可能なメンバ、プロパティ、値、メソッドのポップアップ一覧が表示されます。コードに項目を挿入するにはポップアップ一覧から何れかの項目を選んでください。この任意選択を選ぶことはHide advanced members(老けたメンバを隠す)任意選択を許可します。

Hide advanced members (老けたメンバを隠す)

選択時、最も共通的に使われたそれらの項目だけを表示することによってポップアップ文完成一覧を短くします。他の項目はこの一覧からふり落とされます。

Parameter information (パラメータ情報)

選択時、その利用可能なパラメータの全てと共にエディタ内の挿入点下に現在の宣言または手続きに対する完全な構文が表示されます。割り当てることができる次のパラメータは太字で表示されます。

Settings (設定)**Enable virtual space (仮想空間許可)**

この任意選択が選ばれてWord wrap(ワードラップ)が解除されると、コード エディタ内の行末の向こうの何処かをクリックして入力することができます。この機能はコードの次の一貫性のある地点に注釈を配置するのに使うことができます。

Word wrap (ワードラップ)

選択時、見ることができるエディタ領域を超えて水平方向に延びる行のどの部分も自動的に次の行で表示されます。この任意選択を選ぶことはShow visual glyphs for word wrap(ワードラップ用視覚記号を表示)任意選択を許可します。

注: ワードラップがONの間、仮想空間(Virtual Space)機能はOFFにされます。

Show visual glyphs for word wrap (ワードラップ用視覚記号を表示)

選択時、次行に丸められた長い行の場所で戻り矢印指示子が表示されます。

これらの指示子を表示することを好まない場合、この任意選択を解除してください。

注: これらの注意の矢印はコードに追加されず、印刷もされません。これらは参照用だけです。


Apply Cut or Copy commands to blank lines when there is no selection (選択なし時、空白行に切り取りまたは複製命令を適用)

この任意選択は空白行に挿入点を置いて何も選択せず、その後に複製または切り取りをした時のエディタの動きを設定します。

この任意選択が選ばれると、空白行は複製または切り取られます。その後に張り付ける場合、新しい空白行が挿入されます。

この任意選択が解除されると、切り取り命令は空白行を取り除きます。けれども、クリップボードのデータは保存されます。従って、その後に張り付け命令を使ったなら、クリップボード上に複製された最も最新の内容が貼り付けられます。前に何も複製されていない場合、何も貼り付けられません。

この設定は行が空白でない時に複写や切り取りに影響を及ぼしません。何も選択されない場合、行全体が複写または切り取られません。その後には張り付けをした場合、行全体のテキストとその行終了文字が貼り付けされます。

 **助言:** 空白、タブ、行終了用の指示子を表示し、故に全体的な空白行から字下げした行を識別するには、**Edit(編集)**メニューから**Advanced(高度)**を選んで**View White Space(白空間を表示)**を選んでください。

Display (表示)

Line numbers (行番号)

選択時、行番号がコードの各行の傍に現れます。

注: これらの行番号はコードに追加されず、印刷もされません。これらは参照用だけです。

Enable single-click URL navigation (1クリックURL誘導許可)


選択時、マウスカーソルがエディタ内のURL上を通る時に指さしに変わります。ウェブブラウザで示されるページを表示するのにURLをクリックすることができます。

Navigation bar (誘導バー)

選択時、コードエディタの上部に誘導バーが表示されます。その**Objects(オブジェクト)**と**Members(メンバ)**の引き落とし一覧はコード内の特定のオブジェクトを選び、そのメンバから選び、コードエディタに於いて選んだメンバの宣言へ誘導します。

10.3.4.4. Tabs (タブ) ダイアログ

このダイアログ枠はコードエディタの既定の動き変更を許します。これらの設定は**HTML Designer's Source(HTML設計者のソース)**表示部のようなコードエディタに基づく他のエディタにも適用します。これらの任意選択を表示するには**Tools(ツール)**メニューから**Options(任意選択)**を選んでください。**Text Editor(テキストエディタ)**フォルダ内で**All Languages(全言語)**副フォルダを展開してから**Tabs(タブ)**を選んでください。

 **注意** このページは全ての開発言語に対する既定任意選択を設定します。このダイアログ内の任意選択をリセットすることはここでどんな選択が選ばれても全ての言語で**Tabs(タブ)**任意選択をリセットすることを覚えて置いてください。単に1つの言語についてテキストエディタを変更するにはその言語用の副フォルダを展開してその任意選択ページを選んでください。

特定のプログラミング言語に対して**Tabs(タブ)**任意選択ページで違う設定が選ばれた場合、違う字下げに対して**"The indentation settings for individual text formats conflict with each other,(個別テキスト形式に対する字下げ設定がお互いに矛盾)"**メッセージと違う**タブ**任意選択に対して**"The tab settings for individual text formats conflict with each other,(個別テキスト形式に対するタブ設定がお互いに矛盾)"**メッセージが表示されます。

注: あなたが見るダイアログ枠とメニュー命令はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。設定を変更するには**Tools(ツール)**メニューから**Import and Export Settings(設定のインポートとエクスポート)**を選んでください。

Indenting (字下げ)

None (なし)

選択時、新しい行は字下げされません。挿入点は新しい行の最初の列に置かれます。

Block (塊)

選択時、新しい行は自動的に字下げされます。挿入点は先行する行と同じ開始点に置かれます。

Smart (聡明)

選択時、新しい行は他のコード形式設定と開発言語用のインテリセンス(Intellisense)変換によって、コード状況に合うように位置付けされます。この任意選択は全ての開発言語に対して利用可能ではありません。

例えば、開く中括弧({)と閉じる中括弧(})の間に内包される行は整列された中括弧の位置から追加のタブ位置で自動的に字下げされるかもしれません。

Tab and indent size (タブと字下げの量)

タブ位置間と自動字下げに対する空白での距離を設定します。既定は4つの空白です。指定した量を満たすためにタブ文字、空白文字、またはその両方が挿入されます。

Insert spaces (空白挿入)

選択時、字下げ操作はタブ文字ではなく、空白文字だけを挿入します。例えば、タブと字下げの量が5に設定される場合、**Tab**キーまたは**Formatting(書式設定)**ツールバーで**Increase Indent(字下げ増加)**鈕を押す時には必ず5つの空白文字が挿入されます。

Keep tabs (タブ保持)

選択時、各字下げ操作は1つのタブ文字を挿入します。

10.3.4.5. AVR[®] Assembler Language-specific Settings (AVRアセンブラ言語特有設定)

10.3.4.5.1. General Language (言語全般) 任意選択

このダイアログ枠はコードエディタの既定の動き変更を許します。これらの設定は**HTML Designer's Source(HTML設計者のソース)**表示部のようなコードエディタに基づく他のエディタにも適用します。このダイアログ枠を開くには**Tools(ツール)**メニューから**Options(任意選択)**を選んでください。**Text Editor(テキストエディタ)**フォルダ内で**All Languages(全言語)**副フォルダを展開してから**General(全般)**を選んでください。

注意 このページは全ての開発言語に対する既定任意選択を設定します。このダイアログ内の任意選択をリセットすることはここでどんな選択が選ばれても全ての言語で**General**(全般)任意選択をリセットすることを覚えて置いてください。単に1つの言語についてテキスト エディタ任意選択を変更するにはその言語用の副フォルダを展開してその任意選択ページを選んでください。

任意選択がいくつかのプログラミング言語に対する**General options**(全般任意選択)ページで選択されているけれども他ではされていない時に灰色のチェック記号が表示されます。

注: あなたが見るダイアログ枠とメニュー命令はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。設定を変更するには**Tools**(ツール)メニューから**Import and Export Settings**(インポートとエクスポートの設定)を選んでください。

Statement Completion (文完成)

Auto list members (自動メンバー一覧)

選択時、エディタで入力する時にインテリセンス(intellisense)によって利用可能なメンバ、プロパティ、値、メソッドのポップアップ一覧が表示されます。コードに項目を挿入するにはポップアップ一覧から何れかの項目を選んでください。この任意選択を選ぶことは**Hide advanced members**(老けたメンバを隠す)任意選択を許可します。

Hide advanced members (老けたメンバを隠す)

選択時、最も共通的に使われたそれらの項目だけを表示することによってポップアップ文完成一覧を短くします。他の項目はこの一覧からふるい落とされます。

Parameter information (パラメータ情報)

選択時、その利用可能なパラメータの全てと共にエディタ内の挿入点下に現在の宣言または手続きに対する完全な構文が表示されます。割り当てることができる次のパラメータは太字で表示されます。

Settings (設定)

Enable virtual space (仮想空間許可)

この任意選択が選ばれて**Word wrap**(ワードラップ)が解除されると、コード エディタ内の行末の向こうの何処かをクリックして入力することができます。この機能はコードの次の一貫性のある地点に注釈を配置するのに使うことができます。

Word wrap (ワードラップ)

選択時、見ることができるエディタ領域を超えて水平方向に延びる行のどの部分も自動的に次の行で表示されます。この任意選択を選ぶことは**Show visual glyphs for word wrap**(ワードラップ用視覚記号を表示)任意選択を許可します。

注: ワードラップがONの間、仮想空間(Virtual Space)機能はOFFにされます。

Show visual glyphs for word wrap (ワードラップ用視覚記号を表示)

選択時、次行に丸められた長い行の場所で戻り矢印指示子が表示されます。

これらの指示子を表示することを好まない場合、この任意選択を解除してください。

注: これらの注意の矢印はコードに追加されず、印刷もされません。これらは参考用だけです。

Apply Cut or Copy commands to blank lines when there is no selection (選択なし時、空白行に切り取りまたは複製命令を適用)

この任意選択は空白行に挿入点を置いて何も選択せず、その後に複製または切り取りをした時のエディタの動きを設定します。

この任意選択が選ばれると、空白行は複製または切り取られます。その後に張り付ける場合、新しい空白行が挿入されます。

この任意選択が解除されると、切り取り命令は空白行を取り除きます。けれども、クリップボードのデータは保存されます。従って、その後に張り付け命令を使ったなら、クリップボード上に複製された最も最新の内容が貼り付けられます。前に何も複製されていない場合、何も貼り付けられません。

この設定は行が空白でない時に複製や切り取りに影響を及ぼしません。何も選択されない場合、行全体が複製または切り取られます。その後に張り付けをした場合、行全体のテキストとその行終了文字が貼り付けられます。



助言: 空白、タブ、行終了用の指示子を表示し、故に全体的な空白行から字下げした行を識別するには、**Edit**(編集)メニューから**Advanced**(高度)を選んで**View White Space**(白空間を表示)を選んでください。

Display (表示)

Line numbers (行番号)

選択時、行番号がコードの各行の傍に現れます。

注: これらの行番号はコードに追加されず、印刷もされません。これらは参照用だけです。

Enable single-click URL navigation (1クリックURL誘導許可)

選択時、マウスカーソルがエディタ内のURL上を通る時に指さしに変わります。ウェブブラウザで示されるページを表示するのにURLをクリックすることができます。

Navigation bar (誘導バー)

選択時、コード エディタの上部に誘導バーが表示されます。その**Objects**(オブジェクト)と**Members**(メンバ)の引き落とし一覧はコード内の特定のオブジェクトを選び、そのメンバから選び、コード エディタに於いて選んだメンバの宣言へ誘導します。

10.3.4.5.2. Tabs (タブ) ダイアログ

このダイアログ枠はコード エディタの既定の動き変更を許します。これらの設定はHTML Designer's Source(HTML設計者のソース)表示部のようなコード エディタに基づく他のエディタにも適用します。これらの任意選択を表示するにはTools(ツール)メニューからOptions(任意選択)を選んでください。Text Editor(テキスト エディタ)フォルダ内でAll Languages(全言語)副フォルダを展開してからTabs(タブ)を選んでください。

注意 このページは全ての開発言語に対する既定任意選択を設定します。このダイアログ内の任意選択をリセットすることはここでどんな選択が選ばれても全ての言語でTabs(タブ)任意選択をリセットすることを覚えて置いてください。単に1つの言語についてテキスト エディタを変更するにはその言語用の副フォルダを展開してその任意選択ページを選んでください。

特定のプログラミング言語に対してTabs(タブ)任意選択ページで違う設定が選ばれた場合、違う字下げに対して”The indentation settings for individual text formats conflict with each other,(個別テキスト形式に対する字下げ設定がお互いに矛盾)”メッセージと違うタブ任意選択に対して”The tab settings for individual text formats conflict with each other,(個別テキスト形式に対するタブ設定がお互いに矛盾)”メッセージが表示されます。

注: あなたが見るダイアログ枠とメニュー命令はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。設定を変更するにはTools(ツール)メニューからImport and Export Settings(設定のインポートとエクスポート)を選んでください。

Indenting (字下げ)

None (なし)

選択時、新しい行は字下げされません。挿入点は新しい行の最初の列に置かれます。

Block (塊)

選択時、新しい行は自動的に字下げされます。挿入点は先行する行と同じ開始点に置かれます。

Smart (聡明)

選択時、新しい行は他のコード形式設定と開発言語用のインテリセンス(Intellisense)変換によって、コード状況に合うように位置付けされます。この任意選択は全ての開発言語に対して利用可能ではありません。

例えば、開く中括弧({)と閉じる中括弧(})の間に内包される行は整列された中括弧の位置から追加のタブ位置で自動的に字下げされるかもしれません。

Tab and indent size (タブと字下げの量)

タブ位置間と自動字下げに対する空白での距離を設定します。既定は4つの空白です。指定した量を満たすためにタブ文字、空白文字、またはその両方が挿入されます。

Insert spaces (空白挿入)

選択時、字下げ操作はタブ文字ではなく、空白文字だけを挿入します。例えば、タブと字下げの量が5に設定される場合、TabキーまたはFormatting(書式設定)ツールバーでIncrease Indent(字下げ増加)鈕を押す時には必ず5つの空白文字が挿入されます。

Keep tabs (タブ保持)

選択時、各字下げ操作は1つのタブ文字を挿入します。

10.3.4.6. AVR® GCC Language-specific Settings (AVR GCC言語特有設定)

10.3.4.6.1. General Language (言語全般) 任意選択

このダイアログ枠はコード エディタの既定の動き変更を許します。これらの設定はHTML Designer's Source(HTML設計者のソース)表示部のようなコード エディタに基づく他のエディタにも適用します。このダイアログ枠を開くにはTools(ツール)メニューからOptions(任意選択)を選んでください。Text Editor(テキスト エディタ)フォルダ内でAll Languages(全言語)副フォルダを展開してからGeneral(全般)を選んでください。

注意 このページは全ての開発言語に対する既定任意選択を設定します。このダイアログ内の任意選択をリセットすることはここでどんな選択が選ばれても全ての言語でGeneral(全般)任意選択をリセットすることを覚えて置いてください。単に1つの言語についてテキスト エディタ任意選択を変更するにはその言語用の副フォルダを展開してその任意選択ページを選んでください。

任意選択がいくつかのプログラミング言語に対するGeneral options(全般任意選択)ページで選択されているけれども他ではされていない時に灰色のチェック記号が表示されます。

注: あなたが見るダイアログ枠とメニュー命令はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。設定を変更するにはTools(ツール)メニューからImport and Export Settings(設定のインポートとエクスポート)を選んでください。

Statement Completion (文完成)

Auto list members (自動メンバー一覧)

選択時、エディタで入力する時にインテリセンス(Intellisense)によって利用可能なメンバー、プロパティ、値、メソッドのポップアップ一覧が表示されます。コードに項目を挿入するにはポップアップ一覧から何れかの項目を選んでください。この任意選択を選ぶことはHide advanced members(老けたメンバーを隠す)任意選択を許可します。

Hide advanced members (老けたメンバーを隠す)

選択時、最も共通的に使われたそれらの項目だけを表示することによってポップアップ文完成一覧を短くします。他の項目はこの一覧からふるい落とされます。

Parameter information (パラメータ情報)

選択時、その利用可能なパラメータの全てと共にエディタ内の挿入点下に現在の宣言または手続きに対する完全な構文が表示されます。割り当てることができる次のパラメータは太字で表示されます。

Settings (設定)**Enable virtual space (仮想空間許可)**

この任意選択が選ばれて**Word wrap**(ワードラップ)が解除されると、コード エディタ内の行末の向こうの何処かをクリックして入力することができます。この機能はコードの次の一貫性のある地点に注釈を配置するのに使うことができます。

Word wrap (ワードラップ)

選択時、見ることができるエディタ領域を超えて水平方向に延びる行のどの部分も自動的に次の行で表示されます。この任意選択を選ぶことは**Show visual glyphs for word wrap**(ワードラップ用視覚記号を表示)任意選択を許可します。

注: ワードラップがONの間、仮想空間(Virtual Space)機能はOFFにされます。

Show visual glyphs for word wrap (ワードラップ用視覚記号を表示)

選択時、次行に丸められた長い行の場所で戻り矢印指示子が表示されます。

これらの指示子を表示することを好まない場合、この任意選択を解除してください。

注: これらの注意の矢印はコードに追加されず、印刷もされません。これらは参考用だけです。

Apply Cut or Copy commands to blank lines when there is no selection (選択なし時、空白行に切り取りまたは複製命令を適用)

この任意選択は空白行に挿入点を置いて何も選択せず、その後に複製または切り取りをした時のエディタの動きを設定します。

この任意選択が選ばれると、空白行は複製または切り取られます。その後に張り付ける場合、新しい空白行が挿入されます。

この任意選択が解除されると、切り取り命令は空白行を取り除きます。けれども、クリップボードのデータは保存されます。従って、その後に張り付け命令を使ったなら、クリップボード上に複製された最も最新の内容が貼り付けられます。前に何も複製されていない場合、何も貼り付けられません。

この設定は行が空白でない時に複製や切り取りに何の効果もありません。何も選択されない場合、行全体が複製または切り取られます。その後に張り付けをした場合、行全体のテキストとその行終了文字が貼り付けられます。



助言: 空白、タブ、行終了用の指示子を表示し、故に全体的な空白行から字下げした行を識別するには、**Edit**(編集)メニューから**Advanced**(高度)を選んで**View White Space**(白空間を表示)を選んでください。

Display (表示)**Line numbers (行番号)**

選択時、行番号がコードの各行の傍に現れます。

注: これらの行番号はコードに追加されず、印刷もされません。これらは参照用だけです。

Enable single-click URL navigation (1クリックURL誘導許可)

選択時、マウスカーソルがエディタ内のURL上を通る時に指しに変わります。ウェブブラウザで示されるページを表示するのにURLをクリックすることができます。

Navigation bar (誘導バー)

選択時、コード エディタの上部に誘導バーが表示されます。その**Objects**(オブジェクト)と**Members**(メンバ)の引き落とし一覧はコード内の特定のオブジェクトを選び、そのメンバから選び、コード エディタに於いて選んだメンバの宣言へ誘導します。

10.3.4.6.2. Tabs (タブ) ダイアログ

このダイアログ枠はコード エディタの既定の動き変更を許します。これらの設定は**HTML Designer's Source**(HTML設計者のソース)表示部のようなコード エディタに基づく他のエディタにも適用します。これらの任意選択を表示するには**Tools**(ツール)メニューから**Options**(任意選択)を選んでください。**Text Editor**(テキスト エディタ)フォルダ内で**All Languages**(全言語)副フォルダを展開してから**Tabs**(タブ)を選んでください。



注意 このページは全ての開発言語に対する既定任意選択を設定します。このダイアログ内の任意選択をリセットすることはここでどんな選択が選ばれても全ての言語で**Tabs**(タブ)任意選択をリセットすることを覚えて置いてください。単に1つの言語についてテキスト エディタを変更するにはその言語用の副フォルダを展開してその任意選択ページを選んでください。

特定のプログラミング言語に対して**Tabs**(タブ)任意選択ページで違う設定が選ばれた場合、違う字下げに対して”**The indentation settings for individual text formats conflict with each other**, (個別テキスト形式に対する字下げ設定がお互いに矛盾)”メッセージと違う**タブ**任意選択に対して”**The tab settings for individual text formats conflict with each other**, (個別テキスト形式に対する**タブ**設定がお互いに矛盾)”メッセージが表示されます。

注: あなたが見るダイアログ枠とメニュー命令はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。設定を変更するには**Tools**(ツール)メニューから**Import and Export Settings**(設定のインポートとエクスポート)を選んでください。

Indenting (字下げ)**None (なし)**

選択時、新しい行は字下げされません。挿入点は新しい行の最初の列に置かれます。

Block (塊)

選択時、新しい行は自動的に字下げされます。挿入点は先行する行と同じ開始点に置かれます。

Smart (聡明)

選択時、新しい行は他のコード形式設定と開発言語用のインテリセンス(Intellisense)変換によって、コード状況に合うように位置付けされます。この任意選択は全ての開発言語に対して利用可能ではありません。

例えば、開く中括弧({)と閉じる中括弧(})の間に内包される行は整列された中括弧の位置から追加のタブ位置で自動的に字下げされるかもしれません。

Tab and indent size (タブと字下げの量)

タブ位置間と自動字下げに対する空白での距離を設定します。既定は4つの空白です。指定した量を満たすためにタブ文字、空白文字、またはその両方が挿入されます。

Insert spaces (空白挿入)

選択時、字下げ操作はタブ文字ではなく、空白文字だけを挿入します。例えば、タブと字下げの量が5に設定される場合、TabキーまたはFormatting(書式設定)ツールバーでIncrease Indent(字下げ増加)鈕を押す時には必ず5つの空白文字が挿入されます。


Keep tabs (タブ保持)

選択時、各字下げ操作は1つのタブ文字を挿入します。

10.3.4.7. Plain Text Settings (平文設定)

10.3.4.7.1. General Language (言語全般) 任意選択

このダイアログ枠はコードエディタの既定の動き変更を許します。これらの設定はHTML Designer's Source(HTML設計者のソース)表示部のようなコードエディタに基づく他のエディタにも適用します。このダイアログ枠を開くにはTools(ツール)メニューからOptions(任意選択)を選んでください。Text Editor(テキストエディタ)フォルダ内でAll Languages(全言語)副フォルダを展開してからGeneral(全般)を選んでください。

 **注意** このページは全ての開発言語に対する既定任意選択を設定します。このダイアログ内の任意選択をリセットすることはここでどんな選択が選ばれても全ての言語でGeneral(全般)任意選択をリセットすることを覚えて置いてください。単に1つの言語についてテキストエディタ任意選択を変更するにはその言語用の副フォルダを展開してその任意選択ページを選んでください。

任意選択がいくつかのプログラミング言語に対するGeneral options(全般任意選択)ページで選択されているけれども他ではされていない時に灰色のチェック記号が表示されます。

注: あなたが見るダイアログ枠とメニュー命令はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。設定を変更するにはTools(ツール)メニューからImport and Export Settings(設定のインポートとエクスポート)を選んでください。

Statement Completion (文完成)

Auto list members (自動メンバー一覧)

選択時、エディタで入力する時にインテリセンス(intellisense)によって利用可能なメンバー、プロパティ、値、メソッドのポップアップ一覧が表示されます。コードに項目を挿入するにはポップアップ一覧から何れかの項目を選んでください。この任意選択を選ぶことはHide advanced members(老けたメンバーを隠す)任意選択を許可します。

Hide advanced members (老けたメンバーを隠す)

選択時、最も共通的に使われたそれらの項目だけを表示することによってポップアップ文完成一覧を短くします。他の項目はこの一覧からふり落とされます。

Parameter information (パラメータ情報)

選択時、その利用可能なパラメータの全てと共にエディタ内の挿入点下に現在の宣言または手続きに対する完全な構文が表示されます。割り当てることができる次のパラメータは太字で表示されます。

Settings (設定)

Enable virtual space (仮想空間許可)

この任意選択が選ばれてWord wrap(ワードラップ)が解除されると、コードエディタ内の行末の向こうの何処かをクリックして入力することができます。この機能はコードの次の一貫性のある地点に注釈を配置するのに使うことができます。

Word wrap (ワードラップ)

選択時、見ることができるエディタ領域を超えて水平方向に延びる行のどの部分も自動的に次の行で表示されます。この任意選択を選ぶことはShow visual glyphs for word wrap(ワードラップ用視覚記号を表示)任意選択を許可します。

注: ワードラップがONの間、仮想空間(Virtual Space)機能はOFFにされます。

Show visual glyphs for word wrap (ワードラップ用視覚記号を表示)

選択時、次行に丸められた長い行の場所で戻り矢印指示子が表示されます。

これらの指示子を表示することを好まない場合、この任意選択を解除してください。

注: これらの注意の矢印はコードに追加されず、印刷もされません。これらは参照用だけです。

Apply Cut or Copy commands to blank lines when there is no selection (選択なし時、空白行に切り取りまたは複製命令を適用)
この任意選択は空白行に挿入点を置いて何も選択せず、その後に複製または切り取りをした時のエディタの動きを設定します。
この任意選択が選ばれると、空白行は複製または切り取られます。その後に張り付ける場合、新しい空白行が挿入されます。
この任意選択が解除されると、切り取り命令は空白行を取り除きます。けれども、クリップボードのデータは保存されます。従って、その後
に張り付け命令を使ったなら、クリップボード上に複製された最も最新の内容が貼り付けられます。前に何も複製されていない場
合、何も貼り付けられません。

この設定は行が空白でない時に複製や切り取りに影響を及ぼしません。何も選択されない場合、行全体が複製または切り取られま
す。その後に張り付けをした場合、行全体のテキストとその行終了文字が貼り付けられます。



助言: 空白、タブ、行終了用の指示子を表示し、故に全体的な空白行から字下げした行を識別するには、**Edit**(編集)メニューか
ら**Advanced**(高度)を選んで**View White Space**(白空間を表示)を選んでください。

Display (表示)

Line numbers (行番号)

選択時、行番号がコードの各行の傍に現れます。

注: これらの行番号はコードに追加されず、印刷もされません。これらは参照用だけです。

Enable single-click URL navigation (1クリックURL誘導許可)

選択時、マウスカーソルがエディタ内のURL上を通る時に指さしに変わります。ウェブブラウザで示されるページを表示するのにURLをクリック
することができます。

Navigation bar (誘導バー)

選択時、コードエディタの上部に誘導バーが表示されます。その**Objects**(オブジェクト)と**Members**(メンバ)の引き落とし一覧はコード内の
特定のオブジェクトを選び、そのメンバから選び、コードエディタに於いて選んだメンバの宣言へ誘導します。

10.3.4.7.2. Tabs (タブ) ダイアログ

このダイアログ枠はコードエディタの既定の動き変更を許します。これらの設定は**HTML Designer's Source**(HTML設計者のソース)表示部
のようなコードエディタに基づく他のエディタにも適用します。これらの任意選択を表示するには**Tools**(ツール)メニューから**Options**(任意選択)
を選んでください。**Text Editor**(テキストエディタ)フォルダ内で**All Languages**(全言語)副フォルダを展開してから**Tabs**(タブ)を選んでください。



注意 このページは全ての開発言語に対する既定任意選択を設定します。このダイアログ内の任意選択をリセットすることはここでど
んな選択が選ばれても全ての言語で**Tabs**(タブ)任意選択をリセットすることを覚えて置いてください。単に1つの言語について
テキストエディタを変更するにはその言語用の副フォルダを展開してその任意選択ページを選んでください。

特定のプログラミング言語に対して**Tabs**(タブ)任意選択ページで違う設定が選ばれた場合、違う字下げに対して**The indentati
on settings for individual text formats conflict with each other**, (個別テキスト形式に対する字下げ設定がお互いに矛盾)メッ
セージと違う**タブ**任意選択に対して**The tab settings for individual text formats conflict with each other**, (個別テキスト形式に
対する**タブ**設定がお互いに矛盾)メッセージが表示されます。

注: あなたが見るダイアログ枠とメニュー命令はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。設定を更
えるには**Tools**(ツール)メニューから**Import and Export Settings**(設定のインポートとエクスポート)を選んでください。

Indenting (字下げ)

None (なし)

選択時、新しい行は字下げされません。挿入点は新しい行の最初の列に置かれます。

Block (塊)

選択時、新しい行は自動的に字下げされます。挿入点は先行する行と同じ開始点に置かれます。

Smart (聡明)

選択時、新しい行は他のコード形式設定と開発言語用のインテリセンス(Intellisense)変換によって、コード状況に合うように位置付けされま
す。この任意選択は全ての開発言語に対して利用可能ではありません。

例えば、開く中括弧({)と閉じる中括弧(})の間に内包される行は整列された中括弧の位置から追加の**タブ**位置で自動的に字下げさ
れるかもしれません。

Tab and indent size (タブと字下げの量)

タブ位置間と自動字下げに対する空白での距離を設定します。既定は4つの空白です。指定した量を満たすために**タブ**文字、空白
文字、またはその両方が挿入されます。

Insert spaces (空白挿入)

選択時、字下げ操作は**タブ**文字ではなく、空白文字だけを挿入します。例えば、**タブ**と字下げの量が5に設定される場合、**Tab**キーま
たは**Formatting**(書式設定)ツールバーで**Increase Indent**(字下げ増加)鈕を押す時には必ず5つの空白文字が挿入されます。

Keep tabs (タブ保持)

選択時、各字下げ操作は1つの**タブ**文字を挿入します。

10.3.4.8. XML Settings (XML設定)

10.3.4.8.1. General Language (言語全般) 任意選択

このダイアログ枠はコード エディタの既定の動き変更を許します。これらの設定はHTML Designer's Source(HTML設計者のソース)表示部のようなコード エディタに基づく他のエディタにも適用します。このダイアログ枠を開くにはTools(ツール)メニューからOptions(任意選択)を選んでください。Text Editor(テキスト エディタ)フォルダ内でAll Languages(全言語)副フォルダを展開してからGeneral(全般)を選んでください。

注意 このページは全ての開発言語に対する既定任意選択を設定します。このダイアログ内の任意選択をリセットすることはここでどんな選択が選ばれても全ての言語でGeneral(全般)任意選択をリセットすることを憶えて置いてください。単に1つの言語についてテキスト エディタ任意選択を変更するにはその言語用の副フォルダを展開してその任意選択ページを選んでください。

任意選択がいくつかのプログラミング言語に対するGeneral options(全般任意選択)ページで選択されているけれども他ではされていない時に灰色のチェック記号が表示されます。

注: あなたが見るダイアログ枠とメニュー命令はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。設定を変更するにはTools(ツール)メニューからImport and Export Settings(設定のインポートとエクスポート)を選んでください。

Statement Completion (文完成)

Auto list members (自動メンバー一覧)

選択時、エディタで入力する時にインテリセンス(intellisense)によって利用可能なメンバ、プロパティ、値、メソッドのポップアップ一覧が表示されます。コードに項目を挿入するにはポップアップ一覧から何れかの項目を選んでください。この任意選択を選ぶことはHide advanced members(老けたメンバを隠す)任意選択を許可します。

Hide advanced members (老けたメンバを隠す)

選択時、最も共通的に使われたそれらの項目だけを表示することによってポップアップ文完成一覧を短くします。他の項目はこの一覧からふり落とされます。

Parameter information (パラメータ情報)

選択時、その利用可能なパラメータの全てと共にエディタ内の挿入点下に現在の宣言または手続きに対する完全な構文が表示されます。割り当てることができる次のパラメータは太字で表示されます。

Settings (設定)

Enable virtual space (仮想空間許可)

この任意選択が選ばれてWord wrap(ワードラップ)が解除されると、コード エディタ内の行末の向こうの何処かをクリックして入力することができます。この機能はコードの次の一貫性のある地点に注釈を配置するのに使うことができます。

Word wrap (ワードラップ)

選択時、見ることができるエディタ領域を超えて水平方向に延びる行のどの部分も自動的に次の行で表示されます。この任意選択を選ぶことはShow visual glyphs for word wrap(ワードラップ用視覚記号を表示)任意選択を許可します。

注: ワードラップがONの間、仮想空間(Virtual Space)機能はOFFにされます。

Show visual glyphs for word wrap (ワードラップ用視覚記号を表示)

選択時、次行に丸められた長い行の場所で戻り矢印指示子が表示されます。

これらの指示子を表示することを好まない場合、この任意選択を解除してください。

注: これらの注意の矢印はコードに追加されず、印刷もされません。これらは参照用だけです。

Apply Cut or Copy commands to blank lines when there is no selection (選択なし時、空白行に切り取りまたは複製命令を適用)

この任意選択は空白行に挿入点を置いて何も選択せず、その後に複製または切り取りをした時のエディタの動きを設定します。

この任意選択が選ばれると、空白行は複製または切り取られます。その後に張り付ける場合、新しい空白行が挿入されます。

この任意選択が解除されると、切り取り命令は空白行を取り除きます。けれども、クリップボードのデータは保存されます。従って、その後に張り付け命令を使ったなら、クリップボード上に複製された最も最新の内容が貼り付けられます。前に何も複製されていない場合、何も貼り付けられません。

この設定は行が空白でない時に複製や切り取りに影響を及ぼしません。何も選択されない場合、行全体が複製または切り取られません。その後に張り付けをした場合、行全体のテキストとその行終了文字が貼り付けられます。



助言: 空白、タブ、行終了用の指示子を表示し、故に全体的な空白行から字下げした行を識別するには、Edit(編集)メニューからAdvanced(高度)を選んでView White Space(白空間を表示)を選んでください。

Display (表示)

Line numbers (行番号)

選択時、行番号がコードの各行の傍に現れます。

注: これらの行番号はコードに追加されず、印刷もされません。これらは参照用だけです。

Enable single-click URL navigation (1クリックURL誘導許可)

選択時、マウスカーソルがエディタ内のURL上を通る時に指しに変わります。ウェブブラウザで示されるページを表示するのにURLをクリックすることができます。

Navigation bar (誘導バー)

選択時、コードエディタの上部に誘導バーが表示されます。その**Objects**(オブジェクト)と**Members**(メンバ)の引き落とし一覧はコード内の特定のオブジェクトを選び、そのメンバから選び、コードエディタに於いて選んだメンバの宣言へ誘導します。

10.3.4.8.2. Tabs (タブ) ダイアログ

このダイアログ枠はコードエディタの既定の動き変更を許します。これらの設定は**HTML Designer's Source**(HTML設計者のソース)表示部のようなコードエディタに基づく他のエディタにも適用します。これらの任意選択を表示するには**Tools**(ツール)メニューから**Options**(任意選択)を選んでください。**Text Editor**(テキストエディタ)フォルダ内で**All Languages**(全言語)副フォルダを展開してから**Tabs**(タブ)を選んでください。

注意 このページは全ての開発言語に対する既定任意選択を設定します。このダイアログ内の任意選択をリセットすることはここでどんな選択が選ばれても全ての言語で**Tabs**(タブ)任意選択をリセットすることを覚えて置いてください。単に1つの言語についてテキストエディタを変更するにはその言語用の副フォルダを展開してその任意選択ページを選んでください。

特定のプログラミング言語に対して**Tabs**(タブ)任意選択ページで違う設定が選ばれた場合、違う字下げに対して**"The indentation settings for individual text formats conflict with each other, (個別テキスト形式に対する字下げ設定がお互いに矛盾)"**メッセージと違うタブ任意選択に対して**"The tab settings for individual text formats conflict with each other, (個別テキスト形式に対するタブ設定がお互いに矛盾)"**メッセージが表示されます。

注: あなたが見るダイアログ枠とメニュー命令はあなたの有効な設定や版に応じてヘルプで記述されるものと違うかもしれません。設定を変更するには**Tools**(ツール)メニューから**Import and Export Settings**(設定のインポートとエクスポート)を選んでください。

Indenting (字下げ)**None (なし)**

選択時、新しい行は字下げされません。挿入点は新しい行の最初の列に置かれます。

Block (塊)

選択時、新しい行は自動的に字下げされます。挿入点は先行する行と同じ開始点に置かれます。

Smart (聡明)

選択時、新しい行は他のコード形式設定と開発言語用のインテリセンス(Intellisense)変換によって、コード状況に合うように位置付けされます。この任意選択は全ての開発言語に対して利用可能ではありません。

例えば、開く中括弧({)と閉じる中括弧(})の間に内包される行は整列された中括弧の位置から追加のタブ位置で自動的に字下げされるかもしれません。

Tab and indent size (タブと字下げの量)

タブ位置間と自動字下げに対する空白での距離を設定します。既定は4つの空白です。指定した量を満たすためにタブ文字、空白文字、またはその両方が挿入されます。

Insert spaces (空白挿入)

選択時、字下げ操作はタブ文字ではなく、空白文字だけを挿入します。例えば、タブと字下げの量が5に設定される場合、Tabキーまたは**Formatting**(書式設定)ツールバーで**Increase Indent**(字下げ増加)鈕を押す時には必ず5つの空白文字が挿入されます。

Keep tabs (タブ保持)

選択時、各字下げ操作は1つのタブ文字を挿入します。

10.3.4.8.3. XML Formatting (XML書式) 任意選択

このダイアログ枠はXMLエディタに対する書式設定の指定を許します。**Tools**(ツール)メニューから**Options**(任意選択)ダイアログ枠にアクセスすることができます。

注: これらの設定は**Options**(任意選択)ダイアログ枠から**Text Editor**(テキストエディタ)フォルダ、**XML**フォルダ、その後に**Formatting**(書式)任意選択を選ぶ時に利用可能です。

Attributes (属性)

手動属性書式設定を防ぎます。属性が書式再設定されません。これは既定です。

注: 属性が複数行の場合、親の要素の字下げと合わせるためにエディタは属性の各行を字下げします。

Align attributes each on their own line (それら自身の行で各々属性を整列)

2つ目とそれに続く属性を垂直方向で先頭行の字下げと合わせるように整列します。以下のXMLテキストは属性がどう整列されるかの例です。

```
<item id = "123-A"
  name = "hammer"
  price = "9.95">
</item>
```

Auto Reformat (自動書式再設定)

On paste from the Clipboard (クリップボードからの貼り付けで)

クリップボードから張り付けられたXMLテキストの書式を再設定します。

On completion of end tag (終了タグの完成で)

終了タグが完成された時に要素の書式を再設定します。

Mixed Content (混合内容)

Preserve mixed content by default (既定によって混ぜられた内容を防止)

エディタが混ぜられた内容を書式再設定するかどうかを決めます。既定によって、内容がxml:space="preserve"範囲で見つかる時を除き、エディタは混ぜられた内容の書式を再設定しようと試みます。

要素がテキストとタグ付けの混合を含む場合、その内容は混ぜられた内容を考慮されます。以下は混ぜられた内容を持つ要素の例です。

```
<dir>c:\¥data¥AlphaProject¥
  <file readOnly="false">test1.txt</file>
  <file readOnly="false">test2.txt</file>
</dir>
```

10.3.4.8.4. XML Miscellaneous (XMLその他) 任意選択

このダイアログ枠はXMLエディタに対して自動完成と枠組みの設定の変更を許します。Tools(ツール)メニューからOptions(任意選択)ダイアログ枠をアクセスすることができます。

注: これらの設定はOptions(任意選択)ダイアログ枠からText Editor(テキスト エディタ)フォルダ、XMLフォルダ、その後にFormatting(書式)任意選択を選ぶ時に利用可能です。

Auto Insert (自動挿入)

Close tags (タグを閉じる)

自動完成設定がチェックされた場合、そのタグが既に閉じられていなければ、開始タグを閉じるために大なり(>)記号を入力した時にエディタは自動的に終了タグを追加します。

空の要素の完成は自動完成設定に依存しません。¥を入力することによって常に空の要素を自動完成することができます。

Attribute quotes (属性引用符)

XML属性を書く時に、エディタは=" "文字を挿入し二重引用符の内側にカレット(^)を配置します。

既定によって選択されます。

Namespace declarations (名前空間宣言)

エディタはそれらが必要とされる時に必ず名前空間宣言を自動的に挿入します。

既定によって選択されます。

Other markup (Comments, CDATA) (他のタグ付け (注釈、CDATA))

注釈、CDATA、DOCTYPE、処理命令、他のタグ付けが自動完成されます。

既定によって選択されます。

Network (ネットワーク)

Automatically download DTDs and schemas (DTDと枠組みを自動ダウンロード)

枠組みと文書形式定義(DTD:Document Type Definition)はHTTP位置から自動的にダウンロードされます。この機能は許可された自動プロキシサーバー検出を持つSystem.Net(名前空間)を使います。

既定によって選択されます。

Outlining (作成)

Enter outlining mode when files open (ファイルを開く時に作成動作形態を入力)

ファイルが開かれる時に作成機能をONにします。

既定によって選択されます。

Caching (キャッシュ)

Schemas (枠組み)

枠組みキャッシュの場所を指定します。検索(...)鈕は現在の枠組みキャッシュ位置でDirectry Browse(ディレクトリ検索)ダイアログ枠を開きます。違うディレクトリを選ぶことができ、またディレクトリ内で何かを見るためにダイアログ内のフォルダを選び、右クリックし、Open(開く)を選ぶことができます。

10.3.5. デバッグ

10.3.5.1. 使い方

Microchip Studioで、変数がどう表示されるか、特定の警告が提示されるかどうか、中断点がどう設定されるか、中断が走行しているプログラムにどう影響を及ぼすかを含むデバッグの動きに対して様々な設定を指定することができます。Options(任意選択)ダイアログ枠でデバッグ設定を指定します。

デバッグ任意設定を設定

Tools(ツール)メニューでOptions(任意選択)をクリックしてください。

Options(任意選択)ダイアログ枠でDebugging(デバッグ)フォルダを開いてください。

Debugging(デバッグ)フォルダで望む任意選択の区分を選んでください。

10.3.5.2. AVR®デバッグ設定

AVR® Communication Timeout (AVR通信制限時間)

後処理部との通信のために使われる制限時間を表示します。ウォッチドッグが制限時間超過を検出した場合に後処理部が再始動されます。既定によって20000msです。

AVR® Debugger Path (AVRデバッグパス)

AVRデバッグに対するパスを表示します。

AVR® Debugger Port (AVRデバッグポート)

AVRデバッグによって使われるWindowsのCOM APIポート番号を示します。既定によって0です。

RPC transaction times (RPC処理時間)

統計記録を置くファイル名。これは後処理部との通信からの記録データです。空は記録が全くないことを意味します。このファイルは使用者が書き込み許可を持つディレクトリ、例えば、**C:\Tmp¥transactionlog.csv**に書かれなければなりません。

User Tool polling (使用者ツールポーリング)

Windows Comm枠組みを信頼する代わりに、ハードウェア ツール発見のために内部ポートポーリング法を使います。活性化された場合、Microchip Studioを再始動しなければならず、それはPCをかなり遅くするかもしれず、故にWindows Comm枠組みに関連する異常がある場合にだけこれを使ってください。既定によって禁止されます。

10.3.6. 高度なソフトウェア枠組み設定

Path of the application used to compare files (ファイルを比較するのに使われる応用のパス)

応用は通常高度なソフトウェア枠組み(ASF)内のファイルを比較するのに使われます。このように、ここでパスを指定しなければなりません。

Command line arguments used for file comparison (ファイル比較に使われるコマンド行引数)

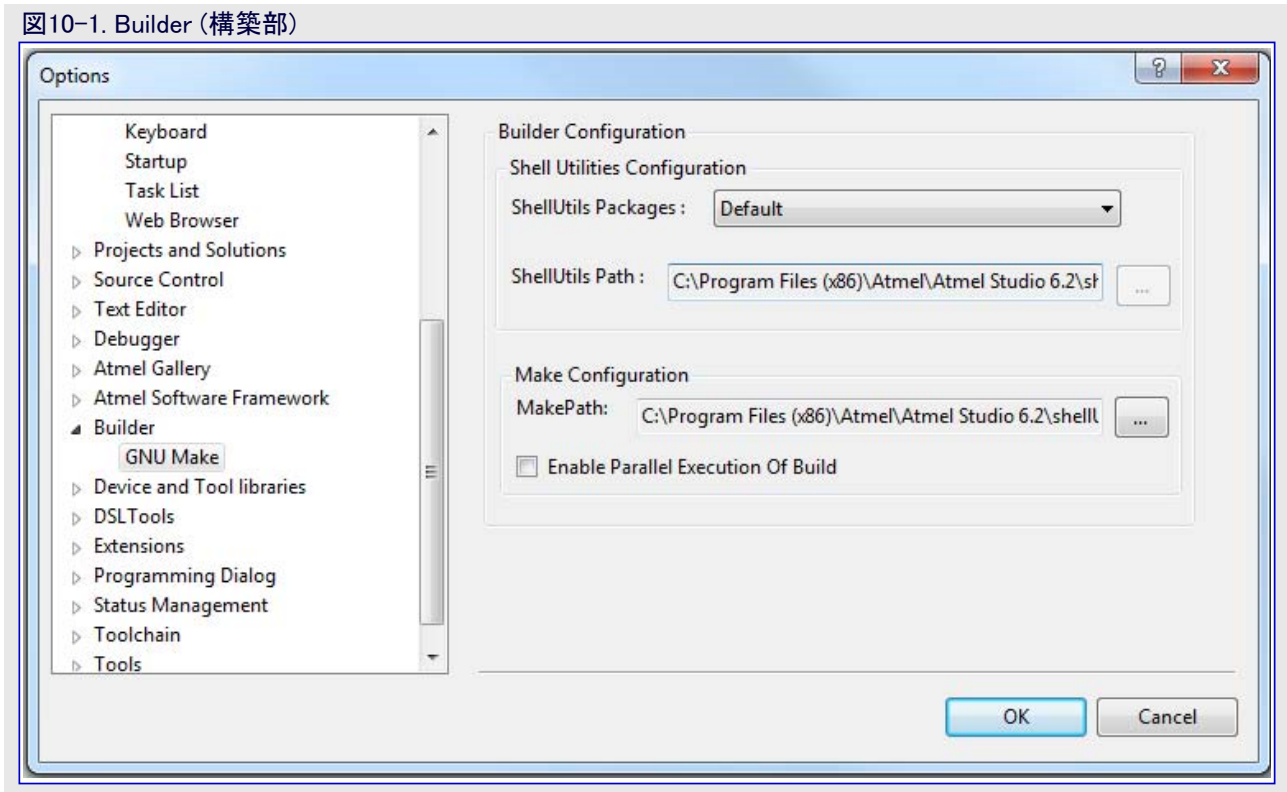
コマンド行引数マクロは次のとおりです。

- **%original** – 元のソフトウェア枠組みファイルのパス
- **%mine** – ローカル プロジェクトで変更されたファイルのパス

構成設定されたファイル比較応用に対するコマンド行が**FileCompare.exe filepath1 filepath2**の場合、**filepath1**に対して**%original**、**filepath2**に対して**%mine**を指定してください。例えば、**WinMerge**を比較応用として構成設定する場合、**%original %mine /s /u**のコマンド行引数を指定してください。

10.3.7. 構築部 (Builder)

図10-1. Builder (構築部)



ShellUtils Packages (シェルユーティリティ一括)

これはDefault(既定)、Custom(独自)、インストールされたシェルユーティリティ拡張の一覧です。

ShellUtils Path (シェルユーティリティのパス)

選ばれた一括に基づいてShellUtils Pathは対応するユーティリティフォルダを指示します。独自シェルユーティリティを選ぶ場合、ファイル選択(...) 鈕をクリックすることによって独自シェルユーティリティフォルダを構成設定することができます。既定またはインストールされたシェル拡張一括を選ぶ場合、このパスは読み出し専用で一括のパスを指示します。

Make Configuration (Make構成設定)

ファイル選択(...)鈕をクリックすることによって実行可能なMakeへのパスを構成設定してください。既定で、これは[インストールディレクトリ]¥shell Utils¥make.exeを指示し、チェック枠をチェックすることによって並行構築を許可することができます。

10.3.8. デバイスとツール ライブラリ

Devices(デバイス)補助メニューでデバイスに対する独自ライブラリへのパスを指定することができます。Tools(ツール)補助メニューでデバイスに対する独自ツールへのパスを指定することができます。

10.3.9. 状態管理

記録ファイルへのパスと記録設定を含みます。

Location (場所)

記録ファイルへのパス。クリックして望む場所を検索することによってこれを変更することができます。

Severity threshold (厳格度閾値)

記録項を生成するために出来事がどれ程厳格でなければならないか。全ての操作が成功の時(OKレベル)、いくつかの変則的なコードが存在する時(Infoレベル)、いくつかの操作が取り消された時(Cancel設定)に出力が欲しいかどうかを選ぶことができます。コードが潜在的に不安定または誤りの時にだけ出力を望む場合、Warning(警告)またはError(異常)のどちらかの設定を選んでください。

Component filter (構成部品選別)

設計内の標準または独自の構成部品に対してソースコードから来る選別メッセージ。

Severity threshold (厳格度閾値)

ソースコード記録生成用厳格度閾値と同じ意味。

Use filter (使用者閾値)

記録処理がコード出力からの構成部品出力を分離するための選別部を使うべきかどうか。

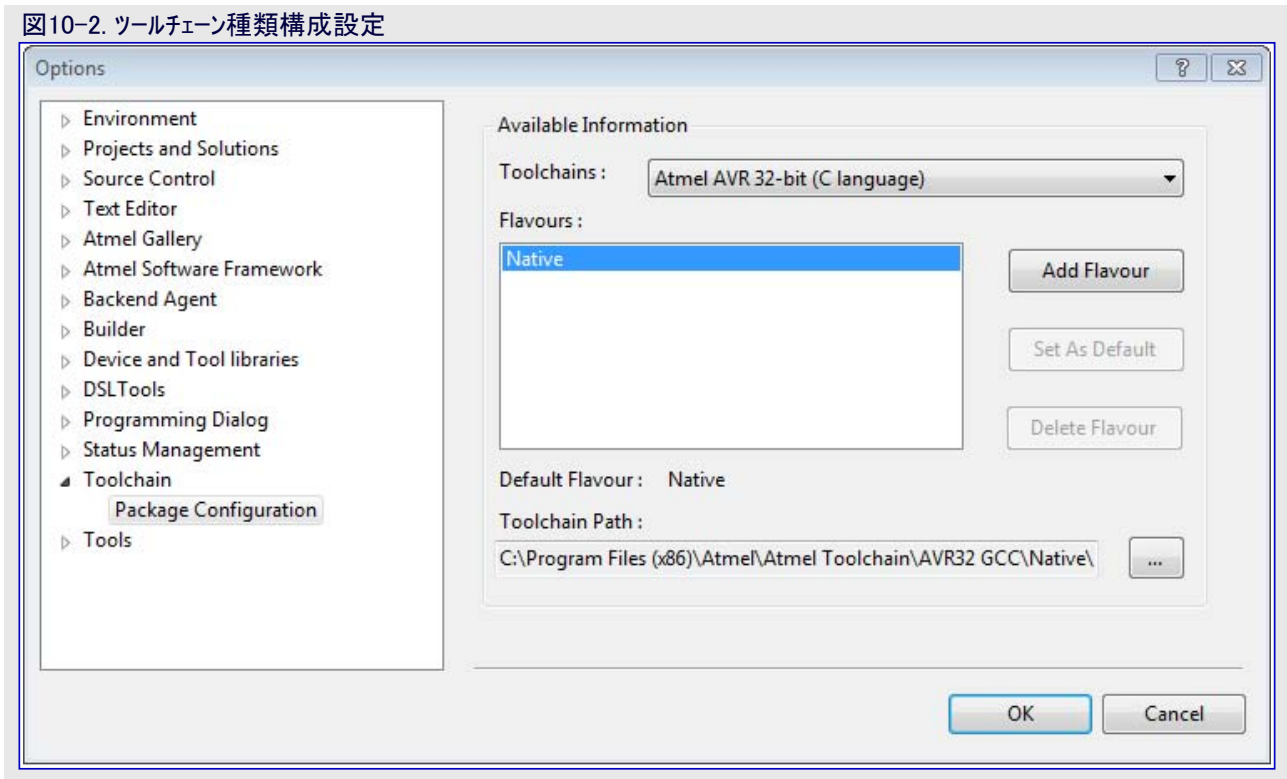
10.3.10. テキスト雛形

Show security message (安全性メッセージ表示)

テキスト変換操作が始められる時に信頼できるソースからのテキスト雛形であることを確実にするために使用者に指示を求めるダイアログを表示します。

10.3.11. ツールチェーン

図10-2. ツールチェーン種類構成設定



Toolchain (ツールチェーン)

ツールチェーンはコンパイラ、リンク、そしてソースコードをAVRデバイスを目標とする実行可能な形式に変換するのに使われます。既定により、Microchip Studioは右のツールチェーン型拡張を持ちます。

表10-1. ツールチェーン任意選択

ツールチェーン型	言語	説明
AVRアセンブラ	アセンブリ	8ビットアセンブラプロジェクト構築に使用
AVR 8ビット	C/C++	8ビットC/C++プロジェクト構築に使用
AVR 32ビット	C/C++	32ビットC/C++プロジェクト構築に使用
Arm 32ビット	C/C++	Arm C/C++プロジェクト構築に使用

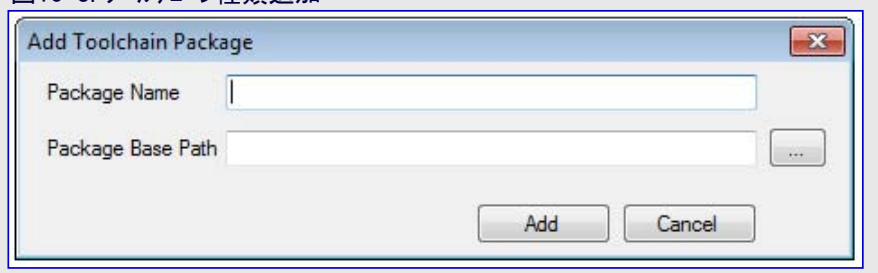
10.3.11.1. Flavor (種類)

Flavor(種類)は望むツールチェーン型のツールチェーン拡張の特定版を識別します。Microchip Studioにインストールされた同じツールチェーン型拡張の違う種類を持つことができます。

10.3.11.1.1. Add Flavor (種類追加)

1. 新しい種類が追加されるツールチェーン型を選んでください。

図10-3. ツールチェーン種類追加



2. 新しいFlavor Name(種類名)を入力してください。
3. 種類(Flavor)に対するツールチェーンパスを構成設定してください。このパスは望むツールチェーン実行可能物、例えば、AVR 8ビットについてはavr-gcc.exeを含むべきです。
4. Add(追加)鈕をクリックしてください。

10.3.11.1.2. Set Default Flavor (既定種類設定)

1. 既定として設定する種類(Flavor)を選んでください。その種類(Flavor)は選んだツールチェーン型に対する既定になります。故に、ツールチェーン型を使う新しいプロジェクトは構成設定された種類(Flavor)設定を使います。
2. 「3.2.7.6. 高度な任意選択」で示されるプロジェクト プロパティ ページを通してプロジェクトを作成した後、様々な種類(Flavor)を見て切り替えることができます。

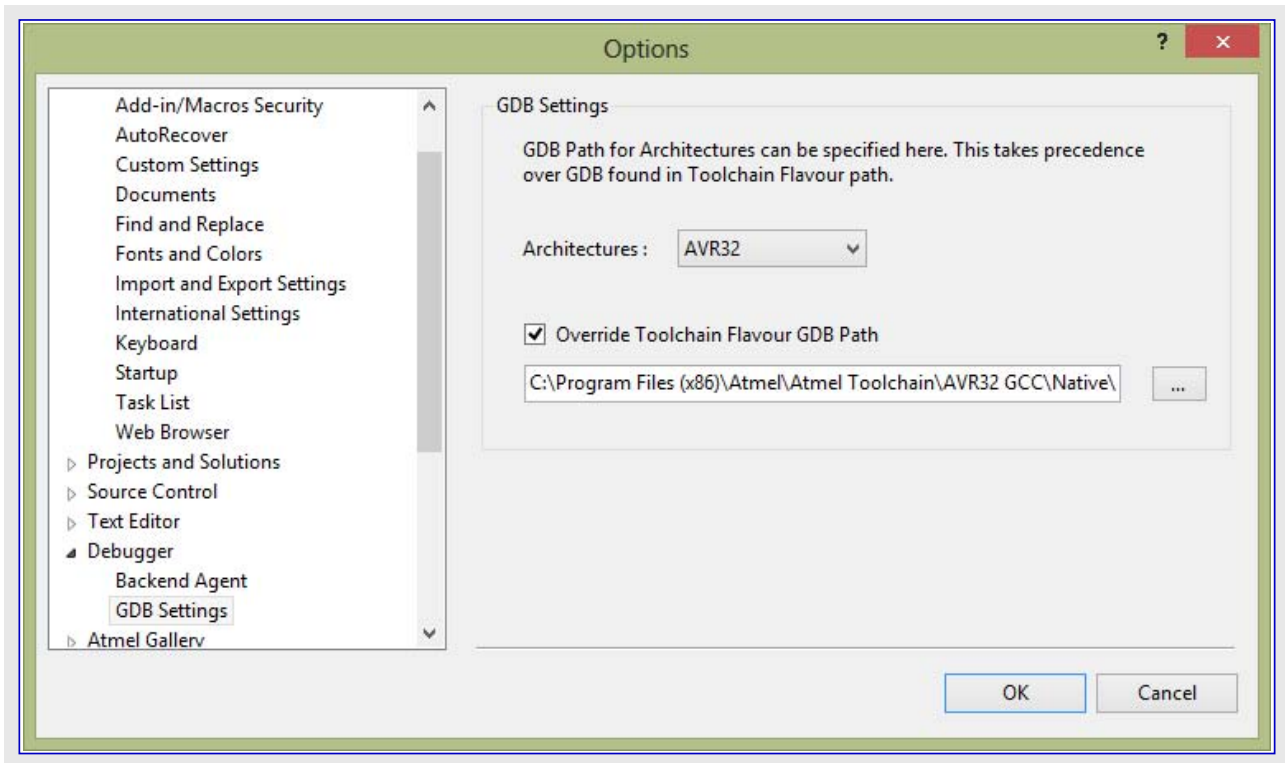
10.3.11.1.3. Delete Flavor (種類削除)

Delete Flavor(種類削除) 釦押下は種類(Flavor)構成設定を削除します。

注: 独自化した既定種類(Flavor)が削除された場合、既定としてNative(本来の)種類(Flavor)が設定されます。また、削除された種類(Flavor)で構成設定されたプロジェクトはプロジェクトが次回開かれた時に各々のツールチェーン型の既定種類(Flavor)に変更されます。

10.3.12. GDB Settings (GDB設定)

このページで基本構造特有GDBパスを構成設定することができます。これは既定ツールチェーン種類(Flavor)GDBパスを上書きします。



10.4. コード断片管理部

コードの断片は特にAVR GCC応用を書く時に有用です。コード断片選択取得部がXML.snippetファイルに対して走査するフォルダ一覧にフォルダを追加するのにコード断片管理部(Code Snippets Manager)を使うことができます。自由に使えるコードのこれらの構築部分を持つことはオプロジェクト開発を容易にすることができます。

コード断片管理部はTools(ツール)メニューからアクセスすることができます。

10.4.1. コード断片管理

コード断片管理部にアクセス

Tools(ツール)メニューでCode Snippets Manager(コード断片管理部)をクリックしてください。

コード断片管理部にディレクトリを追加

1. Language(言語)で: 引き落とし一覧でディレクトリを追加したい言語を選んでください。
2. Add(追加)をクリックしてください。これはCode Snippets Directry(コード断片ディレクトリ)ウィンドウを開きます。
3. コード断片管理部に追加したいディレクトリを選んでOKをクリックしてください。

コード断片管理部からディレクトリを削除

1. 削除したいディレクトリを選んでください。
2. Remove(削除)をクリックしてください。

コード断片管理部にコード断片をインポート

1. **Language**(言語)引き落とし一覧でコード断片を追加したい言語を選んでください。
2. インポートされたコード断片を配置したい既存フォルダを選んでください。
3. **Import**(インポート)をクリックしてください。これは**Code Snippets Directry**(コード断片ディレクトリ)ウィンドウを開きます。
4. コード断片管理部に追加したいコード断片ファイルを選んで**OK**をクリックしてください。コード断片はコードエディタでの挿入用に直ぐに利用可能です。

10.4.2. コード断片管理部配置

Language (言語)

フォルダ一覧で表示されるコード断片フォルダの開発言語を選びます。

Location (場所)

フォルダ一覧内のフォルダ、またはそれらで選択されたコード断片ファイルに対するパスを表示します。

Folder list (フォルダ一覧)

選んだ言語に対して利用可能なコード断片ファイルと、もしあれば副フォルダの組を示します。それを展開してファイルを一覧するには何れかのフォルダをクリックしてください。

Description (説明)

フォルダ一覧で選んだフォルダまたはコード断片ファイルの情報を表示します。コード断片ファイルが選ばれると、その著者、説明、ショートカット、型の領域からテキストを表示します。

Add (場所)

Code Snippets Directry(コード断片ディレクトリ)ダイアログ枠を開きます。ローカルドライブまたはサーバー上の望む断片フォルダへ誘導してフォルダ一覧でそれをインクルードすることを許します。

Remove (削除)

フォルダ一覧から選んだ最上位フォルダとその内容を削除します。フォルダを物理的に削除しません。

Import (インポート)

Code Snippets Directry(コード断片ディレクトリ)ダイアログ枠を開きます。ローカルドライブまたはサーバー上の望む断片へ誘導して既存コード断片フォルダにそれを追加することを許します。

Security (安全性)

コード断片管理部によってアクセスされるフォルダに新しい断片を格納する時は常にそのコードが応用の残りと同じ位安全に構成されることを確実にする責任があります。コード断片を使うことが開発時間を省くため、断片は応用を構成する時に繰り返して再利用することができます。従って、断片で保存される典型的なコードがアドレス安全性の問題に対して設計されることを保証すべきです。開発チームは一般的な安全規格に応じるためにコード断片を再調査する手順を確立すべきです。

10.4.3. 既存コード断片の変更

インテリセンス(IntelliSense)コード断片はMicrochip Studioを含む何れかのXMLエディタを用いて容易に変更することができる**.snippet**ファイル名拡張子を持つXMLファイルです。

既存インテリセンスコード断片を変更

1. 変更したい断片の位置を突き止めるためにコード断片管理部を使ってください。
2. コード断片のパスをクリップボードに複製して**OK**をクリックしてください。
3. **File**(ファイル)メニューで**Open**(開く)をクリックして**File**(ファイル)をクリックしてください。
4. **File location**(ファイル位置)枠に断片のパスを貼り付けて**OK**をクリックしてください。
5. 断片を変更してください。
6. **File**(ファイル)メニューで**Save**(保存)をクリックしてください。保存するにはファイルに対する書き込みアクセス(権)を持たなければなりません。

10.5. 外部ツール

Microchip Studio内から外部ツールを開始することを許すTools(ツール)メニューに項目を追加することができます。例えば、**avrdue**や違うツールのようなユーティリティを開始するためにTools(ツール)メニューに項目を追加することができます。

10.5.1. Tools(ツール)メニューへの外部ツール追加

統合開発環境(IDE)内からNotepadのような別の応用を開始するためにTools(ツール)メニューに命令を追加することができます。

このダイアログは以前に定義された全ての外部ツールが一覧にされる一覧枠を含みます。定義されたとのツールも持たない場合、この一覧枠は空です。

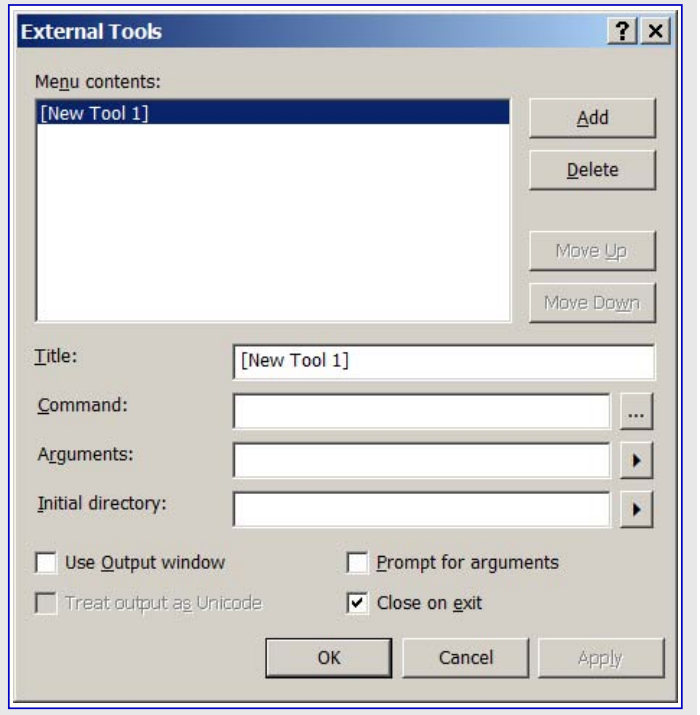
- Tools(ツール)メニューでExternal Tools(外部ツール)を選んでください。
- External Tools(外部ツール)ダイアログ枠でAdd(追加)を選び、Title(表題)枠にメニュー任意選択用の名前を入力してください。



助言: それがTools(ツール)メニューで現れる時に命令用の加速キーを作成するにはツール名で1つの文字の前に'&'を入力してください。例えば、M&y External Toolを使う場合、文字'y'が加速キーになります。より多くの情報については「10.5.5. キーボードショートカット割り当て」をご覧ください。

- Command(命令)枠で開始を意図するファイルへのパスを入力するか、またはファイルへ誘導するために検索(...)を選んでください。
注: ファイルがシステムパスに属す場合、単にファイル名を入力してください。そうでなければ、ファイルに対する完全なパスを入力してください。
- 必要に応じてUse Output window(出力ウィンドウを使用)とClose on exit(終了で閉じる)を選び、その後にOKを選んでください。

図10-4. External Tool(外部ツール)ダイアログ



10.5.2. 外部ツールへの変数渡し

コンソール応用に対するコマンド行スイッチのように、その開始時に或る情報が渡されること指定することができます。

手動または自動充填鈕を使って、必要な開始引数でArguments(引数)枠を満たしてください。

この自動充填引数鈕は次表で記述されるマクロを提供することができます。

表10-2. 外部ツール マクロ

名前	引数	説明
Item Path	\$(ItemPath)	(ドライブ+パス+ファイル名として定義された)現在のソースの完全なファイル名、非ソース ウィンドウが活性の場合は空白
Item Directory	\$(ItemDir)	(ドライブ+パスとして定義された)現在のソースのディレクトリ、非ソース ウィンドウが活性の場合は空白
Item File Name	\$(ItemFilename)	(ファイル名として定義された)現在のソースのファイル名、非ソース ウィンドウが活性の場合は空白
Item Extension	\$(ItemExt)	現在のソースのファイル名拡張子
Current Line	\$(CurLine)	エディタ内のカーソルの現在の行位置
Current Column	\$(CurCol)	エディタ内のカーソルの現在の列位置
Current Text	\$(CurText)	選択されたテキスト
Target Path	\$(TargetPath)	(ドライブ+パス+ファイル名として定義された)構築されるべき項目の完全なファイル名
Target Directory	\$(TargetDir)	構築されるべき項目のディレクトリ
Target Name	\$(TargetName)	構築されるべき項目のファイル名
Target Extension	\$(TargetExt)	構築されるべき項目のファイル名拡張子
Binary Directory	\$(BinDir)	(ドライブ+パスとして定義された)構築されつつあるバイナリの最終位置
Project Directory	\$(ProjectDir)	(ドライブ+パスとして定義された)現在のプロジェクトのディレクトリ
Project filename	\$(ProjectFileName)	(ドライブ+パス+ファイル名として定義された)現在のプロジェクトのファイル名
Solution Directory	\$(SolutionDir)	(ドライブ+パスとして定義された)現在の解決策のディレクトリ
Solution filename	\$(SolutionFileName)	(ドライブ+パス+ファイル名として定義された)現在の解決策のファイル名

10.5.3. 初期デレトリ

ツールまたは命令用の作業デレトリを指定することもできます。例えば、ツールが現在のデレトリからファイル システムのデータを読む場合、ツールは始動で或るプログラム構成部品が現在のデレトリに存在することを要求します。

10.5.4. 走行動作

Arguments(引数)枠の下でツールの動きを変更することができます。

Use Output window (出力ウィンドウを使用) - この枠がチェックされた場合、ツールはMicrochip Studioの出力ウィンドウに処理情報を出力し、さもなければ出力は隠されます。

Close on exit (終了で閉じる) - この枠がチェックされた場合、もしあればツール ウィンドウは全ての操作完了後自動的に閉じられます。

Prompt for arguments (引数に対して指示要求) - ツールチェーン自動化に使われます。この枠がチェックされた場合、外部ツールは追加の処理パラメータを入力するために使用者の介入が必要です。さもなければツールは沈黙を守ります。

Treat output as Unicode (ユニコードとして出力を処理) - 国際化任意選択。いくつかのツールはより良い解釈に帰着するユニコードを出力することができます。この任意選択はこのようなツールを使う場合に正しい出力表現を許します。

10.5.5. キーボード ショートカット割り当て

命令にショートカット(加速器)を割り当てるにはツールの表題に於いてアクセス キーとして使いたい文字の直前にアンド記号(&)を追加してください。

アンド記号が追加された後、加速器はキーボード ショートカットとして含められることが必要です。

- **Tools**(ツール)メニューで**Options**(任意選択)をクリックしてください。
- **Environment**(環境)ページで**Keyboard**(キーボード)を選んでください。
- **Show commands containing**(含む命令を表示)一覧で**Tools**(ツール)を入力してください。
- **Command names**(命令名)一覧で適切な**External Command n**(外部命令n)登録を配置してください。

注: 最大12の外部ツールに対してキーボード ショートカットを定義することができます。外部ツールは**Command names**(命令名)一覧で**External Command 1**~**20**として一覧にされます。この番号は**Tools**(ツール)メニューで独自外部命令名の左の番号に対応します。メニュー命令がそれに割り当てられたショートカットを既に持つ場合、**Shortcuts for selected command**(選んだ命令に対するショートカット)一覧にその情報が現れます。

- **Press shortcut keys**(ショートカット キー押下)枠にカーソルを置き、その後に外部ツールに割り当てたいキーを押してください。
注: キーボード ショートカットが既に別の命令に割り当てられていた場合、**Shortcut currently assigned**(現在割り当てショートカット)一覧がその情報を表示します。
- **Assign**(割り当て)をクリックしてください。

10.6. 予め定義されたキーボード ショートカット

Microchip StudioはMicrosoft Visual Studio 2010からのVisual Studioシェル枠組みを使い、従って、統合開発環境(IDE)はVisual Studioでのそれらと同様な少数の予め定義されたキーボード ショートカット体系を含みます。初めてMicrochip Studioを開始してあなたの設定を選ぶと、関連する体系が自動的に設定されます。その後Options(任意選択)ダイアログ枠でKeyboard options(キーボード任意選択)頁を使うことによって、追加の体系から選ぶことができ、あなた自身のキーボード ショートカットを作成することもできます。

設計者と編集者、共用ショートカット

これらのショートカットは設計者と編集者の両方で動きます。

命令	ショートカット	説明
Edit.Copy	Ctrl+CまたはCtrl+Insert	選択した項目をクリップボードに複写
Edit.Cut	Ctrl+XまたはShift+Delete	ファイルから選択した項目を削除してそれをクリップボードに複写
Edit.CycleClipboardRing	Ctrl+Shift+V	環状クリップボードから項目をファイル内のカーソル位置に貼り付け。代わりに環状クリップボード内の次の項目を貼り付けるには再びショートカットを押してください。
Edit.Delete	Delete	カーソルの右側の1文字を削除
Edit.Find	Ctrl+F	検索と置換ダイアログ枠のQuick(迅速)タブを表示
Edit.FindAllReferences	Shift+Alt+F	選択したシンボルに対する参照の一覧を表示
Edit.FindinFiles	Ctrl+Shift+F	検索と置換ダイアログ枠のFiles(ファイル)タブを表示
Edit.FindNext	F3	検索テキストの次の発生(一致)を検索
Edit.FindNextSelected	Ctrl+F3	現在の選択テキストまたはカーソルの語の次の発生(一致)を検索
Edit.FindPrevious	Shift+F3	検索テキストの直前の発生(一致)を検索
Edit.FindPreviousSelected	Ctrl+Shift+F3	現在の選択テキストまたはカーソルの語の直前の発生(一致)を検索
Edit.FindSymbol	Alt+F12	検索と置換ダイアログ枠のFind Symbol(検索シンボル)枠を表示
Edit.GoToFindCombo	Ctrl+D	標準ツールバーのFind/Command(検索/命令)枠にカーソルを配置
Edit.IncrementalSearch	Ctrl+I	逐次検索を活性。逐次検索がONだけでも入力が全く渡されない場合は直前の検索問い合わせが使われます。検索入力が見つかった場合、次の呼び出しは入力テキストの次の発生に対して検索します。
Edit.Paste	Ctrl+VまたはShift+Insert	カーソル位置にクリップボードの内容を挿入
Edit.QuickFindSymbol	Shift+Alt+F12	選択したオブジェクトまたはメンバに対して検索し、Find Symbol Results(シンボル検索結果)ウィンドウに一致を表示
Edit.NavigateTo	Ctrl+,	Navigate To(~に誘導)ダイアログ枠を表示
Edit.Redo	Ctrl+Yまたは Shift+Alt+BackSpace またはCtrl+Shift+Z	最も最新の活動を繰り返し
Edit.Replace	Ctrl+H	Find and Replace(検索と置換)ダイアログ枠のQuick(迅速)タブにReplace(置換)任意選択を表示
Edit.ReplaceinFiles	Ctrl+Shift+H	Find and Replace(検索と置換)ダイアログ枠のFiles(ファイル)タブにReplace(置換)任意選択を表示
Edit.SelectAll	Ctrl+A	現在の文書で全てを選択
Edit.StopSearch	Alt+F3, S	ファイル操作で現在の検索を停止
Edit.Undo	Ctrl+Zまたは Alt+BackSpace	最後の編集活動を逆戻り
View.ViewCode	Ctrl+Alt+0	選択した項目に対して、対応するファイルを開き、現在位置にカーソルを配置

テキスト誘導

これらのショートカットは開いた文書での周辺移動用です。

命令	ショートカット	説明
Edit.CharLeft	←	カーソルを左に1文字移動
Edit.CharRight	→	カーソルを右に1文字移動
Edit.DocumentEnd	Ctrl+End	カーソルを文書の最終行に移動
Edit.DocumentStart	Ctrl+Home	カーソルを文書の先頭行に移動
Edit.GoTo	Ctrl+G	Go To Line(行へ行く)ダイアログ枠を表示
Edit.GoToDefinition	Alt+G	コード内の選択したシンボルに対する宣言へ誘導
Edit.GoToNextLocation	F8	カーソルをTask List(タスク一覧)ウィンドウやFind Results(検索結果)ウィンドウでの検索一致のような次の項目へ移動。後続する呼び出しは一覧内の次の項目へ移動します。
Edit.GoToPrevLocation	Shift+F8	カーソルを直前の項目へ戻すように移動
Edit.IncrementalSearch	Ctrl+I	逐次検索開始。逐次検索が開始されたが、どの文字も入力されない場合は直前の様式が再び呼び出されます。テキストが見つかった場合、次の発生に対して検索します。
Edit.LineDown	↓	カーソルを1行下へ移動
Edit.LineEnd	End	カーソルを現在の行末へ移動
Edit.LineStart	Home	カーソルを現在の行先頭へ移動
Edit.LineUp	↑	カーソルを1行上へ移動
Edit.NextBookmark	Ctrl+K, Ctrl+N	文書内の次の葉へ移動
Edit.NextBookmarkInFolder	Ctrl+Shift+K, Ctrl+Shift+N	現在の葉がフォルダ内なら、そのフォルダ内の次の葉へ移動。フォルダ外の葉は飛ばします。 現在の葉がフォルダ内でなければ、同じレベルで次の葉へ移動。 Bookmark(葉)ウィンドウがフォルダを含むなら、フォルダ内の葉を飛ばします。
Edit.PageDown	PageDown	エディタ ウィンドウ内で1画面下スクロール
Edit.PageUp	PageUp	エディタ ウィンドウ内で1画面上スクロール
Edit.PreviousBookmark	Ctrl+K, Ctrl+P	カーソルを直前の葉の位置へ移動
Edit.PreviousBookmarkInFolder	Ctrl+Shift+K, Ctrl+Shift+P	現在の葉がフォルダ内なら、そのフォルダ内の直前の葉へ移動。フォルダ外の葉は飛ばします。 現在の葉がフォルダ内でなければ、同じレベルで直前の葉へ移動。 Bookmark(葉)ウィンドウがフォルダを含むなら、フォルダ内の葉を飛ばします。
Edit.ReverseIncrementalSearch	Ctrl+Shift+I	ファイルの最後で開始して先頭に向かって進むように逐次検索の方向を変更
Edit.ScrollLineDown	Ctrl+ ↓	テキストを1行下スクロール。テキスト エディタでだけ利用可能
Edit.ScrollLineUp	Ctrl+ ↑	テキストを1行上スクロール。テキスト エディタでだけ利用可能
Edit.ViewBottom	Ctrl+PageDown	活性なウィンドウの見える最後の行へ移動
Edit.ViewTop	Ctrl+PageUp	活性なウィンドウの見える最初の行へ移動
Edit.WordNext	Ctrl+→	カーソルを1語右へ移動
Edit.WordPrevious	Ctrl+←	カーソルを1語左へ移動
View.NavigateBackward	Ctrl+-	コードの直前の閲覧した行へ移動
View.NavigateForward	Ctrl+Shift+-	コードの次の閲覧した行へ移動
View.NextError	Ctrl+Shift+F12	エディタ内のテキストの影響を及ぼされた項へ自動的にスクロールする、Error List(異常一覧)ウィンドウで次の異常登録へ移動
View.NextTask		Task List(タスク一覧)ウィンドウで次のタスクまたは注釈へ移動

ビジュアル アシスト (Visual Assist) ショートカット

これらのショートカットはビジュアル アシスト(Visual Assist)用です。

命令	ショートカット	説明
VAssistX.FindReference	Shift+Alt+F	記されたテキストへの全ての参照を検索
VAssistX.FindSymbolDialog	Shift+Alt+S	プロジェクト内の全シンボルを一覧にするシンボルダイアログを開く
VAssistX.GotoImplementation	Alt+G	実装へ行く
VAssistX.ListMethodsInCurrentFile	Alt+M	現在のファイル内の全メソッドの一覧を開く
VAssistX.OpenCorrespondingFile	Alt+O	対応するファイル(例えば.h/.c)を開く
VAssistX.OpenFileInSolutionDialog	Shift+Alt+O	解決策内の全ファイルの一覧を表示
VAssistX.Paste	Ctrl+Shift+V	貼り付け履歴メニューを見せる
VAssistX.RefactorContextMenu	Shift+Alt+Q	リファクタリング状況メニューを見せる
VAssistX.RefactorRename	Shift+Alt+R	改名ダイアログを見せる
VAssistX.ScopeNext	Alt+ ↓	次の範囲へ飛ぶ
VAssistX.ScopePrevious	Alt+ ↑	直前の範囲へ飛ぶ

テキスト選択

これらのショートカットは開いた文書でのテキスト選択用です。

命令	ショートカット	説明
Edit.CharLeftExtend	Shift+←	カーソルを1文字左に移動して現在の選択を延長
Edit.CharLeftExtendColumn	Shift+Alt+←	カーソルを1文字左に移動して列選択を拡張
Edit.CharRightExtend	Shift+→	カーソルを1文字右に移動して現在の選択を延長
Edit.CharRightExtendColumn	Shift+Alt+→	カーソルを1文字右に移動して列選択を拡張
Edit.DocumentEndExtend	Ctrl+Shift+End	カーソルから文書の最終行までのテキストを選択
Edit.DocumentStartExtend	Ctrl+Shift+Home	カーソルから文書の先頭行までのテキストを選択
Edit.LineDownExtend	Shift+ ↓	カーソルの位置で始めて1行下へテキスト選択拡張
Edit.LineDownExtendColumn	Shift+Alt+ ↓	列選択を拡張して1行下へポインタを移動
Edit.LineEndExtend	Shift+End	カーソルから現在行末までのテキストを選択
Edit.LineEndExtendColumn	Shift+Alt+End	列選択を拡張してカーソルを行末まで移動
Edit.LineStartExtend	Shift+Home	カーソルから行先頭までテキストを選択
Edit.LineStartExtendColumn	Shift+Alt+Home	列選択を拡張してカーソルを行先頭まで移動
Edit.LineUpExtend	Shift+ ↑	カーソルの位置から始めて行単位で上へテキストを選択
Edit.LineUpExtendColumn	Shift+Alt+ ↑	列選択を拡張してカーソルを1行上へ移動
Edit.PageDownExtend	Shift+PageDown	1頁下へ選択拡張
Edit.PageUpExtend	Shift+PageUp	1頁上へ選択拡張
Edit.SelectCurrentWord	Ctrl+W	カーソルを含む語またはカーソルの右側の語を選択
Edit.SelectionCancel	Esc	現在の選択を取り消し
Edit.ViewBottomExtend	Ctrl+Shift+PageDown	表示部の最終行へカーソルを移動して選択を拡張
Edit.ViewTopExtend	Ctrl+Shift+PageUp	活性化ウィンドウの先頭へ選択を拡張
Edit.WordNextExtend	Ctrl+Shift+→	右へ1語選択を延長
Edit.WordNextExtendColumn	Ctrl+Shift+Alt+→	列選択を拡張して右へ1語カーソル移動
Edit.WordPreviousExtend	Ctrl+Shift+←	左へ1語選択を延長
Edit.WordPreviousExtendColumn	Ctrl+Shift+Alt+←	列選択を拡張して左へ1語カーソル移動

テキスト表示

これらのショートカットは、例えば、選択領域を隠す、または輪郭法によって、テキスト自体の変更なしにテキストが表示される方法の変更用です。

命令	ショートカット	説明
Edit.ClearBookmarks	Ctrl+K, Ctrl+L	開いた全ての文書で全葉を削除
Edit.CollapseAllOutlining	Ctrl+M, Ctrl+A	階層でまさに最も外側を見せるように全ての領域を収納、代表的に領域と名前空間の定義で使用またはインポート
Edit.CollapseCurrentRegion	Ctrl+M, Ctrl+S	省略記号が後続する領域のまさに先頭線を見せるようにカーソルを含む領域を収納。領域は文書ウィンドウの左端の三角形によって示されます。
Edit.CollapseTag	Ctrl+M, Ctrl+T	選択されたHTMLタグを隠して代わりに省略記号(...)を表示。マウスポインタを省略記号上に置くことによって情報(Tooltip)として完全なタグを見ることができます。
Edit.CollapseToDefinitions	Ctrl+M, Ctrl+O	ソースファイルで型やメンバの高位表示を提供するために既存領域を収納
Edit.EnableBookmark		現在の文書で葉の使用を許可
Edit.ExpandAllOutlining	Ctrl+M, Ctrl+X	ページで収納された全ての領域を展開
Edit.ExpandCurrentRegion	Ctrl+M, Ctrl+E	現在の領域を展開。この命令を使うためにカーソルを収納された領域上に置く
Edit.HideSelection	Ctrl+M, Ctrl+H	選択したテキストを隠す。ファイルの隠したテキストの位置を合図アイコンで記す。
Edit.StopHidingCurrent	Ctrl+M, Ctrl+U	現在選択した領域に対する概略情報を削除
Edit.StopOutlining	Ctrl+M, Ctrl+P	文書全体から全ての概略情報を削除
Edit.ToggleAllOutlining	Ctrl+M, Ctrl+L	以前に収納した全輪郭領域の収納と展開の状態を交互切り替え
Edit.ToggleBookmark	Ctrl+K, Ctrl+K	現在の行で葉を設定または削除
Edit.ToggleOutliningExpansion	Ctrl+M, Ctrl+M	現在選択されて収納した輪郭領域の収納と展開の状態を交互切り替え
Edit.ToggleTaskListShortcut	Ctrl+K, Ctrl+H	現在の行でショートカットを設定または削除
Edit.ToggleWordWrap	Ctrl+E, Ctrl+W	エディタでのワードラップを許可または禁止
Edit.ViewWhiteSpace	Ctrl+R, Ctrl+W	空白とタブの印を見せるまたは隠す

テキスト操作

これらのショートカットは開いた文書でのテキストの削除、移動、書式設定用です。

命令	ショートカット	説明
Edit.BreakLine	Enter	新しい行を挿入
Edit.CharTranspose	Ctrl+T	カーソルの左右で文字を入れ替え。例えば、AC BDはAB CDになります。
Edit.CommentSelection	Ctrl+K, Ctrl+C	現在の選択に現在の言語用の注釈文字を適用
Edit.CompleteWord	Alt+→またはCtrl+SpaceBar	完成一覧で現在の語を完成
Edit.DeleteBackwards	BackSpace	カーソルの左側の1文字を削除
Edit.FormatDocument	Ctrl+K, Ctrl+D	現在の言語に対して、Options(任意選択)ダイアログ枠のFormatting(書式)枠で指定された字下げとコード書式設定に従って現在の文書を書式設定します。
Edit.FormatSelection	Ctrl+K, Ctrl+F	現在の言語に対して、Options(任意選択)ダイアログ枠のFormatting(書式)枠で指定された字下げとコード書式設定に従って現在の選択を書式設定します。
Edit.InsertSnippet	Ctrl+K, Ctrl+X	コード断片選択取得部を表示。選択したコード断片がカーソル位置に挿入されます。
Edit.InsertTab	Tab	テキストの行に指定された数の空白で字下げ
Edit.LineCut	Ctrl+L	選択された全ての行、または何も選択されていない場合は現在の行をクリップボードに切り取り

[次頁へ続く](#)

前頁からの続き

命令	ショートカット	説明
Edit.LineDelete	Ctrl+Shift+L	全選択行または選択されていない場合は現在行を削除
Edit.LineOpenAbove	Ctrl+Shift+Enter	カーソルの上に空白行を挿入
Edit.LineOpenBelow	Ctrl+Enter	カーソルの下に空白行を挿入
Edit.LineTranspose	Shift+Alt+T	カーソルを含む行を次の行の下へ移動
Edit.ListMembers	Ctrl+J	インテリセンス(IntelliSense)完成一覧を呼び出し
Edit.MakeLowercase	Ctrl+U	選択したテキストを小文字に変更
Edit.MakeUppercase	Ctrl+Shift+U	選択したテキストを大文字に変更
Edit.Overtypemode	Insert	挿入動作と上書き動作間を交互切り替え
Edit.ParameterInfo	Ctrl+Shift+SpaceBar	指定したメソッドに対して必要されるパラメータの名前、数、型を表示
Edit.SurroundWith	Ctrl+K, Ctrl+S	コード断片選択取得部を表示。選んだコード断片は選択したテキストに周囲を囲まれます。
Edit.TabifySelectedLines		選択したテキストで空白をタブに置換
Edit.TabLeft	Shift+Tab	選択した行を1つ左のタブ位置へ移動
Edit.UncommentSelection	Ctrl+K, Ctrl+U	コードの現在行から注釈構文を削除
Edit.UntabifySelectedLines		選択したテキストでタブを空白に置換
Edit.WordDeleteToEnd	Ctrl+Delete	カーソルの右側の語を削除
Edit.WordDeleteToStart	Ctrl+BackSpace	カーソルの左側の語を削除
Edit.WordTranspose	Ctrl+Shift+T	カーソルの両側の語を置き換え。例えば、 End SubはSub End を読むように変更されます。

ファイルとプロジェクトの操作

これらのショートカットはファイルとプロジェクトの操作用で、IDEで何処でも使うことができます。

命令	ショートカット	説明
Build.BuildSelection		選択したプロジェクトとその依存物を構築
Build.BuildSolution	F7	解決策内の全プロジェクトを構築
Build.Cancel	Ctrl+Break	現在の構築を停止
Build.Compile	Ctrl+F7	選択したファイル用の機械語、リンク指令、領域、外部参照、関数/データの名前を含むオブジェクトファイルを作成
Build.RebuildSolution	Ctrl+Alt+F7	解決策を再構築
File.NewFile	Ctrl+N	現在のプロジェクトに新しいファイルを追加できるようにNew File(新しいファイル)ダイアログ枠を表示
File.NewProject	Ctrl+Shift+N	New Project(新しいプロジェクト)ダイアログ枠を表示
File.OpenFile	Ctrl+O	Open File(ファイルを開く)ダイアログ枠を表示
File.OpenProject	Ctrl+Shift+O	解決策に既存プロジェクトを追加できるようにOpen Project(プロジェクトを開く)ダイアログ枠を表示
File.Print	Ctrl+P	印刷機設定を選べるようにPrint(印刷)ダイアログ枠を表示
File.Rename	F2	Solution Explorer(解決策エクスプローラ)で選択した項目の名前の変更を許します。
File.SaveAll	Ctrl+Shift+S	現在の解決策内の全文書と外部ファイル、プロジェクト内の全ファイルを保存
File.SaveSelectedItems	Ctrl+S	現在のプロジェクトで選択された項目を保存
File.SaveSelectedItemsAs		エディタで項目が選択された時にSave File As(~としてファイル)を保存)ダイアログ枠を表示
Project.AddExistingItem		現在のプロジェクトに既存ファイルの追加を許す、Add Existing Item(既存項目を追加)ダイアログ枠を表示
Project.AddNewItem		現在のプロジェクトに新しいファイルの追加を許す、Add New Item(新しい項目を追加)ダイアログ枠を表示
Project.Properties		編集枠組みで現在のプロジェクトに対するProject Properties(プロジェクトプロパティ)ダイアログ枠を表示

ウィンドウ管理

これらのショートカットはツール ウィンドウと文書ウィンドウでの移動、閉じる、誘導用です。

命令	ショートカット	説明
View.FullScreen	Shift+Alt+Enter	全画面動作ON/OFF交互切り替え
Window.ActivateDocumentWindow	Esc	メニューまたはダイアログ枠を閉じ、進行中の操作を取り消し、または現在の文書ウィンドウにフォーカスを置く。
Window.CloseDocumentWindow	Ctrl+F4	現在のタブを閉じる
Window.CloseToolWindow	Shift+Esc	現在のツール ウィンドウを閉じる
Window.Dock		IDEでその最も最近の接合位置に浮きツールまたは文書ウィンドウを戻す。
Window.NextDocumentWindow	Ctrl+F6	開いた文書を通して巡回
Window.NextDocumentWindowNav	Ctrl+Tab	選択した最初の文書ウィンドウでIDE誘導部を表示
Window.NextPane	Alt+F6	現在のツールまたは文書ウィンドウの次の枠へ移動
Window.NextToolWindow		次のツール ウィンドウへ移動
Window.NextToolWindowNav	Alt+F7	選択した最初のツール ウィンドウでIDE誘導部を表示
Window.PreviousDocumentWindow	Ctrl+Shift+F6	エディタで直前の文書へ移動
Window.PreviousDocumentWindowNav	Ctrl+Shift+Tab	選択した直前の文書ウィンドウでIDE誘導部を表示
Window.PreviousPane	Shift+Alt+F6	直前に選択したウィンドウへ移動
Window.ShowEzMDIFileList	Ctrl+Alt+ ↓	開いている全ての文書だけを一覧にするポップアップを表示

ツール ウィンドウ

これらのショートカットはIDEで何処でもツール ウィンドウを開くため用です。

命令	ショートカット	説明
Tools.CodeSnippetsManager	Ctrl+K, Ctrl+B	ファイルでコード断片の検索と挿入を許すコード断片管理部を表示
Tools.GoToCommandLine	Ctrl+ /	標準ツールバーのFind/Command(検索/命令)ボックスにポインタを置く
View.BookmarkWindow	Ctrl+K, Ctrl+W	Bookmark(葉)ウィンドウを表示
View.CallHierarchy	Ctrl+Alt+K	Call Hierachy(呼び出し階層)ウィンドウを表示
View.CommandWindow	Ctrl+Alt+A	IDEに変更させるために命令を呼び出すことができるCommand(命令)ウィンドウを表示
View.EditLabel	F2	Solution Explorer(解決策エクスプローラ)で選択した項目の名前の変更を許します。
View.ErrorList	Ctrl+¥, E	Error List(異常一覧)ウィンドウを表示
View.FindSymbolResults	Ctrl+Alt+F12	Find Synbol Results(シンボル検索結果)ウィンドウを表示
View.Output	Ctrl+Alt+O	走行時に状態メッセージを見るためにOutput(出力)ウィンドウを表示
View.SolutionExplorer	Ctrl+Alt+L	現在のプロジェクトとファイルを一覧にするSolution Explorer(解決策エクスプローラ)を表示
View.TaskList	Ctrl+¥, T	独自タスク、注釈、ショートカット、警告、異常のメッセージを表示するTask List(タスク一覧)ウィンドウを表示
View.WebBrowser	Ctrl+Alt+R	インターネットでページを見させるウェブ ブラウザ ウィンドウを表示
Window.PreviousToolWindow		直前のツール ウィンドウにフォーカスを持ってくる。
Window.PreviousToolWindowNav	Shift+Alt+F7	直前に選択したツール ウィンドウでIDE誘導部を表示

葉ウィンドウ

これらのショートカットは葉ウィンドウまたはエディタのどちらかで葉と共に動くため用です。より多くの情報については下表をご覧ください。

命令	ショートカット	説明
Edit.ClearBookmarks	Ctrl+K, Ctrl+L	開いた全ての文書で全ての葉を削除
Edit.EnableBookmark		現在の文書で葉の使用を許可
Edit.NextBookmark	Ctrl+K, Ctrl+N	文書内で次の葉へ移動
Edit.NextBookmarkInFolder	Ctrl+Shift+K, Ctrl+Shift+N	現在の葉がフォルダ内なら、そのフォルダ内の次の葉に移動。フォルダの外側の葉は飛ばします。 現在の葉がフォルダ内でなければ、同じレベルで次の葉へ移動 Bookmark(葉)ウィンドウがフォルダを含む場合、フォルダ内の葉は飛ばされます。
Edit.ToggleBookmark	Ctrl+K, Ctrl+K	文書内の現在行の葉を交互切り替え
View.BookmarkWindow	Ctrl+K, Ctrl+W	Bookmark(葉)ウィンドウを表示
Edit.PreviousBookmark	Ctrl+K, Ctrl+P	カーソルを直前の葉の位置へ移動
Edit.PreviousBookmarkInFolder	Ctrl+Shift+K, Ctrl+Shift+P	現在の葉がフォルダ内なら、そのフォルダ内の直前の葉に移動。フォルダの外側の葉は飛ばします。 現在の葉がフォルダ内でなければ、同じレベルで直前の葉へ移動 Bookmark(葉)ウィンドウがフォルダを含む場合、フォルダ内の葉は飛ばされます。

デバッグ

これらのショートカットはコード デバッグ用です。

命令	ショートカット	説明
Debug.Autos	Ctrl+Alt+V, A	コードの現在と直前の行で使われる変数を表示するAuto(自動)ウィンドウを表示
Debug.BreakAll	Ctrl+F5	デバッグ作業での全ての処理の実行を一時停止。走行動作でだけ利用可能
Debug.BreakatFunction	Ctrl+B	New Breakpoint(新しい中断点)ダイアログ枠を表示
Debug.Breakpoints	Alt+F9またはCtrl+Alt+B	中断点を追加、削除、変更することができるBreakpoints(中断点)ダイアログ枠を表示
Debug.CallStack	Alt+7またはCtrl+Alt+C	活性な全メソッドまたは現在のスレッドの実行用のスタック フレームを表示するCall Stack(呼び出しスタック)ウィンドウを表示
Debug.DeleteAllBreakpoints	Ctrl+Shift+F9	プロジェクト内の全ての中断点を解除
Debug.Disassembly	Alt+8またはCtrl+Alt+D	Disassembly(逆アセンブリ)ウィンドウを表示
Debug.EnableBreakpoint	Ctrl+F9	中断点の禁止と許可を交互切り替え
Debug.Exceptions	Ctrl+Alt+E	Exceptions(例外)ダイアログ枠を表示
Debug.Immediate	Ctrl+Alt+I	式を評価することができるImmediate(即値)ウィンドウを表示
Debug.Locals	Alt+4またはCtrl+Alt+V, L	現在のスタック フレーム内の各メソッドに対する局所変数とそれらの値を表示するLocals(局所)ウィンドウを表示
Debug.Memory1	Ctrl+Alt+M, 1	Watch(監視)やVariables(変数)のウィンドウではっきりと表示されない大きな緩衝部、文字列、他のデータを見るためにMemory 1(メモリ1)ウィンドウを表示
Debug.Memory2	Ctrl+Alt+M, 2	Watch(監視)やVariables(変数)のウィンドウではっきりと表示されない大きな緩衝部、文字列、他のデータを見るためにMemory 2(メモリ2)ウィンドウを表示
Debug.Memory3	Ctrl+Alt+M, 3	Watch(監視)やVariables(変数)のウィンドウではっきりと表示されない大きな緩衝部、文字列、他のデータを見るためにMemory 3(メモリ3)ウィンドウを表示

[次頁](#)に続く

前頁からの続き

命令	ショートカット	説明
Debug.Memory4	Ctrl+Alt+M, 4	Watch(監視)やVariables(変数)のウィンドウではっきりと表示されない大きな緩衝部、文字列、他のデータを見るためにMemory 4(メモリ4)ウィンドウを表示
Debug.Modules	Ctrl+Alt+U	プログラムで使われる.dllまたは.exeのファイルを見ることを許すModules(単位部)ウィンドウを表示。マルチプロセスデバッグでは全てのプログラムに対して右クリックしてその後にShow Modules(単位部を表示)をクリックすることができます。
Debug.ParallelStacks	Ctrl+Shift+D, S	Parallel Stacks(並行スタック)ウィンドウを表示
Debug.ParallelTasks	Ctrl+Shift+D, K	Parallel Tasks(並行タスク)ウィンドウを表示
Debug.Processes	Ctrl+Alt+Z	Processes(処理)ウィンドウを表示。走行動作で利用可能
Debug.QuickWatch	Ctrl+Alt+QまたはShift+F9	選択した式の現在の値を持つQuickWatch(一瞬監視)ウィンドウを表示。中断動作でだけ利用可能。監視式を定義しなかった変数、プロパティ、他の式の現在の値を調べるのにこの命令を使ってください。
Debug.Registers	Alt+5またはCtrl+Alt+G	本来のコード応用のデバッグに対してレジスタ内容を表示するRegisters(レジスタ)ウィンドウを表示
Debug.RunToCursor	Ctrl+F10	中断動作では、現在の文から選択した文までコードの実行を再開。余白指示部バーに現在実行行余白指示部が現れます。設計動作では、デバッグを開始してポインタ位置までコードを実行します。
Debug.Start	F5	始動プロジェクトからの設定に基づくデバッグ下で応用を開始。中断動作時、この命令呼び出しは応用を次の中断点まで走らせます。
Debug.StepInto	F11	メソッド呼び出し内の実行に従って、1度に1つの文でコードを実行。
Debug.StepIntoCurrentProcess	Ctrl+Alt+F11	Processes(処理)ウィンドウから利用可能
Debug.StepOver	F10	選択したコードの行に実行点を設定
Debug.StopDebugging	Ctrl+Shift+F5	デバッグ下の現在の応用の走行を停止
Debug.Threads	Ctrl+Alt+H	走行しているスレッドを見るのにThreads(スレッド)ウィンドウを表示
Debug.ToggleBreakpoint	F9	現在の行で中断点を設定または削除
Debug.Watch1	Ctrl+Alt+W, 1	選んだ変数または監視式の値を表示するWatch(監視)ウィンドウを表示
Debug.Watch2	Ctrl+Alt+W, 2	選んだ変数または監視式の値を表示するWatch2(監視)ウィンドウを表示
Debug.Watch3	Ctrl+Alt+W, 3	選んだ変数または監視式の値を表示するWatch3(監視)ウィンドウを表示
Debug.Watch4	Ctrl+Alt+W, 4	選んだ変数または監視式の値を表示するWatch4(監視)ウィンドウを表示

ヘルプ

これらのショートカットはヘルプ内の話題を見ることとそれらの中の移動用です。

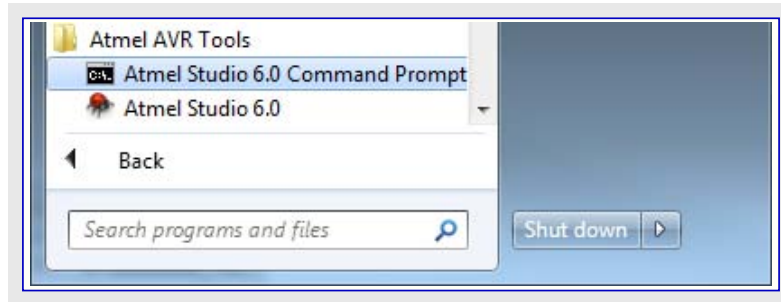
命令	ショートカット	説明
Help.F1Help	F1	フォーカスを持つユーザーインターフェースに対応するヘルプからの話題を表示
Help.ManageHelpSettings	Ctrl+Alt+F1	ヘルプ ライブラリ管理部を表示
Help.WindowHelp	Shift+F1	フォーカスを持つユーザーインターフェースに対応するヘルプからの話題を表示

11. コマンド行ユーティリティ (CLI)

Microchip Studioは`atprogram.exe`と呼ばれるコマンド行ソフトウェア ユーティリティが付属しています。



助言: 下図で示されるように、スタート⇒All programs⇒Microchip Studio⇒Microchip Studio Command Promptをクリックすることによって`atprogram`を走らせるためのパス構成設定を持つコマンド シェルを開始することができます。



`atprogram.exe`は以下に使うことができます。

- デバイスに`.bin`、`.hex`、`.elf`のファイルを書き込み
- 書き込みが正しいことを検証
- デバイスのメモリの読み込み、書き込み、消去
- ヒューズ、施錠ビット、保護ビット、使用者ページ、使用者識票の書き込み
- デバイスに製品ファイルを書き込み (**脚注**)
- 接続した全てのツールを一覧出力
- インターフェースとインターフェース クロック速度を設定

このユーティリティの使い方のヘルプを得るには`atprogram.exe`を実行してください。

これは標準出力に`atprogram CLIヘルプ`文章を打ち出します。

注: ELF製品ファイル形式は1つの単一ファイルでフラッシュ メモリとEEPROMの両方と使用者識票(XMEGAデバイスのみ)だけでなく、ヒューズと施錠ビットの構成設定の内容も保持することができます。この形式は実行可能/リンク可能形式(ELF:Executable and Linkable Form at)に基づきます。

製品ファイル形式は現在、tinyAVR、megaAVR、XMEGAに対して支援されます。このようなファイルを生成するためのプロジェクトの構成設定方法の記述については「[3.2.7.7. 他のメモリ形式を持つELFファイルの作成](#)」をご覧ください。

12. 良くある質問

Microchip Studioについての良くある質問

Atmel Studio 7またはMicrochip Studioの旧版からMicrochip Studioへの更新後に問題を経験

1. Microchip Studio上を右クリックして'Run as administrator(管理者として実行)'を選んでください。これはインストール中に構成部分が失敗した場合に失ったどの構成部品も復元することをMicrochip Studioに許します。その後にMicrochip Studioを閉じて標準アクセス権で開始してください。
2. 以下のフォルダを削除することによって旧版でキャッシュされたStudioデータを削除してください。
`%localappdata%\Atmel\AtmelStudio\7.0`
`%appdata%\Atmel\AtmelStudio\7.0`
 これは更新された版で矛盾するかもしれないウィンドウ配置のようなキャッシュされた構成設定を削除します。
3. Microchip Studioを次のように再インストールしてください。
 1. 管理者としてコマンド プロンプトを実行してください。
 2. Studioインストール ディレクトリ、例えば、`C:\Program Files (x86)\Atmel\Studio\7.0`へ行ってください。
 3. `>StudioInstallAgent.exe /uninstall`
 4. `>StudioInstallAgent.exe /install`

Atmel USBドライバは何?

Atmel USBドライバは全てのツールに必要とされるUSBドライバを同梱する集約的なインストーラです。

Atmel USBドライバ一括のインストール中に異常を受け取る

Atmel USBドライバ一括のインストール中に異常の0x800b010a - A certificate chain could not be built to a trusted root authority(一連の証明書が信頼されるルート権限に構築することができません。)を受け取るかもしれません。これはインストーラに署名した証明書がWindowsに対して証明書権限構築を使って検証することができないことを意味します。

証明書を検証できない理由は一連の証明書がWindows Updateを通して更新されることが必要だからです。Windowsが証明書を検証することができるように、全ての更新を受け取ることを確実にしてください。

コンピュータがオフラインまたは何処かしら制限されているためにコンピュータを更新することができない場合、ルート証明書更新は<http://support2.microsoft.com/kb/931125>からダウンロードすることができます。

Microchip StudioはAtmel Studio、AVR Studio、AVR32 Studioの旧版と並行して動きますか?

Microchip StudioはAtmel Studio 7を更新し、故にAtmel Studio 7と並行で動きません。

Microchip StudioはAtmel Studio 6.2とより古い版と並行で動きます。

Norton AntiVirus走行時にMicrochip Studioがデバッグや書き込み器を見つけられません。

Norton AntiVirusが走っている場合、Microchip Studioは接続したどのツールも見えないかもしれません。Norton AntiVirusを確実に動くようにするには、プログラムを許すNorton AntiVirus内の例外として`atbackend.exe`を追加することによって`atprogram.exe`にMicrochip Studioとの通信を許してください。これは既定防止ポートによるどのアンチウイルスプログラムでも同じです。

Microchip Studioインストーラを走らせようとする時にWindowsが以下のメッセージを持つメッセージ枠を示します。”Windows cannot access the specified device, path or file. You may not have the appropriate permissions to access the item. (Windowsは指定されたデバイス、パス、またはファイルにアクセスすることができません。あなたはその項目にアクセスするための適切な許可を持っていないかもしれません。)”

これはMicrochip Studioのインストールを妨げるアンチウイルスプログラムによって起こされるかもしれません。私たちはこれをSophos antivirus一括で見ました。マシンで走行しているSophosサービス(または対応する何れかのアンチウイルス サービス)を一時的に禁止してインストールを再び開始してください。

Microchip Studioが開始に非常に長い時間かかるけれども、仮想マシン(VM)環境では上手く動きます。

Visual Studioシェル(従ってMicrochip Studio)は始動中にかなり大量の処理を行います。この操作の一部は更新されたグラフィックドライバとドライバから大いに恩恵を受けるWPF操作です。最新のグラフィックドライバのインストールが通常動作中と始動中の両方で性能増加を与えるかもしれません。

STK500使用時に直列ポート緩衝部溢れ異常メッセージで度々書き込みと検証が失敗します。

これは既知の問題です。DPC遅延のため、直列通信はUARTチップセットで緩衝部溢れを持ち得ます。殆どのシステムに対して動く対策はUSB-シリアルアダプタを使うことです。

ゲストアカウントからの開始時、Microchip Studio開始時に以下の異常が表示されます。”Exception has been thrown by the target of an invocation(呼び出しの目的対象によって例外が投げられました。)”

Microchip Studioはゲストアカウント下でインストールも走行もしません。

仮想マシンでMicrochip Studioをインストールして動かすことができますか？

はい、シミュレータで全く問題があるべきではありません。けれども、デバッグや書き込み器のような物理的な装置でVMは物理的なUSBとシリアルポートの接続に対する支援を提供しなければなりません。

どのようにしてMicrochip Studioの始動時間を減らすことができますか？

- 不要な拡張のアンインストールを確実にしてください。
- **Allow Add-in components to load**(読み込むアドイン構成部品を許す)を禁止してください。
 - a. **Tools**(ツール)⇒**Options**(任意選択)⇒**Add-in/Macro Security**(アドイン/マクロ保護)へ行ってください。
 - b. その後**Allow Add-in components to load**(読み込むアドイン構成部品を許す)任意選択のチェックを外してください。
- 始動頁を禁止してください。
 - a. **Tools**(ツール)⇒**Options**(任意選択)⇒**Environment**(環境)⇒**Startup**(始動)⇒**At Startup**(始動で)へ行ってください。
 - b. **Show empty environment**(空の環境を表示)任意選択を選んでください。

Windowsの何れかの支援される版に対してStudio性能を改善する方法は？

- システムが最新版のWindows自動化APIを持つことを確実にしてください。
- アンチウイルス走査部から以下のディレクトリとファイルを除外してください。
 - Microchip Studioインストール ディレクトリとその内側の全てのフォルダと全てのファイル
 - **%AppData%\Roaming\Atmel**ディレクトリとその内側の全てのフォルダと全てのファイル
 - **%AppData%\Local\Atmel**ディレクトリとその内側の全てのフォルダと全てのファイル
 - あなたのプロジェクト ディレクトリ
- Visual Studioシェルはたくさんのスワップ空間が必要です。ページング ファイルを増やしてください。また、システムを最大性能に置いてください。両方の任意選択はメニューの**Performance**(性能)⇒**Properties**(プロパティ)⇒**System**(システム)⇒**Settings**(設定)で見つかります。

最新のWindows自動化API 3.0をインストールすべきですか？

はい、OSが以下のどれかの場合。

- Windows Server 2008

どのようにしてシステムが最新のWindows自動化API 3.0を持つようになりますか？

Windows 7またはWindows 8を持つ場合、システムは最新版のWindows自動化APIを持ちます。Windows XP、Windows Vista®、Windows Server® 2003、Windows Server 2008だけが旧版のAPIを持ちます。システム内で(通常、**Windows**フォルダで見つかる)**UIAutomationCore.dll**ファイルを見つけて、そのファイルの版番号を比べてください。新しいAPI用の版は**7.X.X.X**であるべきです。最新APIはsupport.microsoft.com/kb/971513で見つけることができます。

プロジェクトが大きくて開くのにか長い時間がかかります。この遅れを避ける何か任意選択はありませんか？

Visual Assist Xは既存プロジェクトを開く時に全てのファイルを解析します。この任意選択を禁止することができます。

1. **Performance**(性能)⇒**Visual Assist X Options**(ビジュアル アシストX任意選択)⇒**VAssistX**へ行ってください。
2. **Parse all files when opening the project**(プロジェクトを開く時に全てのファイルを解析)任意選択のチェックを外してください。

システムで制限されたRAMの量を持ち、同じ実体のMicrochip Studioで長時間動かします。その後暫くしてシステムでMicrochip Studioが遅くなります。

Microchip Studioにゴミ収集(ガベージコレクション)を強制するために**Ctrl+Shift+Atl+F12**を2度押してください。

どのようにしてプロジェクトをより速く構築させられますか？

Tools(ツール)⇒**Options**(任意選択)⇒**Buider**(構築部)⇒**GNU Make**⇒**Make Parallel Execution Of Build**(構築の並列実行を作成)から並列構築任意選択を許可することができます。

12.1. 過去のAVR®ソフトウェアと第三者製品との互換性

12.1.1. デバッグ用に外部ELFファイルをインポートするには？

File(ファイル)⇒**Open object file for debugging**(デバッグ用にオブジェクト ファイルを開く)を使ってください。

12.1.2. AVR® Studio 4のプロジェクトを新しいMicrochip Studioで再利用するには？

1. メニューで**File**(ファイル)⇒**Import AVR Studio 4 project**(AVR Studio 4プロジェクトをインポート)をクリックしてください。
2. “**Import AVR Studio 4 project**(AVR Studio 4プロジェクトをインポート)”ダイアログが現れます。
3. プロジェクト名を入力するか、または**APFS File location**(APFSファイル位置)タブの**Browse**(検索)鈕をクリックすることによってプロジェクト位置を検索してください。
4. **Solution Folder**(解決策フォルダ)タブでプロジェクトの変換の結果となる新しい解決策に名前を付けてください。

5. **Next**(次へ)をクリックしてください。
6. Microchip Studioは変換を続けます。プロジェクトの複雑さと特殊性に依存していくつかの警告や異常があるかもしれません。それらは**Summary**(要約)ウィンドウで示されます。
7. 新しく変換されたプロジェクトにアクセスするため**Finish**(終了)をクリックしてください。

12.2. Microchip Studioインターフェース

12.2.1. コードのデバッグを開始するには? デバッグのためのキーボード ショートカットは何?

AVR Studio 4とは異なり、デバッグを始めることなくプロジェクトを構築することができます。

ソース ファイルへの変更後にプロジェクトの再構築が必要な場合、**Ctrl+Alt+F7**を押してください。

デバッグを開始するには**F5**を押してください。

直接走行せずにデバッグ インターフェースを開くには**Debug**(デバッグ)⇒**Start Debugging and Break**(デバッグ開始と中断)メニュー鈕を押すか、または**F11**を押してください。

行単位のデバッグを開始するには**F10**を押し、命令単位のデバッグ作業を開始するには**F11**を押してください。

デバッグなしでプロジェクトを走らせるには**Debug**(デバッグ)⇒**Start Without Debugging**(デバッグなしで開始)メニュー鈕を押してください。

12.2.2. 解決策は何?

解決策はMicrochip Studioでプロジェクトを編制するための構造です。解決策は**.sln**(テキストに基づく、共用)と**.suo**(2進数、使用者指定解決策任意選択)のファイルでプロジェクトに対する状態情報を保持します。

12.2.3. プロジェクトは何?

プロジェクトはプロジェクトに含まれる全てのソース ファイル、全てのインクルードされるライブラリ、全ての構築実行可能物に対する参照を含む論理的なフォルダです。プロジェクトは複雑な応用に対して繋ぎ目のないコードの再利用と構築処理の容易な自動化を許します。

12.2.4. プロジェクトに外部Makefileを使うには?

外部**makefile**と他のプロジェクト任意選択の使い方はプロジェクト プロパティで構成設定することができます。

外部**makefile**は動くためにMicrochip Studioによって必要とされる規則を含まなければならないことを憶えて置いてください。

12.2.5. 変数監視時、デバッグが「Optimized away(最適化で除去)」と言う

今日の殆どのコンパイラは最適化するコンパイラとして知られているものです。これはプログラムの大きさを減らしたり、速度を上げたりするためにいくつかの策略を使います。

注: この動きは通常、**-On**スイッチによって制御されます。

この異常の発生は通常、何も無いコードの部分をデバッグしようとすることです。以下の例で変数**a**を監視しようとするのがこの動きを起こすかもしれません。

```
int main() {
    int a = 0;
    while (a < 42) {
        a += 2;
    }
}
```

aの増加がコードの他のどの部分にも影響を及ぼさないため、最適化で除去されるべき**a**についての理由は明らかです。この例の多忙待ち繰り返しはこの事実に気付かない場合に予期せぬ動きの最も重要な例です。

これを修正するにはコンパイル中に使われる最適化レベルをより下げるか、またはなるべくなら**a**を**volatile**として宣言してください。変数が**volatile**宣言されるであろう別の状況は、或る変数がコードと割り込み処理ルーチン(ISR:Interrupt Service Routine)間で共用される場合です。

この問題の徹底的な検証についてはこの問題に於ける**クリフ ローソンの優秀な解説**を見てください。

12.2.6. デバッグ作業開始時、「Debug Tool is not Set(デバッグ ツールが設定されていない)」の異常状態になる

このメッセージの理由はプロジェクトをデバッグする能力がある選ばれたツールがないことです。ツールが選ばれない時にそれ空です。引き落としメニューをクリックすることが利用可能な全てのツールを示します。Tool project(ツール プロジェクト)枠へ行って支援されるツールを変更してください(**Project Properties**(プロジェクト プロパティ)⇒**Tool**(ツール)⇒**Select Debugger/Programmer**(デバッグ/書き込み器の選択))。

選んだツールがデバッグを支援しない場合、正しいインターフェースが選ばれていることと、周波数が仕様に従っていることを調べてください。問題が持続する場合、プログラミングが安定する周波数に対してより低い周波数を試みて、その後に安定が保たれる限り緩やかに増加してください。

12.3. 性能の問題

12.3.1. Microchip StudioがPCでの開始に非常に長い時間かかるが、仮想環境では上手く走行。何かできることがある？

Visual Studioシェル(従ってMicrochip Studio)はグラフィック ライブラリとしてWPFを使ってGUIスレッドで大量の処理を行います。WPFはハードウェア加速に対する支援を持ちます。いくつかのグラフィック カードドライバはこれを上手く利用できず、グラフィックの更新が全く必要とされない時でさえ、カーネル空間で時間を費やします。最新のグラフィックドライバのインストールが性能増加を与えるかもしれません。

12.3.2. STK500使用時に度々「Serial Port Buffer Overrun Error」でプログラミングと確認が失敗する。

これは既知の問題です。直列通信の割り込みDPC遅延は他のドライバによって中断されるかもしれず、故にUARTチップセットで緩衝部溢れを起こします。殆どのシステムに対して動く対策はUSB-シリアル アダプタを使うことです。

12.3.3. USBハブを通してツールを接続し、プログラミングとデバッグ中に異常メッセージと矛盾する結果になる。

ツールとデバイスはおそらくデバッグをするPCのUSBポートに直接接続されるでしょう。これが任意選択でなければ、以下によって問題を減らす/なくすかもしれません。

- ・ ハブに接続された他のどのUSB装置も切断してください。
- ・ USBハブ上のポートを替えてください。
- ・ ツール クロック周波数を低く設定してください。例えば、JTAGクロック<600kHzに設定
- ・ ツール/デバイスの組み合わせに対してUse external reset(外部リセットを使用)が任意選択なら、これを許可してください。

注: AVR Dragonはおそらく給電されたUSBハブを通して接続されるでしょう。これはマザーボードが十分な電力を提供しない場合にDragon上の電力供給が弱くなり過ぎになり得るためです。Dragonが時間超過や固まる場合ハブが低品質のためかもしれません。

12.4. ドライバとUSBの問題

12.4.1. Microchip Studioでツールを認証させるには？

これはおそらく自動的に起こるでしょうが、時々Windows®ドライバが正しくツールを認証しません。これを修正するにはWindowsのデバイス マネージャでツールがMicrochip節点下で一覧にされることを調べなければなりません。ツールが一覧にされない場合、Unknown devices(未知の装置)下でそれを探してみてください。そこで見つけたなら、そのツールをダブル クリックし、Driver(ドライバ)タブをクリックしてUpdate Driver(ドライバ更新)を選ぶことによってそのドライバを再インストールしてみてください。Windowsにそのドライバを探させてください。おそらくドライバが再インストールされてツールがMicrochip下に表示されるでしょう。今や、おそらくツールはMicrochip Studioから使用可能でしょう。

12.4.2. 仮想マシンでファームウェア更新失敗または不安定

殆どのツールは標準動作形態からファームウェア更新動作形態へ切り替えることを問われた時にリセットを実行します。これはツールをUSBでの再列挙(接続認証)を強制します。仮想ソフトウェアは再列挙後の再取り付けに失敗し、切断されたツールになるかもしれません。

通常の仮想化ソフトウェアは与えられたゲスト オペレーティング システムに対して自動的な取り付けを望むUSB装置の集合を設定するUSB選別の考えを支援します。これがどう行われるかを見るにはその仮想化解決策用の手引書を調べるか、または「12.4.4. VirtualBoxでファームウェア更新失敗」をご覧ください。

12.4.3. 仮想マシン下でデバッグが決して中断しない

いくつかの仮想化解決策はそれが支援するUSBエンドポイントの数の制限を持ちます。これはエンドポイント数がツール用に必要とされる数よりも少ない場合に問題になるかもしれません。通常これはプログラミングを意図されるように動かしますが、デバッグ事象がより多くのエンドポイント数で送信されるため、デバッグは動きません。

仮想化ソフトウェアがどれ位のエンドポイントが利用可能かと、これに関する仮想化ソフトウェアでのエンドポイント特有の問題を調べてください。

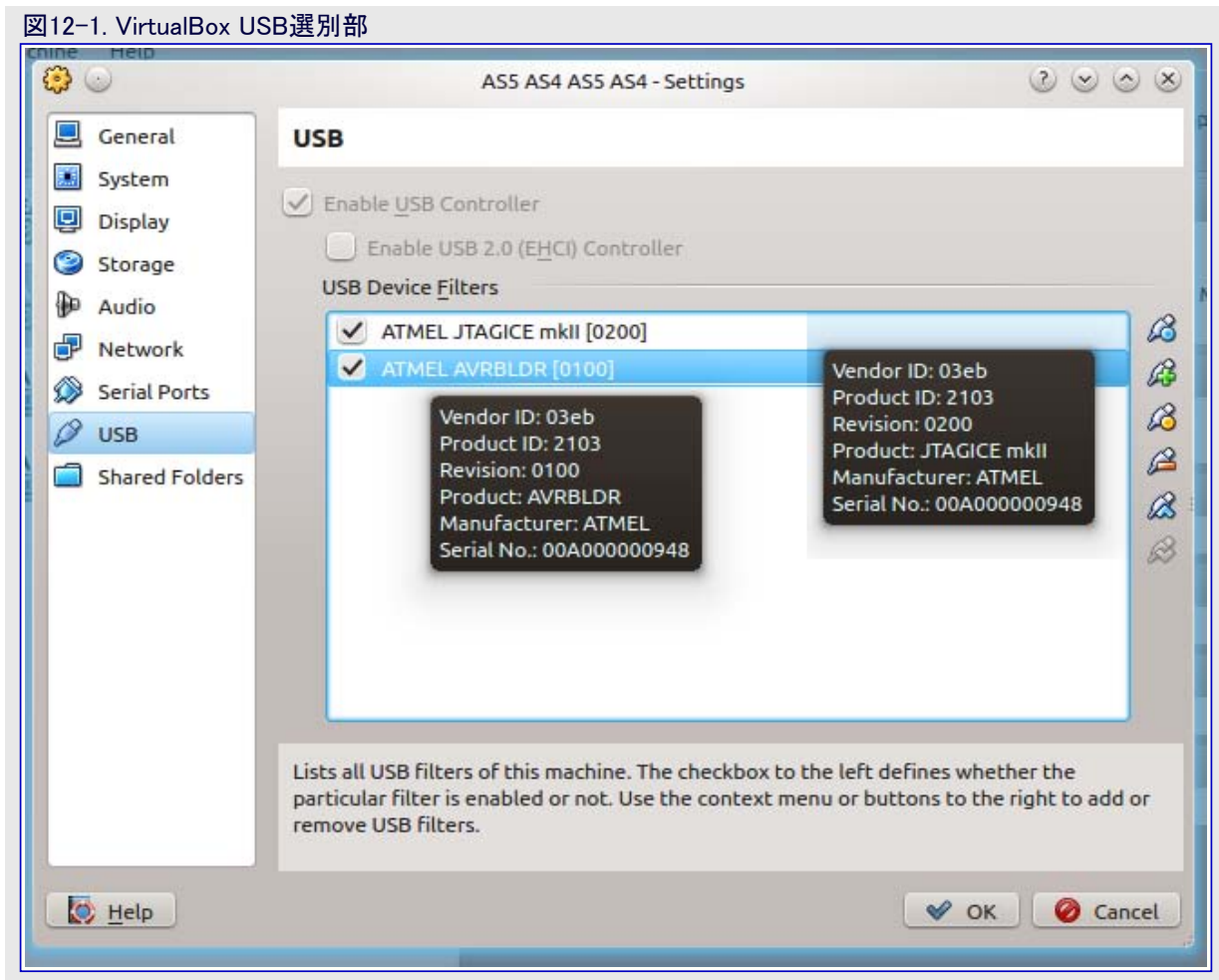
12.4.4. VirtualBoxでファームウェア更新失敗

どれかのツールでファームウェア更新を行っている時に、ツールは普通の操作中に使われるのとは違う別の動作形態で再接続されることが必要です。これはツールに再列挙(接続認証)をさせ、VirtualBox実体から切断されてホスト オペレーティング 枠へ戻ることをツールにさせ得ます。

VirtualBox実体へ自動的にツールを接続させるにはUSB選別部の対を構成設定することが必要です。USB選別部のより多くの情報はVirtualBox資料で見つけることができます。

次図で示される2つと同様に2つの選別部を作ってください。

図12-1. VirtualBox USB選別部



上の図の例がJTAGICE mkIIに対する指定であることに注意してください。ツールに対して1つの登録があり、ここではJTAGICE mkII、それとそのツールに対するファームウェア更新動作であるAVRBLDR用の1つです。名前、通番、供給者ID、製造者IDがあなたのツールと違うかもしれませんが、故にそれによってそれらの値を変更してください。

注: この項はVirtualBoxへの指定を含みます。同じ論理は他の仮想化ソフトウェアに適用されますが、手順が違うかもしれません。

12.4.5. Arm®互換ツールでの問題

いくつかの稀な例では全てのArm互換ツールがMicrochip Studioから消えます。これはWindowsの各種版で使われる異なるdll読み込み方法が(原因として)突き止められました。

それがdll異常であるかを調べるには`atprogram`を用いてチップ情報を読み出してみてください。Microchip Studio内のTools(ツール)メニューまたはスタートメニューからMicrochip Studioコマンドプロンプトを開くことによってこれを行ってください。コマンドプロンプトで以下の命令を入力してそれが失敗しないことを調べてください。

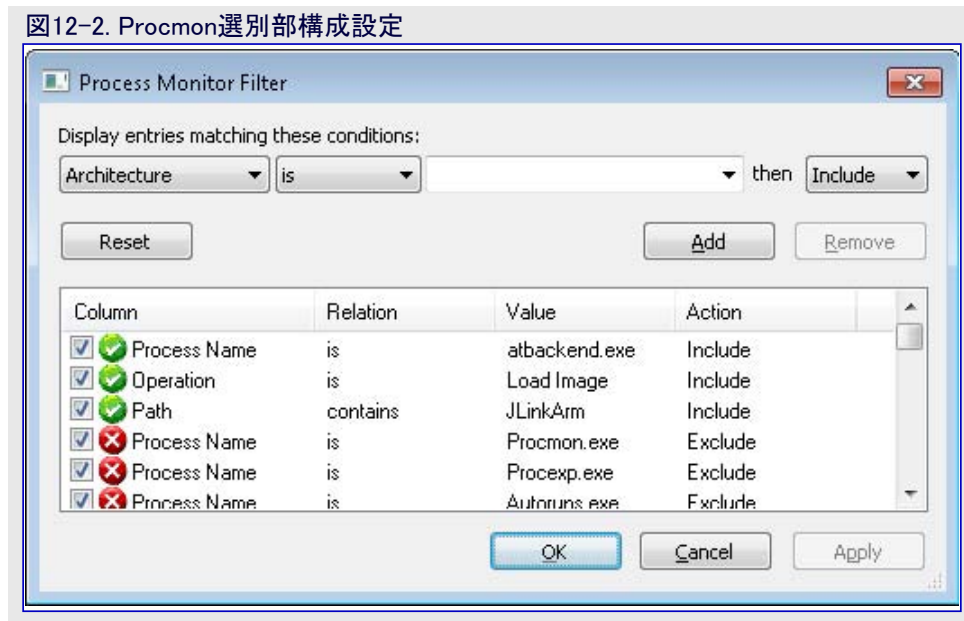
```
atprogram -t <tool> -i <interface> -d <device> info
```

上の断片に於いて、`<tool>`はツール名、例えば、`atmelice`、`samice`、`edbg`で置き換えてください。同様に、`<interface>`を使うインターフェースで、`<device>`をデバイス名、例えば、`atsam3s4c`で置き換えてください。

上の命令の実施はおそらくメモリ配置、チップに対する供給電圧、ヒューズ設定についての情報を出力するでしょう。失敗した場合、それはおそらく「12.4. ドライバとUSBの問題」によって網羅されるドライバの問題です。

`atprogram`がデバイスと通信することができるなら、この問題は多分、読み込み部順位ののために読み込まれつつある`JLinkArm.dll`の不正な版であることを意味します。これを調べるにはどのdllが読み込まれつつあるかを調べるための`Procmon`ツールを使ってください。

`Procmon`ツールをダウンロードして次図で示される選別部を構成設定してMicrochip Studioを開始してください。Microchip Studioが開始された後の数秒で、おそらくdllが読み込まれつつあるパスを示す1行が見えるようになるでしょう。おそらくそれはMicrochip Studioインストール ディレクトリ内の`atbackend`フォルダから読み込まれるでしょう。



dllのパスが違う場合、Microchip Studioが不正なdllを手に取り、このdllがMicrochip Studioと共に出荷されたdllと非互換であることを意味します。この例は下図で示されます。



上の問題を解決するため、読み込まれつつあるdllをバックアップしてその後にMicrochip Studioのインストール ディレクトリ内のatbackendフォルダで見つかるJLinkArm.dllでそれを置き換えることが推奨されます。これはMicrochip Studioに同梱されたdllが読み込まれつつあったものよりも新しく、そのdllが後方互換と言う与えられた前提で行うことができます。

注: それを配置したプログラムとの互換性がないとした場合のため、それを置き換える前に問題のJLinkArm.dllをバックアップすることを覚えて置いてください。

13. 文書改訂履歴

文書改訂	日付	注釈
42167A	2016年7月	初版文書公開
42167B	2016年9月	「Power Debugger」項を追加
A	2018年2月	Microchip形式に変換してAtmel資料番号42167を置き換え。資料全体を通して多数の修正が行われました。
B	2018年5月	項更新: Atmel Studio 7、STARTとソフトウェア内容
C	2019年1月	項追加: Percepio Tracealyzer
C1	2020年12月	MPLAB XC8ツールチェーン支援。製品をAtmel StudioからAVRとSAMデバイス用Microchip Studioに改名。
D	2021年11月	MCCからのプロジェクトのインポートを支援

Microchipウェブ サイト

Microchipはwww.microchip.com/で当社のウェブ サイト経由でのオンライン支援を提供します。このウェブ サイトはお客様がファイルや情報を容易に利用可能にするのに使われます。利用可能な情報のいくつかは以下を含みます。

- **製品支援** – データシートと障害情報、応用記述と試供プログラム、設計資源、使用者の手引きとハードウェア支援資料、最新ソフトウェア配布と保管されたソフトウェア
- **一般的な技術支援** – 良くある質問(FAQ)、技術支援要求、オンライン検討グループ、Microchip設計協力課程会員一覧
- **Microchipの事業** – 製品選択器と注文の手引き、最新Microchip報道発表、セミナーとイベントの一覧、Microchip営業所の一覧、代理店と代表する工場

製品変更通知サービス

Microchipの製品変更通知サービスはMicrochip製品を最新に保つのに役立ちます。加入者は指定した製品系統や興味のある開発ツールに関連する変更、更新、改訂、障害情報がある場合に必ず電子メール通知を受け取ります。

登録するにはwww.microchip.com/pcnへ行って登録指示に従ってください。

お客様支援

Microchip製品の使用者は以下のいくつかのチャネルを通して支援を受け取ることができます。

- 代理店または販売会社
- 最寄りの営業所
- 組み込み解決技術者(ESE:Embedded Solutions Engineer)
- 技術支援

お客様は支援に関してこれらの代理店、販売会社、またはESEに連絡を取るべきです。最寄りの営業所もお客様の手助けに利用できます。営業所と位置の一覧はこの資料の後ろに含まれます。

技術支援はwww.microchip.com/supportでのウェブ サイトを通して利用できます。

Microchipデバイス コード保護機能

Microchip製品での以下のコード保護機能の詳細に注意してください。

- Microchip製品はそれら特定のMicrochipデータシートに含まれる仕様に合致します。
- Microchipは動作仕様内で意図した方法と通常条件下で使われる時に、その製品系統が安全であると考えます。
- Microchipはその知的所有権を尊重し、積極的に保護します。Microchip製品のコード保護機能を侵害する試みは固く禁じられ、デジタル ミレニアム著作権法に違反するかもしれません。
- Microchipや他のどの半導体製造業者もそのコードの安全を保証することはできません。コード保護は製品が”破ることができない”ことを当社が保証するということを意味しません。コード保護は常に進化しています。Microchipは当社製品のコード保護機能を継続的に改善することを約束します。

法的通知

この刊行物と契約での情報は設計、試験、応用とのMicrochip製品の統合を含め、Microchip製品でだけ使えます。他の何れの方法でのこの情報の使用はこれらの条件に違反します。デバイス応用などに関する情報は皆さまの便宜のためにだけ提供され、更新によって取り換えられるかもしれません。皆さまの応用が皆さまの仕様に合致するのを保証するのは皆さまの責任です。追加支援については最寄りのMicrochip営業所にお問い合わせ頂くか、www.microchip.com/en-us/support/design-help/client-support-servicesで追加支援を得てください。

この情報はMicrochipによって「現状そのまま」で提供されます。Microchipは非侵害、商品性、特定目的に対する適合性の何れの黙示的保証やその条件、品質、性能に関する保証を含め、明示的にも黙示的にもその情報に関連して書面または表記された書面または黙示の如何なる表明や保証もしません。

如何なる場合においても、Microchipは情報またはその使用に関連するあらゆる種類の間接的、特別的、懲罰的、偶発的または結果的な損失、損害、費用または経費に対して責任を負わないものとします。法律で認められている最大限の範囲で、情報またはその使用に関連する全ての請求に対するMicrochipの全責任は、もしあれば、情報のためにMicrochipへ直接支払った料金を超えないものとします。生命維持や安全応用でのMicrochipデバイスの使用は完全に購入者の危険性で、購入者はそのような使用に起因する全ての損害、請求、訴訟、費用からMicrochipを擁護し、補償し、免責にすることに同意します。他に言及されない限り、Microchipのどの知的財産権下でも暗黙的または違う方法で許認可は譲渡されません。

商標

Microchipの名前とロゴ、Microchipロゴ、Adaptec、AnyRate、AVR、AVRロゴ、AVR Freaks、BesTime、BitCloud、CryptoMemory、CryptoRF、dsPIC、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemiロゴ、MOST、MOSTロゴ、MPLAB、OptoLyzer、PIC、picoPower、PICSTART、PIC32ロゴ、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SSTロゴ、Super Flash、Symmetricom、SyncServer、Tachyon、TimeSource、tinyAVR、UNI/O、Vectron、XMEGAは米国と他の国に於けるMicrochip Technology Incorporatedの登録商標です。

AgileSwitch、APT、ClockWorks、The Embedded Control Solutions Company、EtherSynch、Flashtec、Hyper Speed Control、Hyper Light Load、IntelliMOS、Libero、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plusロゴ、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、TrueTime、WinPath、ZLは米国に於けるMicrochip Technology Incorporatedの登録商標です。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、Augmented Switching、BlueSky、BodyCom、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、Espresso T1S、EtherGREEN、GridTime、IdealBridge、In-Circuit Serial Programming、ICSP、INICnet、Intelligent Paralleling、Inter-Chip Connectivity、JitterBlocker、Knob-on-Display、maxCrypto、maxView、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certifiedロゴ、MPLIB、MPLINK、MultiTRAK、NetDetach、NVM Express、NVMe、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、Ripple Blocker、RTAX、RTG4、SAM-ICE、Serial Quad I/O、simpleMAP、SimpliPHY、SmartBuffer、SmartHLS、SMART-I.S.、storClad、SQI、SuperSwitcher、SuperSwitcher II、Switchtec、SynchroPHY、Total Endurance、TSHARC、USBCheck、VariSense、VectorBlox、VeriPHY、ViewSpan、WiperLock、XpressConnect、and ZENAは米国と他の国に於けるMicrochip Technology Incorporatedの商標です。

SQTPは米国に於けるMicrochip Technology Incorporatedの役務標章です。

Adaptecロゴ、Frequency on Demand、Silicon Storage Technology、Symmcom、Trusted Timeは他の国に於けるMicrochip Technology Inc.の登録商標です。

GestICは他の国に於けるMicrochip Technology Inc.の子会社であるMicrochip Technology Germany II GmbH & Co. KGの登録商標です。

ここで言及した以外の全ての商標はそれら各々の会社の所有物です。

© 2021年、Microchip Technology Incorporatedとその子会社、不許複製

品質管理システム

Microchipの品質管理システムに関する情報についてはwww.microchip.com/qualityを訪ねてください。

日本語© HERO 2022.

本使用者の手引きはMicrochipのMicrochip Studio使用者の手引き(DS50002718D-2021年11月)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意識されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には()内に英語表記や略称などを残す形で表記しています。

青字の部分はリンクとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。

世界的な販売とサービス

米国	亜細亜/太平洋	亜細亜/太平洋	欧州
本社 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術支援: www.microchip.com/support ウェブアドレス: www.microchip.com	オーストラリア - シドニー Tel: 61-2-9868-6733 中国 - 北京 Tel: 86-10-8569-7000 中国 - 成都 Tel: 86-28-8665-5511 中国 - 重慶 Tel: 86-23-8980-9588 中国 - 東莞 Tel: 86-769-8702-9880 中国 - 広州 Tel: 86-20-8755-8029 中国 - 杭州 Tel: 86-571-8792-8115 中国 - 香港特别行政区 Tel: 852-2943-5100 中国 - 南京 Tel: 86-25-8473-2460 中国 - 青島 Tel: 86-532-8502-7355 中国 - 上海 Tel: 86-21-3326-8000 中国 - 瀋陽 Tel: 86-24-2334-2829 中国 - 深圳 Tel: 86-755-8864-2200 中国 - 蘇州 Tel: 86-186-6233-1526 中国 - 武漢 Tel: 86-27-5980-5300 中国 - 西安 Tel: 86-29-8833-7252 中国 - 廈門 Tel: 86-592-2388138 中国 - 珠海 Tel: 86-756-3210040	インド - ハンガロール Tel: 91-80-3090-4444 インド - ニューデリー Tel: 91-11-4160-8631 インド - フネー Tel: 91-20-4121-0141 日本 - 大阪 Tel: 81-6-6152-7160 日本 - 東京 Tel: 81-3-6880-3770 韓国 - 大邱 Tel: 82-53-744-4301 韓国 - ソウル Tel: 82-2-554-7200 マレーシア - クアラルンプール Tel: 60-3-7651-7906 マレーシア - ペナン Tel: 60-4-227-8870 フィリピン - マニラ Tel: 63-2-634-9065 シンガポール Tel: 65-6334-8870 台湾 - 新竹 Tel: 886-3-577-8366 台湾 - 高雄 Tel: 886-7-213-7830 台湾 - 台北 Tel: 886-2-2508-8600 タイ - バンコク Tel: 66-2-694-1351 ベトナム - ホーチミン Tel: 84-28-5448-2100	オーストリア - ウェルス Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 デンマーク - コペンハーゲン Tel: 45-4485-5910 Fax: 45-4485-2829 フィンランド - エスポー Tel: 358-9-4520-820 フランス - パリ Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 ドイツ - ガルヒング Tel: 49-8931-9700 ドイツ - ハーン Tel: 49-2129-3766400 ドイツ - ハイムブロン Tel: 49-7131-72400 ドイツ - カールスルーエ Tel: 49-721-625370 ドイツ - ミュンヘン Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 ドイツ - ローゼンハイム Tel: 49-8031-354-560 イスラエル - ラーナナ Tel: 972-9-744-7705 イタリア - ミラノ Tel: 39-0331-742611 Fax: 39-0331-466781 イタリア - ハドバ Tel: 39-049-7625286 オランダ - デルフト Tel: 31-416-690399 Fax: 31-416-690340 ノルウェー - トロンハイム Tel: 47-72884388 ポーランド - ワルシャワ Tel: 48-22-3325737 ルーマニア - ブカレスト Tel: 40-21-407-87-50 スペイン - マドリッド Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 スウェーデン - イェテボリ Tel: 46-31-704-60-40 スウェーデン - ストックホルム Tel: 46-8-5090-4654 イギリス - ウォーキングハム Tel: 44-118-921-5800 Fax: 44-118-921-5820
アトランタ Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 オースチン TX Tel: 512-257-3370 ボストン Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 シカゴ Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 ダラス Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 デトロイト Novi, MI Tel: 248-848-4000 ヒューストン TX Tel: 281-894-5983 インディアナポリス Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 ロサンゼルス Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 ローリー NC Tel: 919-844-7510 ニューヨーク NY Tel: 631-435-6000 サンホセ CA Tel: 408-735-9110 Tel: 408-436-4270 カナダ - トロント Tel: 905-695-1980 Fax: 905-695-2078			