Atmel Studio 7とでの開始に際して

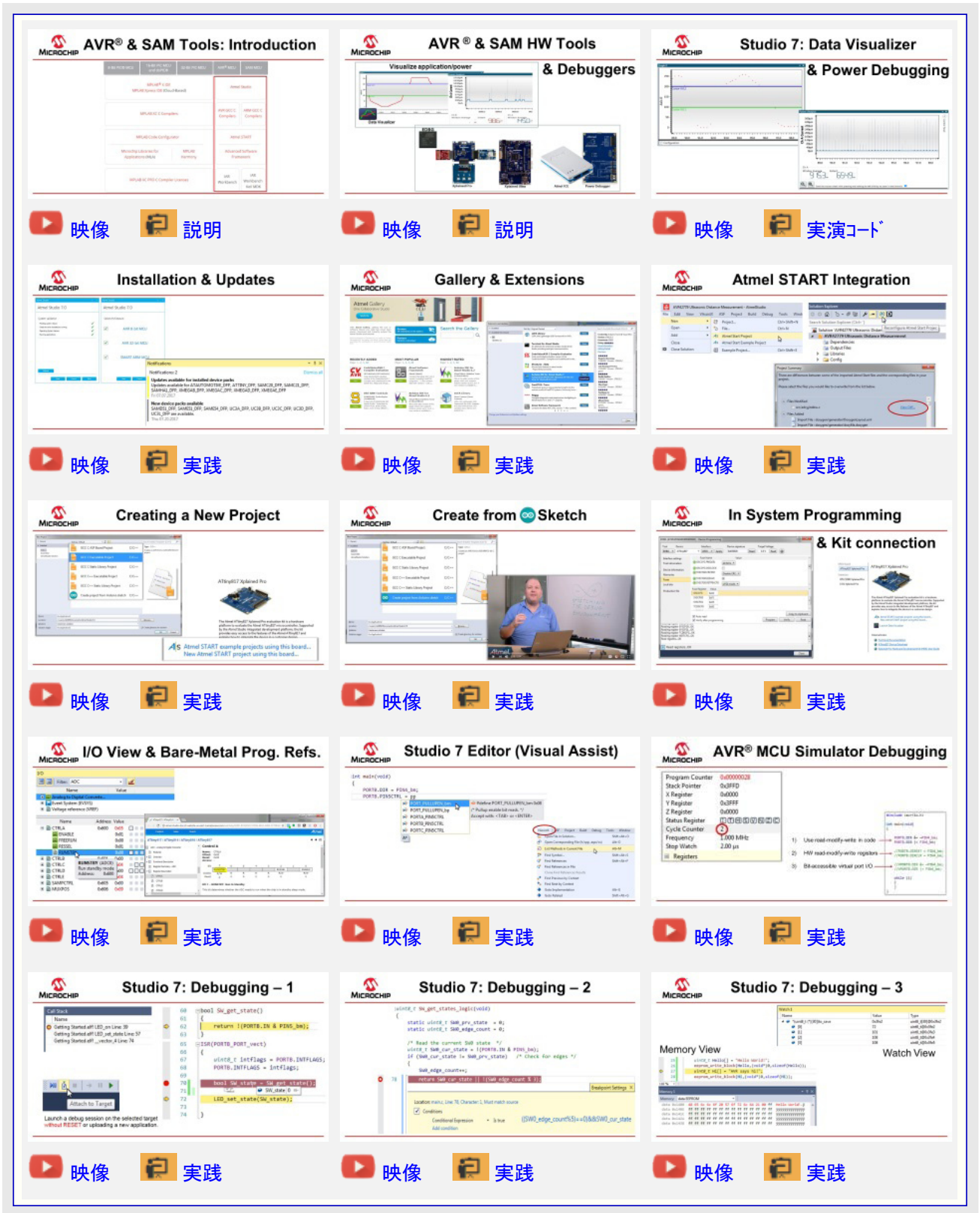
本書は一般の方々の便宜のため有志により作成されたもので、Microchip社とは無関係であることを御承知ください。しおりの[はじめに]での内容にご注意ください。

目次

1. 開始に際して 3
1.1. AVR®とSAMの開発ツール概要 4
1.2. AVR®とSAMのハードウェア ツールとデバッガ 6
1.3. データ可視器とPowerデバッガ実演 7
1.4. インストールと更新 9
1.5. Microchip陳列室とStudio拡張 11
1.6. Atmel START統合 12
1.7. 新規プロジェクト作成 15
1.8. Arduinoスケッチから作成 18
1.9. 実装書き込みとキット接続 18
1.10. I/O表示部と他の空からのプログラミング参考資料 22
1.11. エディタ: コードの記述と整理 (Visual Assist) 31
1.12. AVRシミュレータ デバッグ 38
1.13. デバッグ1: 中断点、段階実行、呼び出しスタック 42
1.14. デバッグ2: 条件付きと活動付きの中断点 48
1.15. デバッグ3: I/O表示部、メモリ表示部、監視 54
2. 改訂履歴 59
Microchipウェブ サイト 60
お客様への変更通知サービス 60
お客様支援 60
Microchipデバイス コード保護機能 60
法的通知 60
商標 61
DNVによって認証された品質管理システム 61
世界的な販売とサービス 62

1. 開始に際して

Atmel Studio 7の開始に際して – 再生一覧



このAtmel Studio 7用訓練開始は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。各項はその項を網羅する映像で始まります。

事前要件

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

ﾊｰﾄﾞｳｪｱ要件:

- ATtiny817 Xplained Pro
- 標準A-ﾏｲｸﾛB USBｹｰﾌﾞﾙ

ｿﾌﾄｳｪｱ要件:

- Atmel Studio 7.0
- avr-gccﾂｰﾙﾁｪｰﾝ
- tineyAVR®ﾃﾞﾊﾞｲｽ用最新部品ﾊﾟｯｸ

使うAtmel Studio 7.0ﾌﾟﾗｸﾞｲﾝ

- Atmwl START 1.0.113.0またはそれ以降
- ﾃﾞｰﾀ可視器(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。

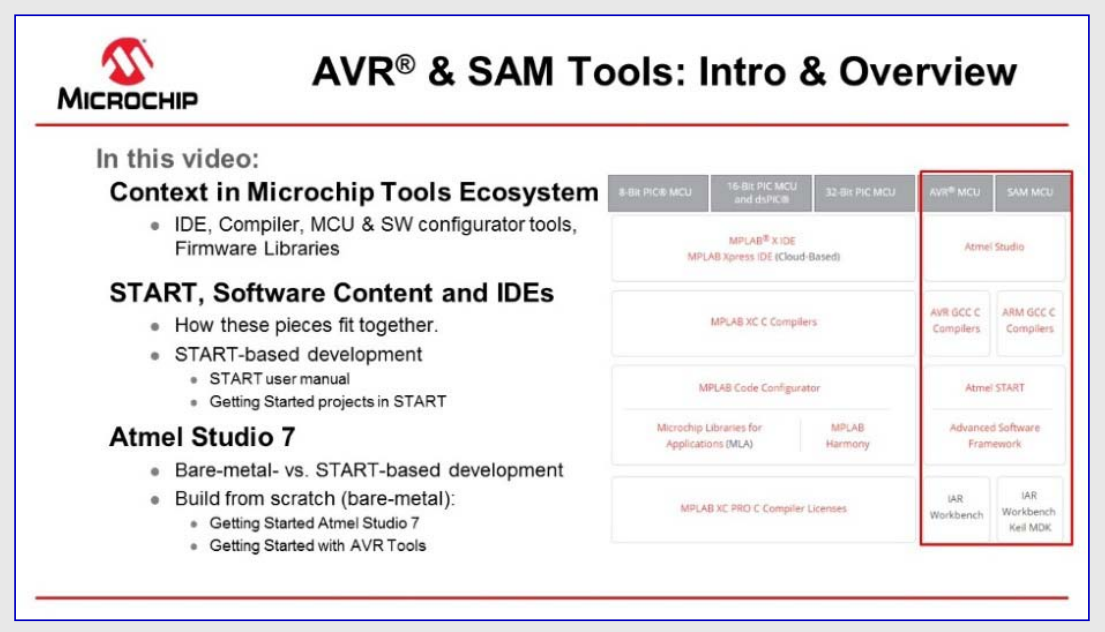
⚠警告 重要な情報を示します。

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。

1.1. AVR®とSAMの開発ﾂｰﾙ概要

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

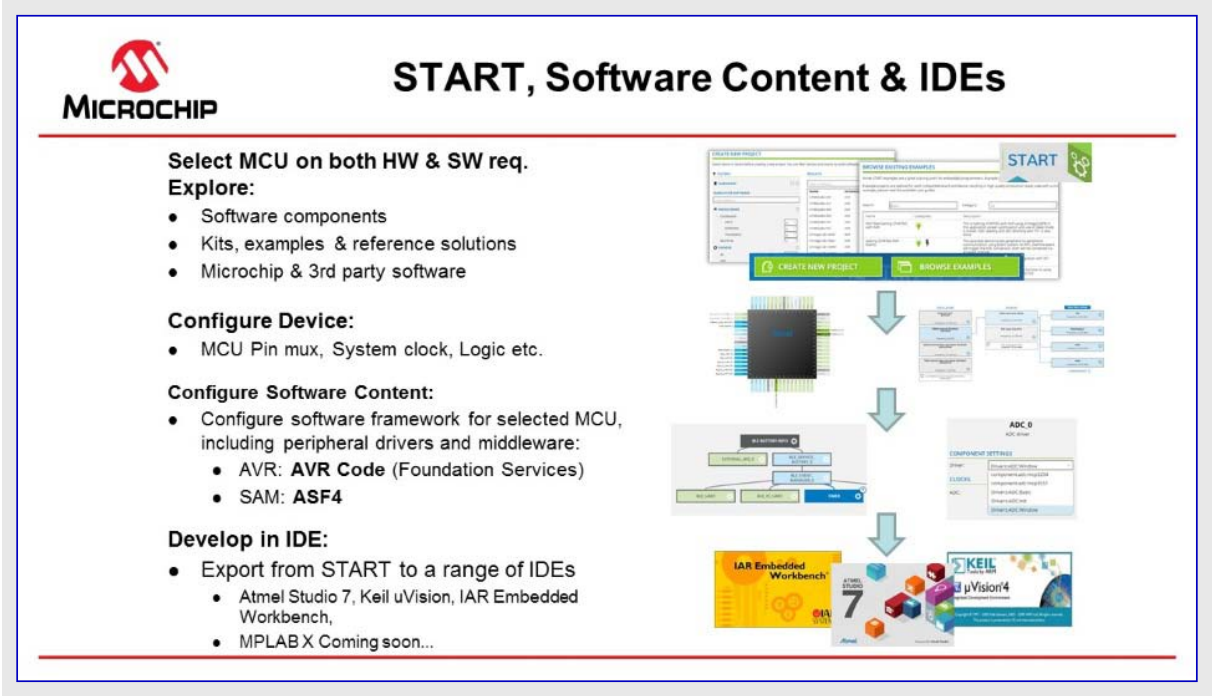


映像: AVRとSAMのﾂｰﾙ体系概要

Atmel STARTは様々なｿﾌﾄｳｪｱ枠組みに対するｳｪﾌﾞ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。新規ﾌﾟﾛｼﾞｪｸﾄまたは例ﾌﾟﾛｼﾞｪｸﾄのどちらかからの開始でも、Atmel STARTは便利で最適化した規則で、(**ASF4**と**Foundation Services**から)あなたの組み込み応用を仕立てるためのﾄﾞﾗｲﾊﾞやﾐﾄﾞﾙｳｪｱ(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、生成されたｺｰﾄﾞ ﾌﾟﾛｼﾞｪｸﾄをﾀﾞｳﾝﾛｰﾄﾞしてAtmel Studio 7、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は以下を許します。

- ｿﾌﾄｳｪｱとﾊｰﾄﾞ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。
- ﾄﾞﾗｲﾊﾞ、ﾐﾄﾞﾙｳｪｱ、例ﾌﾟﾛｼﾞ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。



高度なｿﾌﾄｳｪｱ枠組み(**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ﾄﾞﾊﾞｲﾊﾞとｺｰﾄﾞ単位部の豊富な組を提供します。これはﾄﾞﾗｲﾊﾞと高価値のﾐﾄﾞﾙｳｪｱを通してﾊｰﾄﾞ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 ｺｰﾄﾞ ﾗｲﾌﾞﾗﾘです。

SAM製品部門を支援する**ASF4**はASFの第4主要世代です。Cはﾒﾓﾘ量、ｺｰﾄﾞ性能を改善するだけでなく、Atmel STARTｳｪﾌﾞ使用者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と共に使われなければならず、これはASF2と3のASFｳｨｻﾞｰﾄﾞを置き換えます。

AVR製品部門を支援する**Foundation Services**は8ﾋﾞｯﾄと16ﾋﾞｯﾄのPIC MCUを支援する創設ｻｰﾋﾞｽと等価なAVR 8ﾋﾞｯﾄMCU用の簡単なﾌｧｰﾑｳｪｱ枠組みです。AVR Codeはｺｰﾄﾞ量とｺｰﾄﾞ速度だけでなく、ｺｰﾄﾞの簡潔性と信頼性に対しても最適化されています。**Foundation Services**はAtmel STARTによって構成設定されます。

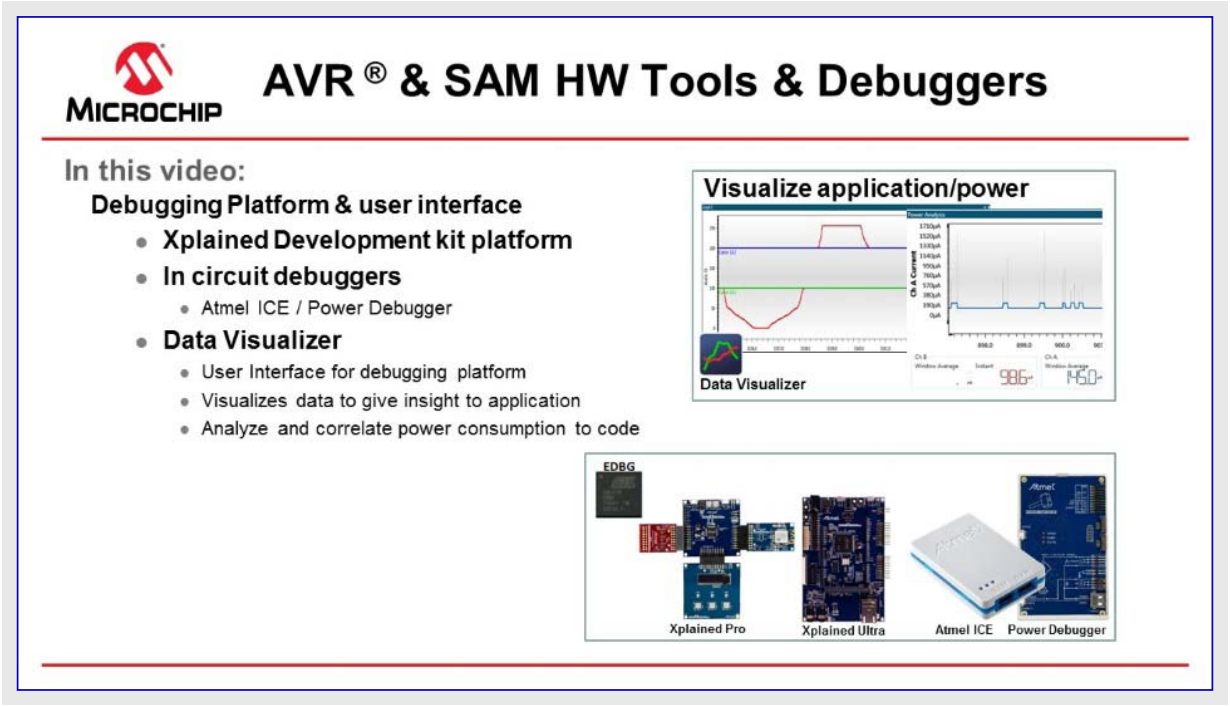
統合開発環境(**IDE**:Integrated Development Environment)はAtmel STARTで構成設定されてｴｸｽﾎﾟｰﾄされたﾄﾞﾗｲﾊﾞとﾐﾄﾞﾙｳｪｱのようなｿﾌﾄｳｪｱ部品に基づく応用の開発(または例応用の更なる開発)に使われます。

Atmel Studio 7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ﾃﾞﾊﾞｯｸﾞするための統合開発基盤(IDP:Integrated Development Platform)です。Atmel Studio 7 IDPはC/C++またはｱｾﾝﾌﾞﾘ言語のｺｰﾄﾞで書かれた応用を書いて構築してﾃﾞﾊﾞｯｸﾞ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のﾃﾞﾊﾞｲｽを支援するﾃﾞﾊﾞｯｶﾞ、書き込み器、開発ｷｯﾄに対しても継ぎ目なく繋げます。Atmel STARTとAtmel Stuido 7間の開発体験は最適化されています。Atmel Studio 7でのAtmel STARTに基づくﾌﾟﾛｼﾞ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。

1.2. AVR®とSAMのﾊｰﾄﾞｳｪｱ ﾂｰﾙとﾃﾞﾊﾞｯｶﾞ

本項はAVR®とSAM MCU用のﾊｰﾄﾞｳｪｱ ﾂｰﾙ体系を記述します。

開始に際しての話題



映像: AVR & SAMﾊｰﾄﾞｳｪｱ ﾂｰﾙ & ﾃﾞﾊﾞｯｶﾞ

ﾃﾞｰﾀ可視器 (Data Visualizer)

ﾃﾞｰﾀ可視器はﾃﾞｰﾀを処理して可視化するためのﾌﾟﾛｸﾞﾗﾑです。ﾃﾞｰﾀ可視器は組み込みﾃﾞﾊﾞｯｶﾞ ﾃﾞｰﾀ中継器ｲﾝﾀｰﾌｪｰｽ(DGI:Data Gateway Interface)とCOMﾎﾟｰﾄのような様々な供給元からﾃﾞ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、ｺｰﾄﾞ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。走行時のｺｰﾄﾞ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Atmel Studio 7用ﾌﾟﾗｸﾞｲﾝの両版が下のﾘﾝｸのｳｪﾌﾞｻｲﾄで入手可能です。

ｳｪﾌﾞｻｲﾄ: ﾃﾞｰﾀ可視器

Atmel-ICE

Atmel-ICEはUPDI、JTAG、PDI、ﾃﾞﾊﾞｯｸﾞ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SW D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 ｺﾝﾄﾛｰﾗのﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞ用の強力な開発ﾂｰﾙです。

Ateml-ICEはARM Cortex-Mに基づくSAMとﾁｯﾌﾟ上ﾃﾞﾊﾞｯｸﾞ能力を持つAVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗをﾌﾟﾛｸﾞﾗﾐﾝｸﾞしてﾃﾞﾊﾞｯｸﾞするための強力な開発ﾂｰﾙです。

ｳｪﾌﾞｻｲﾄ: Atmel-ICE

Powerﾃﾞﾊﾞｯｶﾞ

PowerﾃﾞﾊﾞｯｶﾞはUPDI、JTAG、PDI、ﾃﾞﾊﾞｯｸﾞ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SW D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 ｺﾝﾄﾛｰﾗのﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞ用の強力な開発ﾂｰﾙです。

加えて、Powerﾃﾞﾊﾞｯｶﾞ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ﾈﾙを持ちます。

PowerﾃﾞﾊﾞｯｶﾞはCDC仮想COMﾎﾟｰﾄだけでなく、SPI、UASRT、TWI、GPIO供給元からﾎｽﾄ ｺﾝﾋﾟｭｰﾀへ応用ﾃﾞｰﾀを流すためのﾃﾞｰﾀ中継器ｲﾝﾀｰﾌｪｰｽも含みます。

PowerﾃﾞﾊﾞｯｶﾞはAtmel Studio 7.0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互換ﾃﾞﾊﾞｯｶﾞです。Powerﾃﾞﾊﾞｯｶﾞは実時間分析のために電力測定と応用ﾃﾞﾊﾞｯｸﾞのﾃﾞｰﾀをﾃﾞｰﾀ可視器に流します。

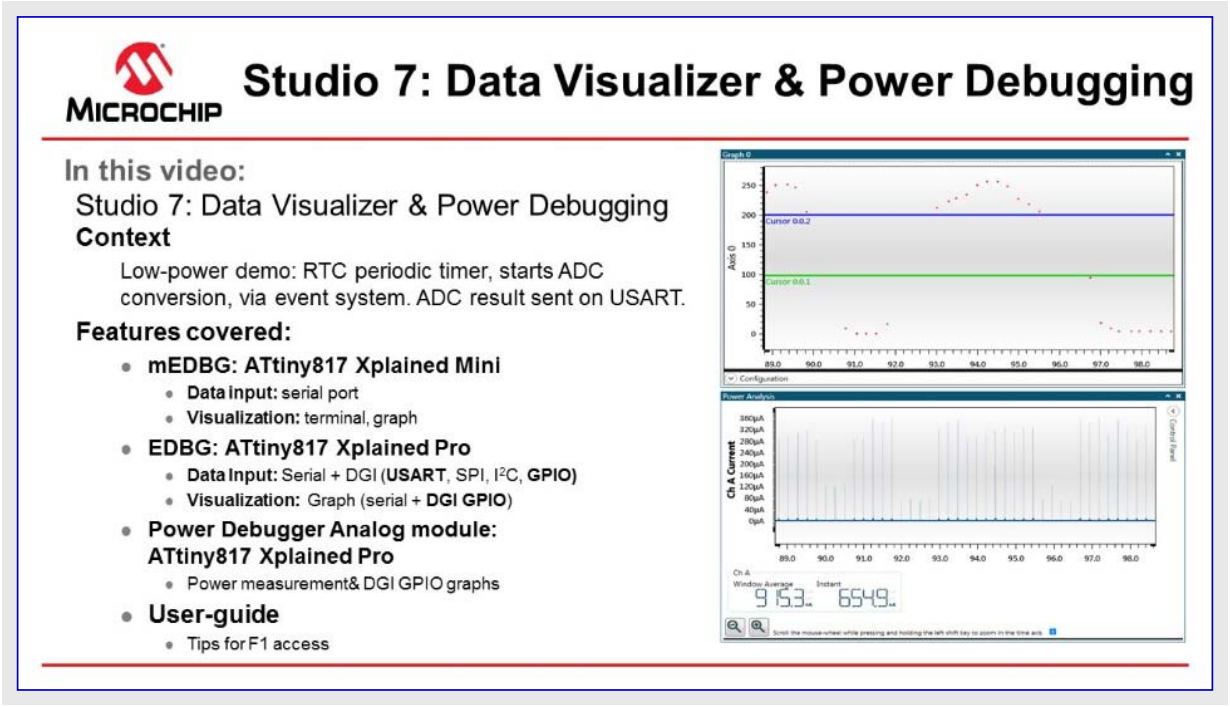
より多くの情報についてはｵﾝﾗｲﾝ使用者の手引きを訪ねてください。

ｳｪﾌﾞｻｲﾄ: Powerﾃﾞﾊﾞｯｶﾞ

1.3. ﾃﾞｰﾀ可視器とPowerﾃﾞﾊﾞｯｶﾞ実演

本項はPowerﾃﾞﾊﾞｯｶﾞを含みﾃﾞ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



映像: ﾃﾞｰﾀ可視器とPowerﾃﾞﾊﾞｯｶﾞ実演

```
/*
* Power_Demo_ADC_SleepWalking.c
* ﾃﾞﾊﾞｲｽ/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)
```

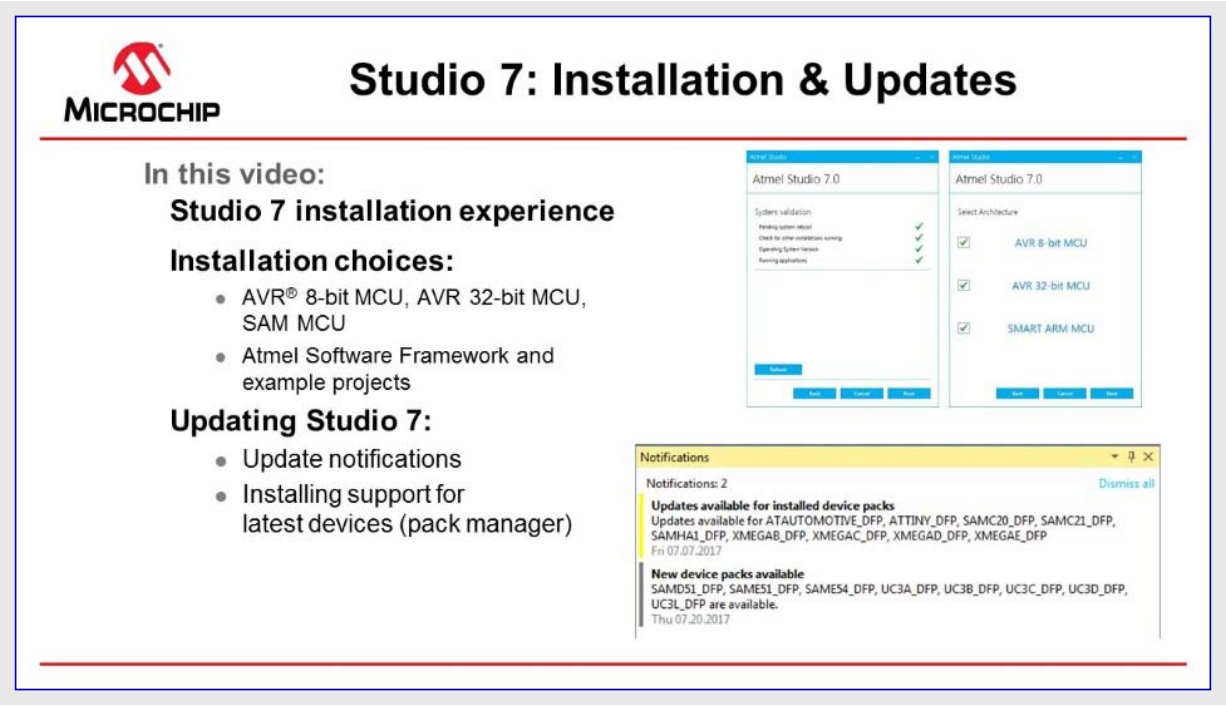
```
{
    ADC0.CTRLC  = ADC_PRESC_DIV8_gc | ADC_REFSEL_VDDREF_gc;
    ADC0.CTRLA  = ADC_ENABLE_bm | ADC_RESSEL_8BIT_gc;
    ADC0.MUXPOS = ADC_MUXPOS_AIN6_gc;

    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のﾎﾞｰﾚｰﾄを比較
// 1ﾊﾞ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ﾋﾟﾝ禁止
}

// 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.PIN0CTRL)[pin] = PORT_ISC_INPUT_DISABLE_gc;
        (&PORTB.PIN0CTRL)[pin] = PORT_ISC_INPUT_DISABLE_gc;
        (&PORTC.PIN0CTRL)[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、ｱｲﾄﾞﾙ、ｽﾀﾝﾊﾞｲと比較
    set_sleep_mode(SLEEP_MODE_STANDBY);

    while (1)
    {
        sleep_mode();
    }
    }

ISR(ADC0_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());
}
```

1.4. インストールと更新

本項はAtmel Studio 7ｲﾝｽﾄｰﾙ、Atmel Studioやﾌﾟﾗｸﾞｲﾝに対する更新ｲﾝｽﾄｰﾙだけでなく、新ﾃﾞﾊﾞ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

1.4.1. インストール

支援ｵﾍﾟﾚｰﾃｨﾝｸﾞ 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ﾋﾞｯﾄ(x86)
- 64ﾋﾞｯﾄ(x64)

ﾊｰﾄﾞｳｪｱ要件

- 1.6GHzまたはより速いﾌﾟﾛｾｯｻを持つｺﾝﾋﾟｭｰﾀ
- RAM
  - x86に対して1Gﾊﾞｲﾄ
  - x64に対して2Gﾊﾞｲﾄ
  - 仮想ﾏｼﾝで走行する場合、追加の512Mﾊﾞｲﾄ RAM
- 6Gﾊﾞｲﾄの利用可能なﾊｰﾄﾞ ﾃﾞｨｽｸ空間

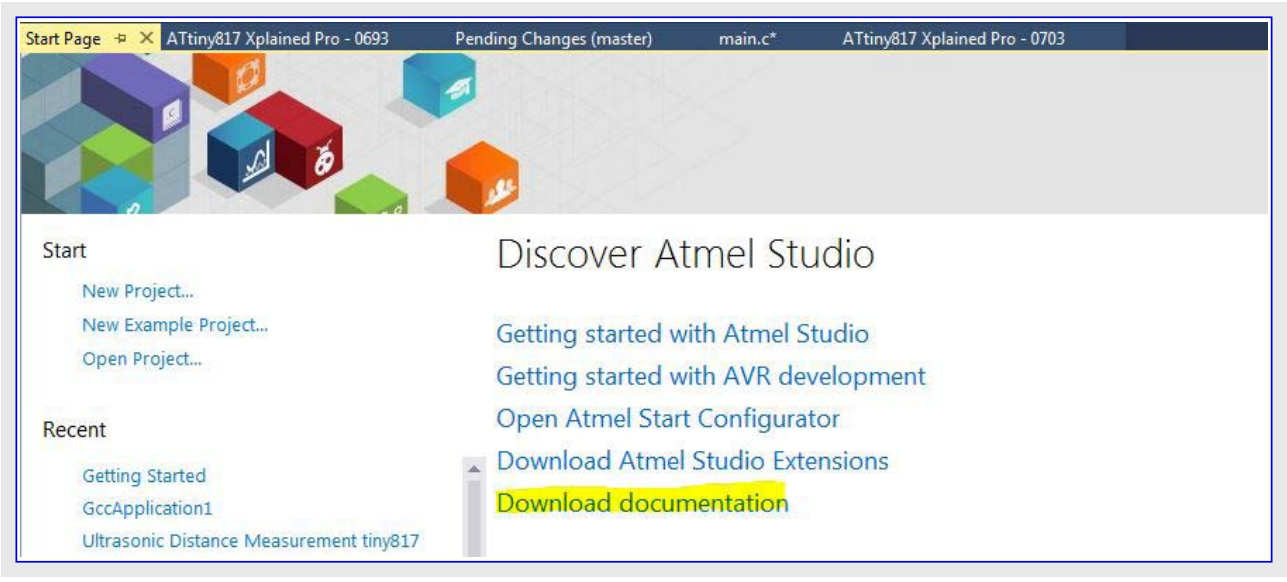
ﾀﾞｳﾝﾛｰﾄﾞとｲﾝｽﾄｰﾙ

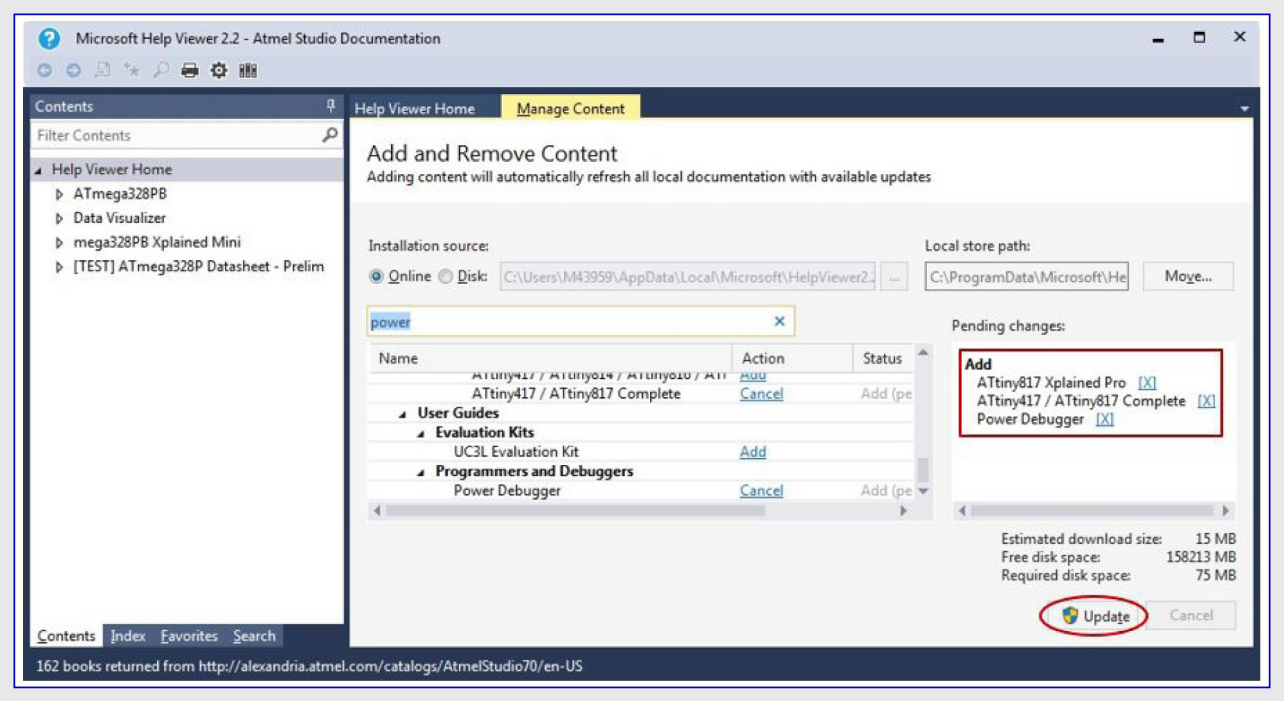
- 最新のAtmel Studioｲﾝｽﾄｰﾗｰをﾀﾞｳﾝﾛｰﾄﾞしてください。: Atmel Studion 7
  - ｳｪﾌﾞ ｲﾝｽﾄｰﾗは(<10Mﾊﾞｲﾄの)小さなﾌｧｲﾙで、必要とされる時に指定される部分をﾀﾞｳﾝﾛｰﾄﾞします。
  - ｵﾌﾗｲﾝ ｲﾝｽﾄｰﾗは組み込まれた全ての部分を持ちます。
- Atmel StudioはAtmel StudioとAVR Studioの旧版と並走することができます。以前の版のｱﾝｲﾝｽﾄｰﾙは不要です。
- 「ｼｽﾃﾑ要件」項でﾊｰﾄﾞｳｪｱとｿﾌﾄｳｪｱの必要条件を確認してください。
- ﾛｰｶﾙ管理者権限を持つことを確実にしてください。
- 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Atmel USB/ｼﾘｱﾙ ﾊｰﾄﾞ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)をﾀﾞﾌﾞﾙ ｸﾘｯｸしてｲﾝｽﾄｰﾙ ｳｨｻﾞｰﾄﾞに従ってください。
- 一旦終了すると、ｲﾝｽﾄｰﾗはStart Atmel Studio after completion(完了後にAtmel 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Atmel 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に於いてﾀｲﾄﾙ ﾊﾞｰの迅速起動領域傍で更新通知(旗のｼﾝﾎﾞﾙ)を見るかもしれません。ここでは更新された構成部分やﾃﾞﾊﾞｲｽの支援を選んでｲﾝｽﾄｰﾙすることができます。

1.4.2. オフライン資料のﾀﾞｳﾝﾛｰﾄﾞ

ｵﾌﾗｲﾝで作業したい場合、Atmel Studio 7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Atmel Studio 7のStart Page(開始頁)からDownload documentation(資料ﾀﾞｳﾝﾛｰﾄﾞ)をｸﾘｯｸしてください。手助け表示部がﾎﾟｯﾌﾟｱｯﾌﾟすると、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って)data sheets(ﾃﾞｰﾀｼｰﾄ)、user manuals(使用者手引書)、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ﾃﾞﾊﾞｯｶﾞ使用者手引書)、ATtiny817 Xplained Pro user manual(ATtiny817 Xplained Pro使用者手引書)だけでなく、ATtiny817 Complete data sheet(ATtiny817完全ﾃﾞｰﾀｼｰﾄ)もﾀﾞｳﾝﾛｰﾄﾞするために選んでいます。その後のUpdate(更新)のｸﾘｯｸがﾀﾞｳﾝﾛｰﾄﾞを開始します。

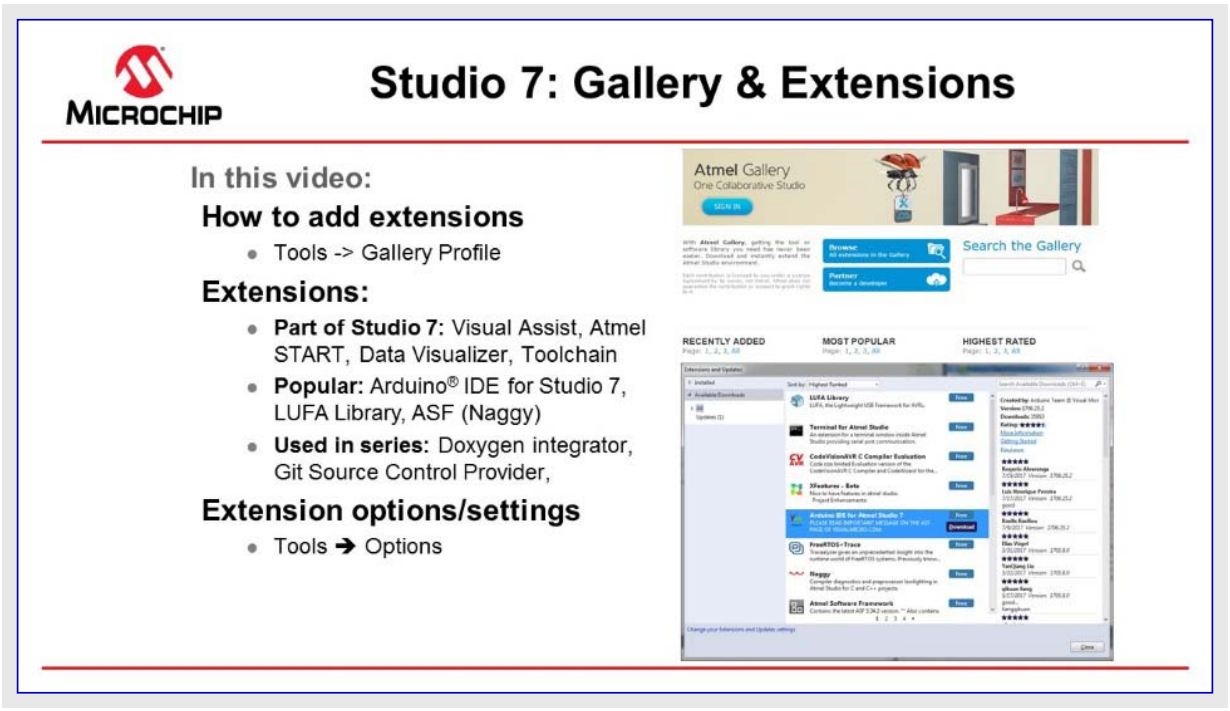




1.5. Microchip陳列室とStudio拡張

本項はMicrochip陳列室(gallery)を通してAtmel 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



映像: 陳列室、Atmel Studio拡張、更新

拡張追加

含まれた拡張

人気の拡張

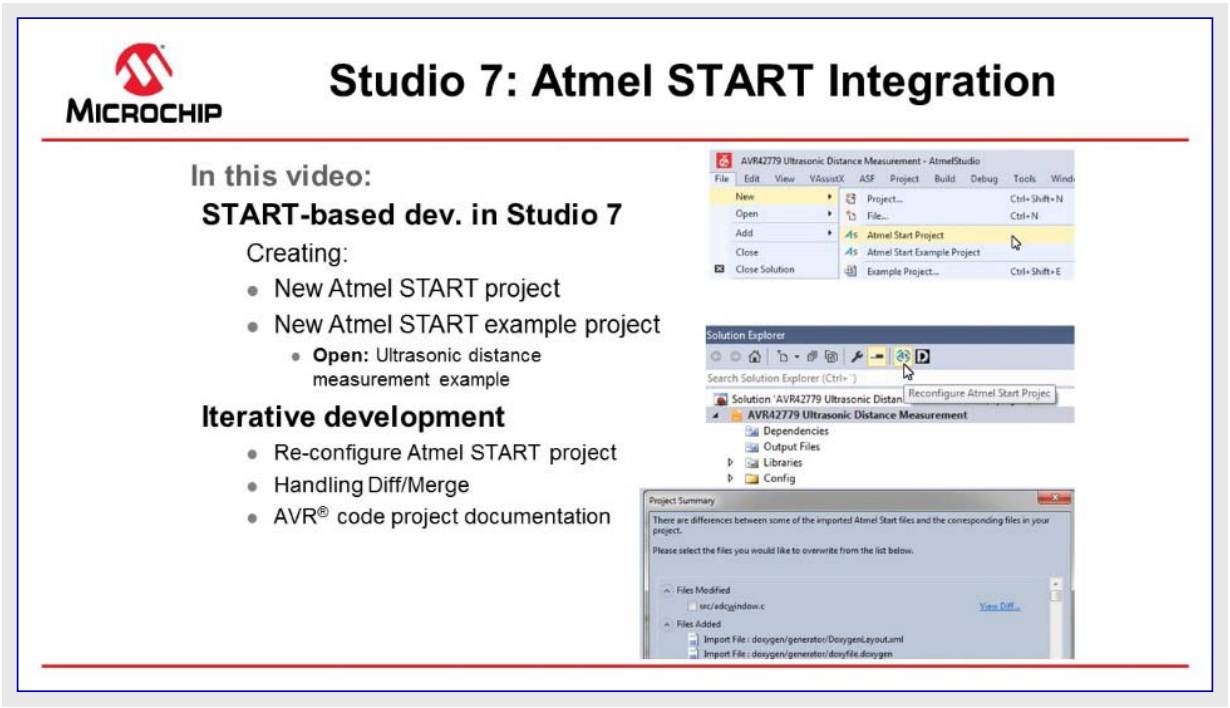
一連で使われる拡張

拡張任意設定/設定

1.6. Atmel START統合

Atmel STARTとAtmel Studio 7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、Atmel Studio 7でSTARTに基づくﾌﾟﾛｼﾞ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

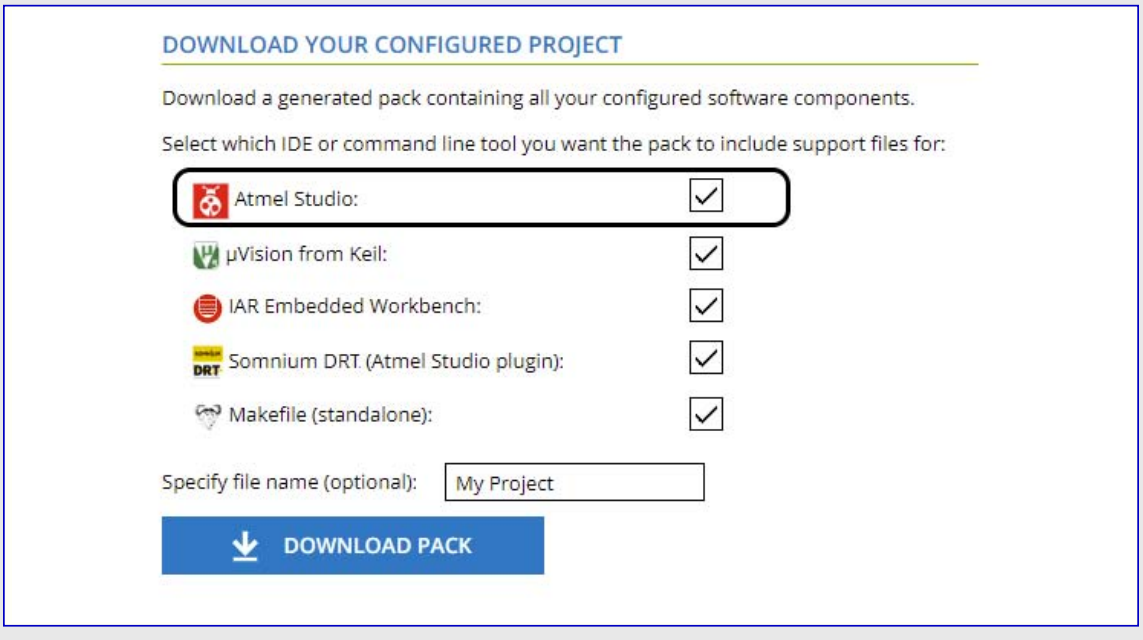


映像: Atmel START統合

すべきこと: Atmel STARTからﾌﾟﾛｼﾞｪｸﾄをｴｸｽﾎﾟｰﾄします。

1. Atmel STARTｳｪﾌﾞｻｲﾄで新しいﾌﾟﾛｼﾞｪｸﾄ(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ﾎﾟｰﾄ)釦をｸﾘｯｸしてください。Atmel 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(一括ﾀﾞｳﾝﾛｰﾄﾞ)上でｸﾘｯｸしてください。atmelstart.atzip一括ﾌｧｲﾙがﾀﾞｳﾝﾛｰﾄﾞされます。

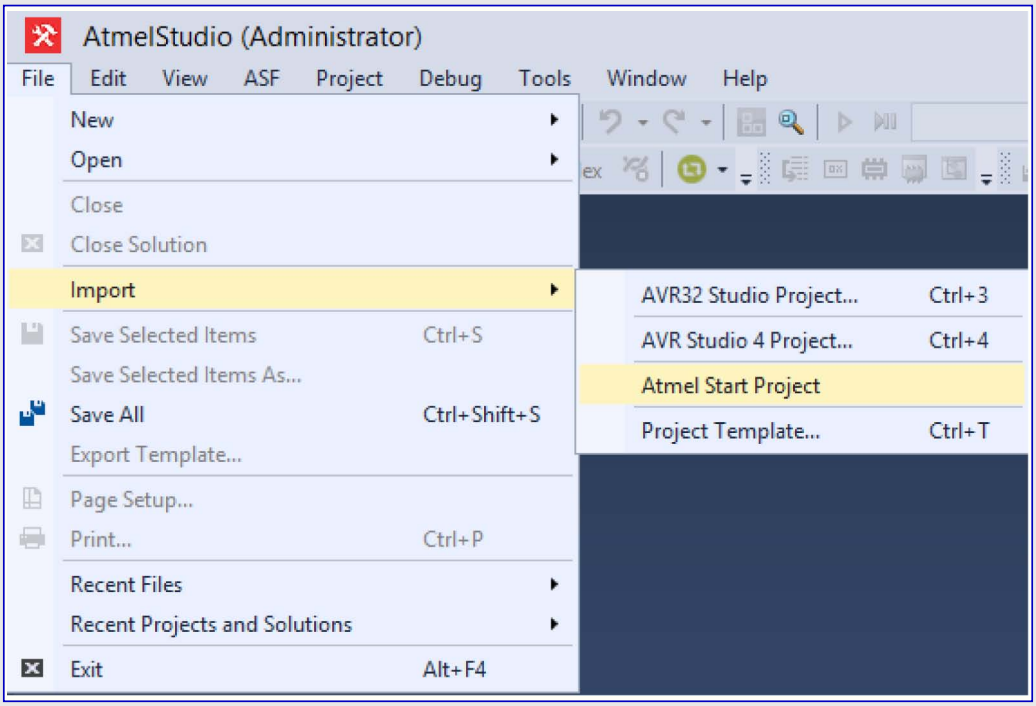
図1-1. 構成設定したﾌﾟﾛｼﾞｪｸﾄのﾀﾞｳﾝﾛｰﾄﾞ



すべきこと: Atmel STARTの出力をAtmel Studioにｲﾝﾎﾟｰﾄします。

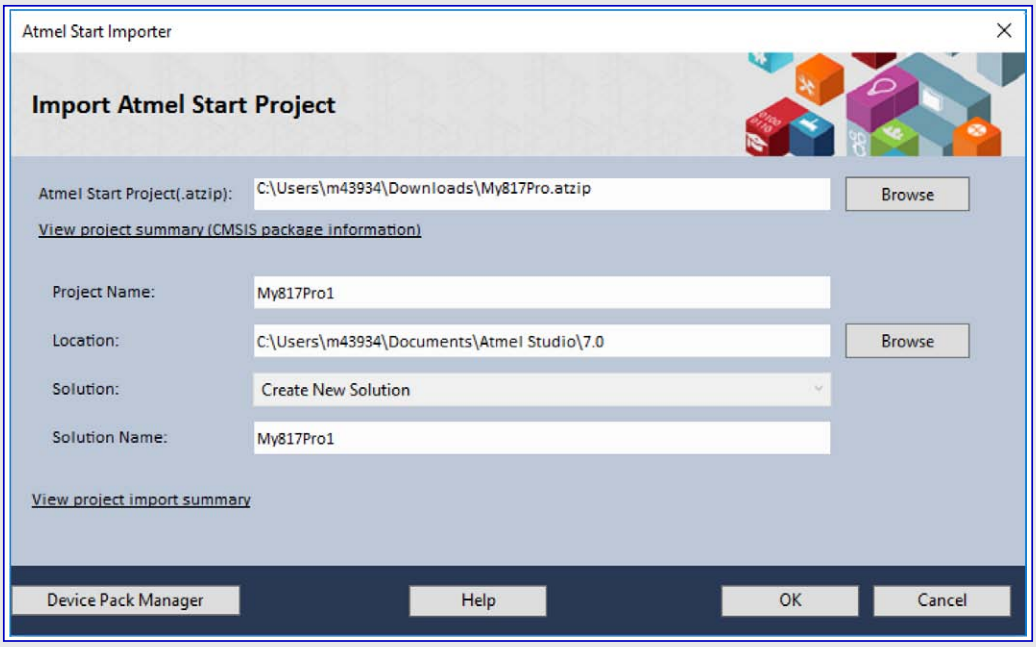
4. Atmel Studioを起動してください。
5. File(ﾌｧｲﾙ)⇒Import(ｲﾝﾎﾟｰﾄ)⇒Atmel START Project(Atmel STARTﾌﾟﾛｼﾞｪｸﾄ)を選んでください。

図1-2. Atmel STARTﾌﾟﾛｼﾞｪｸﾄのｲﾝﾎﾟｰﾄ

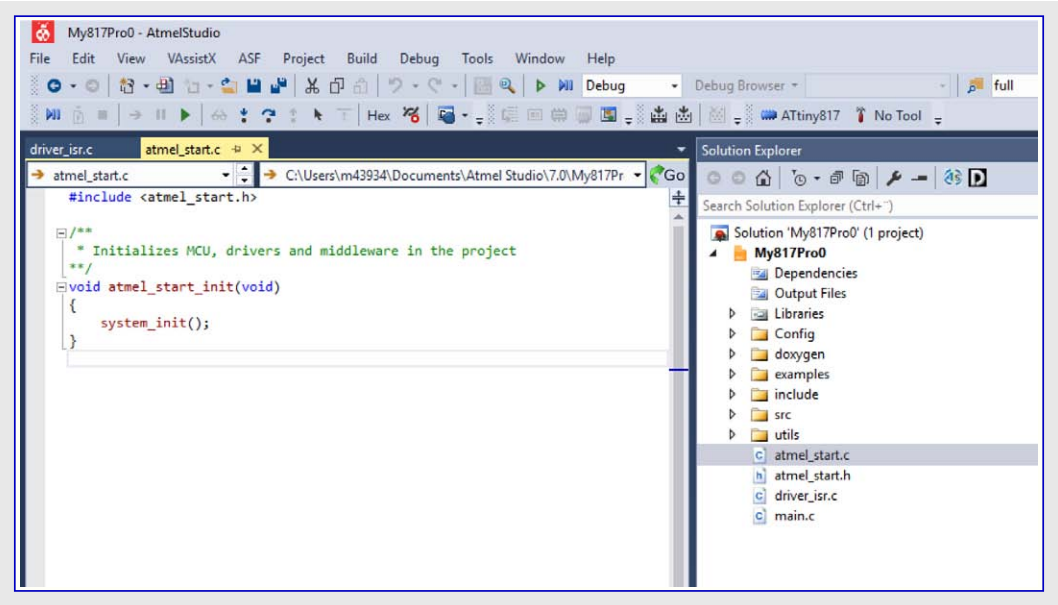


6. ﾀﾞｳﾝﾛｰﾄﾞしたatmelstart.atzipﾌｧｲﾙを探して選んでください。
7. Atmel STARTｲﾝﾎﾟｰﾄ ﾀﾞｲｱﾛｸﾞ ﾎﾞｯｸｽが開きます。Project Name(ﾌﾟﾛｼﾞｪｸﾄ名)、Location(場所)、Solution Name(解決策名)としてﾌﾟﾛｼﾞｪｸﾄの詳細を入力してください。

図1-3. Atmel STARTﾌﾟﾛｼﾞｪｸﾄ ｲﾝﾎﾟｰﾄ部



8. 新しいAtmel Studioﾌﾟﾛｼﾞｪｸﾄが作成され、ﾌｧｲﾙがｲﾝﾎﾟｰﾄされます。



すべきこと: Atmel STARTの出力をAtmel Studioにｲﾝﾎﾟｰﾄします。

9. いくつかのﾌﾟﾛｼﾞｪｸﾄはDoxygen用に形式化された資料を含みます。

   注: Doxygenはhttp://doxygen.orgからﾀﾞｳﾝﾛｰﾄﾞ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Atmel Studioを構成設定することを尋ねられるでしょう。

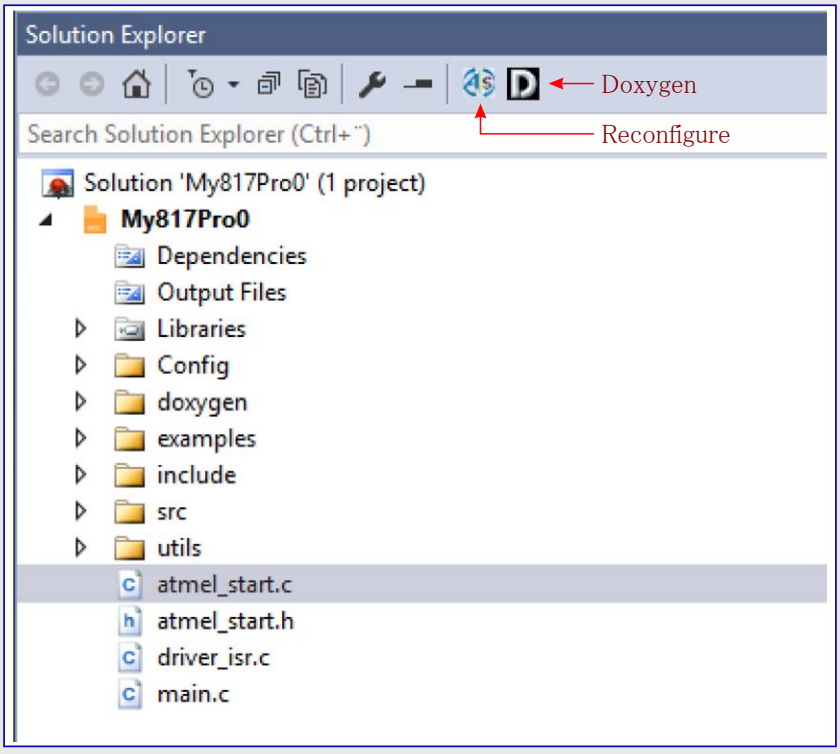
10. 資料を生成するにはDoxyegn釦をｸﾘｯｸしてください。Doxygenが走行して生成された資料が新しいｳｨﾝﾄﾞｳで開きます。

すべきこと: Atmel STARTを用いてﾌﾟﾛｼﾞｪｸﾄを再構成設定します。

11. Reconfigure(再構成設定)釦をｸﾘｯｸするか、またはSokution Explorer内のﾌﾟｼﾞ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ﾌﾟﾛｼﾞｪｸﾄを再構成設定)を選んでください。

12. Atmel Studio内のｳｨﾝﾄﾞｳでAtmel STARTが開きます。

図1-4. Atmel STARTﾌﾟﾛｼﾞｪｸﾄ再構成設定とSoxygen釦

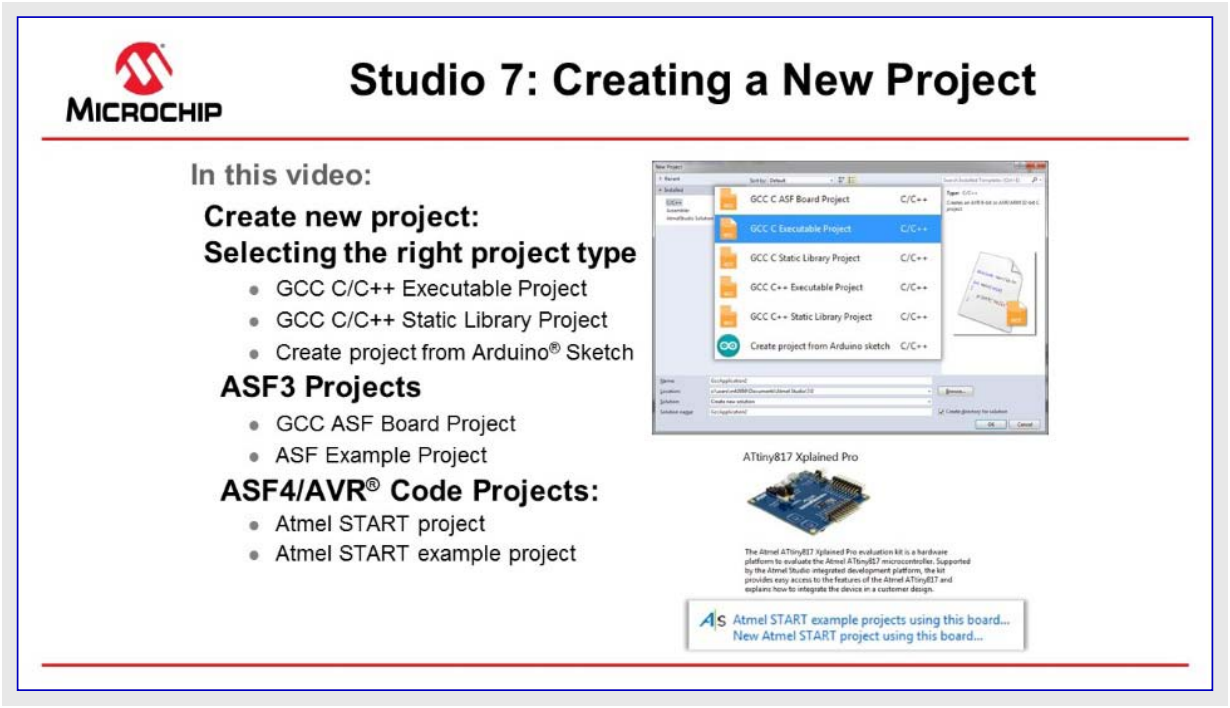


13. ﾌﾟﾛｼﾞｪｸﾄに対して必要な変更を行ってください。Atmel STARTｳｨﾝﾄﾞｳの下部でGENERATE PROJECT(ﾌﾟﾛｼﾞｪｸﾄ生成)釦をｸﾘｯｸしてください。

1.7. 新規ﾌﾟﾛｼﾞｪｸﾄ作成

本項は新しいAtmel Studioﾌﾟﾛｼﾞ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

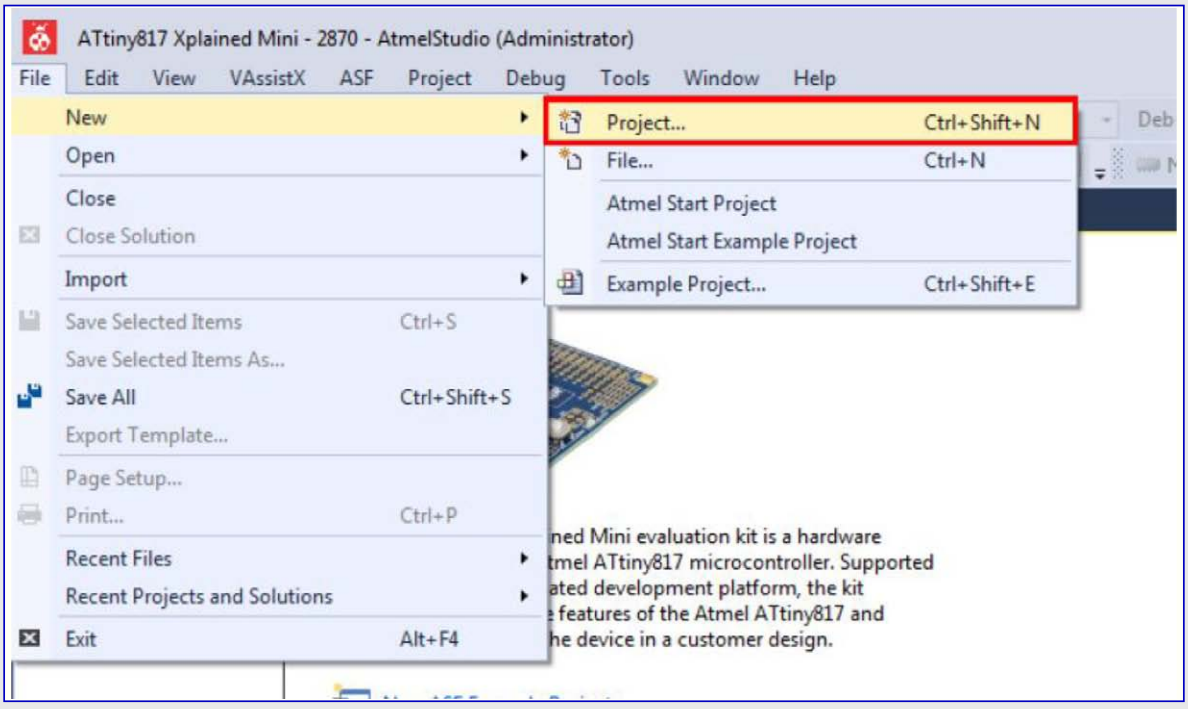


映像: 新規ﾌﾟﾛｼﾞｪｸﾄ作成

すべきこと: AVR ATtiny817ﾃﾞﾊﾞｲｽ用の新しい空GCC C実行可能ﾌﾟﾛｼﾞｪｸﾄを作成します。

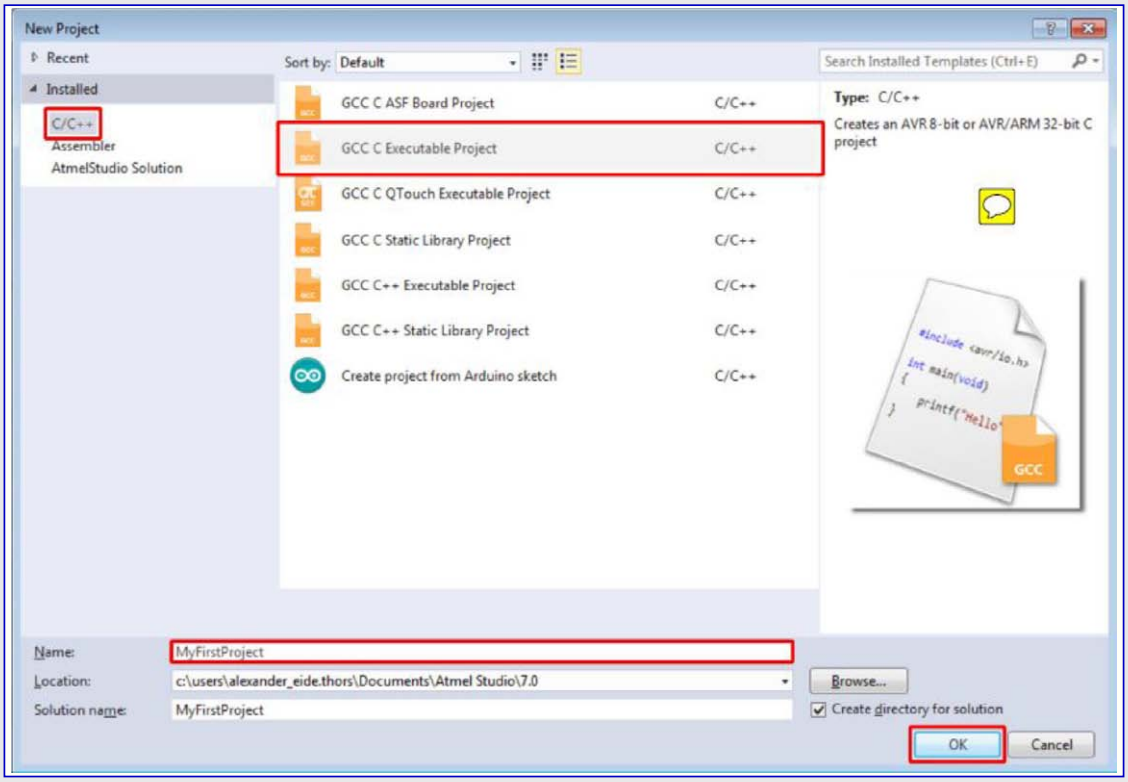
1. Atmel Studioを開いてください。
2. Atmel Studioに於いて、図1-5.で描かれたようにFile(ﾌｧｲﾙ)⇒New(新規)⇒Project(ﾌﾟﾛｼﾞｪｸﾄ)に行ってください。

図1-5. Atmel Studioでの新規ﾌﾟﾛｼﾞｪｸﾄ作成



3. ﾌﾟﾛｼﾞｪｸﾄ生成ｳｨｻﾞｰﾄﾞが現れます。このﾀﾞｲｱﾛｸﾞは使われるﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語とﾌﾟﾛｼﾞｪｸﾄ雛形を指定するための任意選択を提供します。このﾌﾟﾛｼﾞｪｸﾄは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。骨子の実行可能ﾌﾟﾛｼﾞ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にﾌﾟﾛｼﾞｪｸﾄ名を与えてOKをｸﾘｯｸしてください。**図1-6.**をご覧ください。

図1-6. 新規ﾌﾟﾛｼﾞｪｸﾄのﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語と雛形の選択



助言: 全てのAtmel Studioﾌﾟﾛｼﾞｪｸﾄは解決策(solution)に属し、既定によってAtmel Studioは新しく作成された解決策とﾌﾟﾛｼﾞ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(解決策用にﾃﾞｨﾚｸﾄﾘを作成)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、Atmel 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ﾀﾞを作成します。

ﾌﾟﾛｼﾞｪｸﾄ形式について

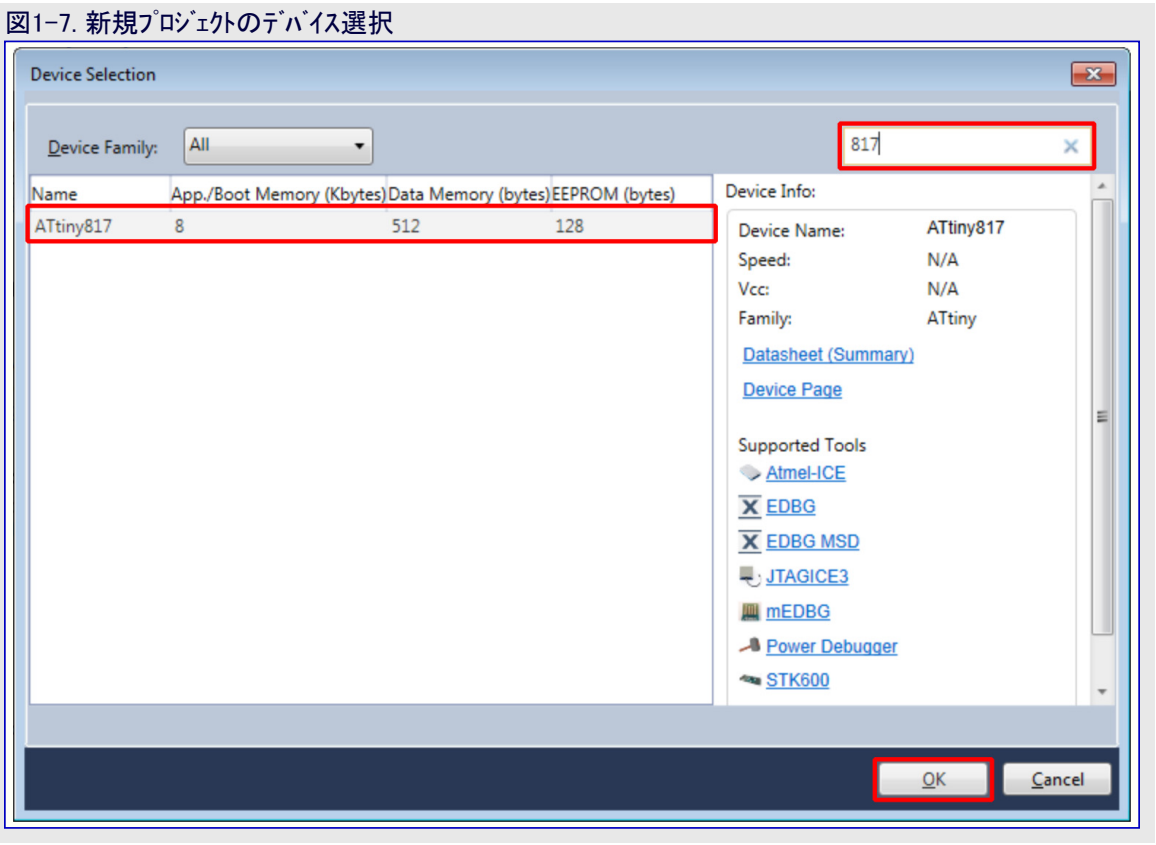
表1-1. ﾌﾟﾛｼﾞｪｸﾄ形式

区分	ﾌﾟﾛｼﾞｪｸﾄ雛形	説明
C/C++	GCC C ASF基板ﾌﾟﾛｼﾞｪｸﾄ	AVR 8ﾋﾞｯﾄかAVR/ARM 32ﾋﾞｯﾄのASF3基板ﾌﾟﾛｼﾞ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実行可能ﾌﾟﾛｼﾞｪｸﾄ	AVR 8ﾋﾞｯﾄかAVR/ARM 32ﾋﾞｯﾄのGCCﾌﾟﾛｼﾞ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 静的ﾗｲﾌﾞﾗﾘ ﾌﾟﾛｼﾞｪｸﾄ	AVR 8ﾋﾞｯﾄかAVR/ARM 32ﾋﾞｯﾄのGCC静的ﾗｲﾌﾞﾗﾘ(LIB)ﾌﾟﾛｼﾞｪｸﾄを作成するにはこの雛形を選んでください。この事前ｺﾝﾊﾟｲﾙしたﾗｲﾌﾞﾗﾘ(.a)は他のﾌﾟﾛｼﾞｪｸﾄ(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(ｺｰﾄﾞ再利用)に使えます。
C/C++	GCC C++実行可能ﾌﾟﾛｼﾞｪｸﾄ	AVR 8ﾋﾞｯﾄかAVR/ARM 32ﾋﾞｯﾄのC++ﾌﾟﾛｼﾞ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++ 静的ﾗｲﾌﾞﾗﾘ ﾌﾟﾛｼﾞｪｸﾄ	AVR 8ﾋﾞｯﾄかAVR/ARM 32ﾋﾞｯﾄのC++静的ﾗｲﾌﾞﾗﾘ(LIB)ﾌﾟﾛｼﾞｪｸﾄを作成するにはこの雛形を選んでください。この事前ｺﾝﾊﾟｲﾙしたﾗｲﾌﾞﾗﾘ(.a)は他のﾌﾟﾛｼﾞｪｸﾄ(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(ｺｰﾄﾞ再利用)に使えます。
ｱｾﾝﾌﾞﾗ	ｱｾﾝﾌﾞﾗ ﾌﾟﾛｼﾞｪｸﾄ	AVR 8ﾋﾞｯﾄ ｱｾﾝﾌﾞﾗ ﾌﾟﾛｼﾞｪｸﾄを作成するにはこの雛形を選んでください。

注意: この表は既定ﾌﾟﾛｼﾞｪｸﾄ形式だけを一覧にします。拡張によって他のﾌﾟﾛｼﾞｪｸﾄ形式が追加されるかもしれません。

4. 次に、ﾌﾟﾛｼﾞｪｸﾄがどのﾃﾞﾊﾞ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。ﾃﾞﾊﾞｲｽの一覧はDevice Selection(ﾃﾞﾊﾞｲｽ選択)ﾀﾞｲｱﾛｸﾞで提示され、これは図1-7.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(ﾃﾞﾊﾞｲｽ系統)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。このﾌﾟﾛｼﾞｪｸﾄはATtiny817 AVRﾃﾞﾊﾞ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。ﾃﾞﾊﾞｲｽ一覧でATtiny817項目を選んでOKをｸﾘｯｸすることによってﾃﾞﾊﾞｲｽ選択を承認してください。

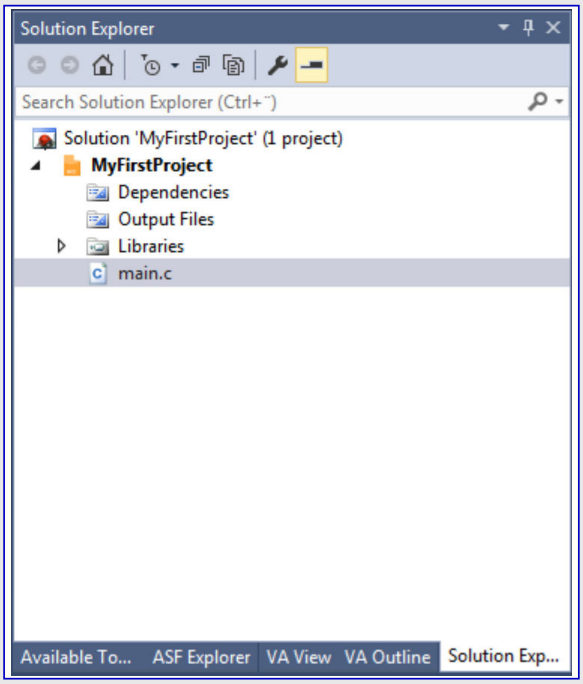
図1-7. 新規ﾌﾟﾛｼﾞｪｸﾄのﾃﾞﾊﾞｲｽ選択



助言: "tiny"に対する検索は支援される全てのATtinyﾃﾞﾊﾞ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ﾃﾞﾊﾞｲｽの一覧が提供されます。Tools(ﾂｰﾙ)⇒Device Pack Manager(ﾃﾞﾊﾞｲｽ一括管理部)は追加ﾃﾞﾊﾞ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実行可能ﾌﾟﾛｼﾞｪｸﾄは今やAVR ATtiny817ﾃﾞﾊﾞｲｽ用に作成されました。Solution Explorer(解決策ｴｸｽﾌﾟﾛｰﾗ)は図1-8.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(解決策ｴｸｽﾌﾟﾛｰﾗ)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。

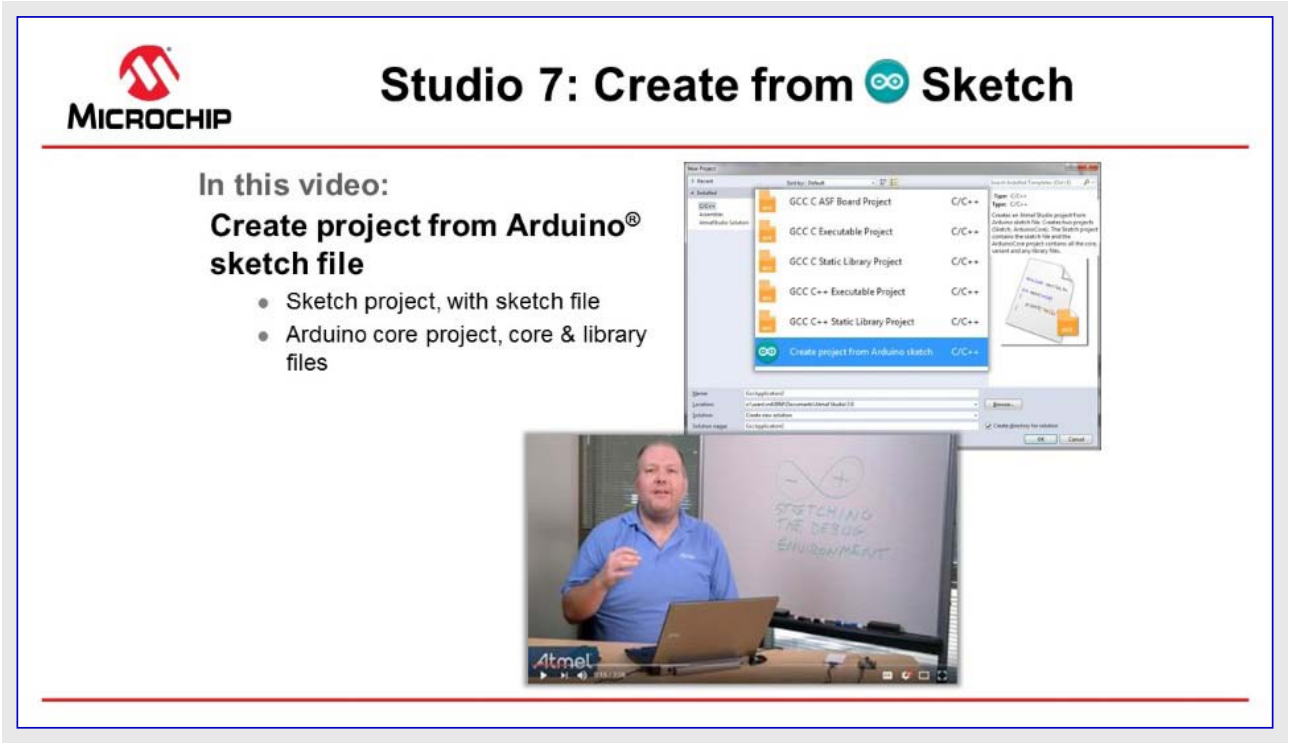
図1-8. 解決策ｴｸｽﾌﾟﾛｰﾗ



1.8. Arduinoスケッチから作成

本項はArduinoスケッチから新しいAtmel 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



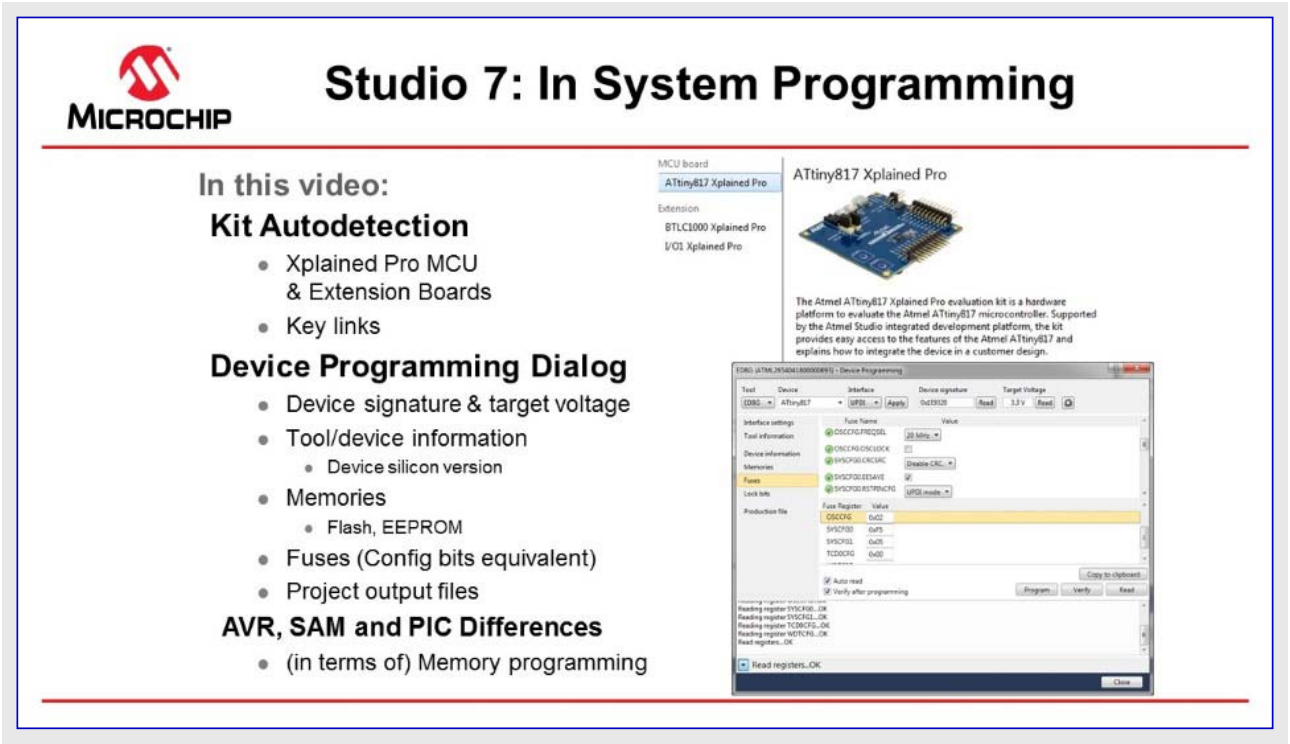
映像: Arduinoスケッチから作成

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。

1.9. 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 ボックスの概要を与えます。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



映像: ｷｯﾄ接続と実装書き込み

すべきこと: ﾌﾟﾛｼﾞｪｸﾄを持つ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ｹｰﾌﾞﾙを使ってATtiny817 Xplained Pro基板をｺﾝﾋﾟｭｰﾀに接続してください。下図のようにAtmel Studioでｷｯﾄ頁が提示されるべきです。

図1-9. ATtiny817 Xplained Pro開始頁

1.1. 基板用の資料とﾃﾞﾊﾞｲｽ用ﾃﾞｰﾀｼｰﾄへのﾘﾝｸがあります。

1.2. 基板用のAtmel STARTﾌﾟﾛｼﾞｪｸﾄを作成することが可能です。Atmel STARTﾘﾝｸのﾌﾟﾛｼﾞｪｸﾄ 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(ﾃﾞﾊﾞｲｽ ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ)によってProgramming(ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ)ﾀﾞｲｱﾛｸﾞを開きます。

2.1. EDBGﾂｰﾙを選んでDevice=ATtiny817を確実にし、その後にﾃﾞﾊﾞｲｽ識票と目的対象電圧を読んでも構いません。

2.2. 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。

2.3. Tool infomation(ﾂｰﾙ情報)：EDBGﾂｰﾙについての情報を表示

2.4. Device information(ﾃﾞﾊﾞｲｽ情報)：ﾃﾞﾊﾞｲｽについての情報を表示。ﾃﾞﾊﾞ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。

2.5. 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。

2.6. Fuses(ﾋｭｰｽﾞ)：ﾋｭｰｽﾞ、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

2.7. Lock bits(施錠ﾋﾞｯﾄ)：ﾒﾓﾘ施錠

2.8. 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ﾃﾞﾊﾞｲｽを書き込み

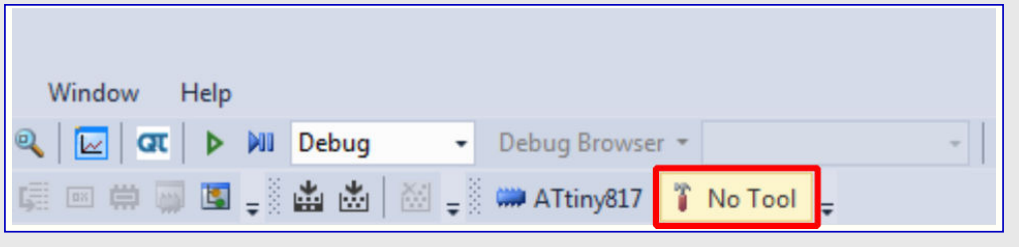
2.9. AVRはHEXﾌｧｲﾙでﾌﾗｯｼｭ ﾒﾓﾘ、EEPﾌｧｲﾙでEEPROMを持ち、一方でPICはHEXﾌｧｲﾙで全てとﾋｭｰｽﾞさえも持ちます。

2.10. 例えば、SAML21JﾃﾞﾊﾞｲｽはEEPROMを持ちません(ﾌﾗｯｼｭ ﾒﾓﾘで模倣できます)。ﾃﾞﾊﾞｲｽを施錠する保護ﾋﾞｯﾄ任意選択も持ちます。

3. File(ﾌｧｲﾙ)⇒New project(新規ﾌﾟﾛｼﾞｪｸﾄ)を選ぶことによるCreate a new project(新規ﾌﾟﾛｼﾞｪｸﾄ作成)で実体に対してC実行可能ﾌﾟﾛｼﾞｪｸﾄを選び、ﾃﾞﾊﾞｲｽ名で選別することによってﾃﾞﾊﾞｲｽを選んでください。違うﾌﾟﾛｼﾞｪｸﾄ形式は別の開始に際しての映像で検討されます。

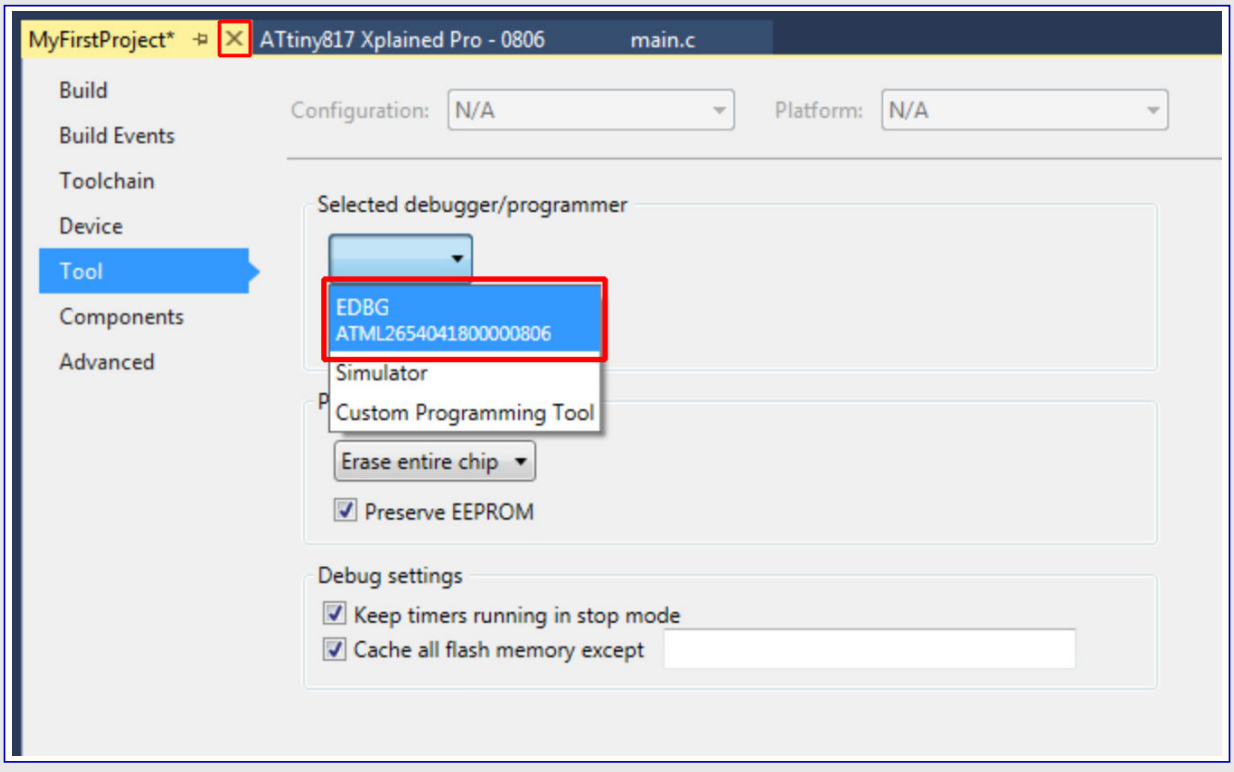
4. ﾌﾟﾛｼﾞｪｸﾄが選ばれたなら、下図で示されるように、ﾂｰﾙ ﾀﾞｲｱﾛｸﾞを開くために上部ﾒﾆｭｰ ﾊﾞｰに配置されたTool(ﾂｰﾙ)釦をｸﾘｯｸしてください。

図1-10. ツール釦



5. Project Properties(ﾌﾟﾛｼﾞｪｸﾄ ﾌﾟﾛﾊﾟﾃｨ)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ﾌﾟﾛｸﾞﾗﾑ/ﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽ(UPDI:Unified Programming Debugging Interface)に初期化されべきです。

図1-11. ﾌﾟﾛｼﾞｪｸﾄ ﾌﾟﾛﾊﾟﾃｨでの書き込み器/ﾃﾞﾊﾞｯｶﾞの選択



助言: ﾂｰﾙの通番は引き落としﾒﾆｭｰでのそれ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/ﾃﾞﾊﾞｯｸﾞ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Atmel Studioによって使われるツールが今や指定されました。

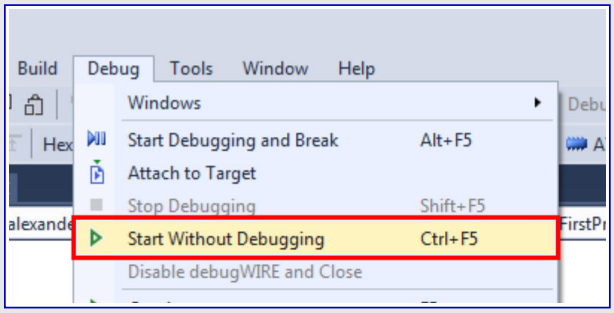
1.9.1. 設定確認

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)釦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1-12. デバッグなしで開始

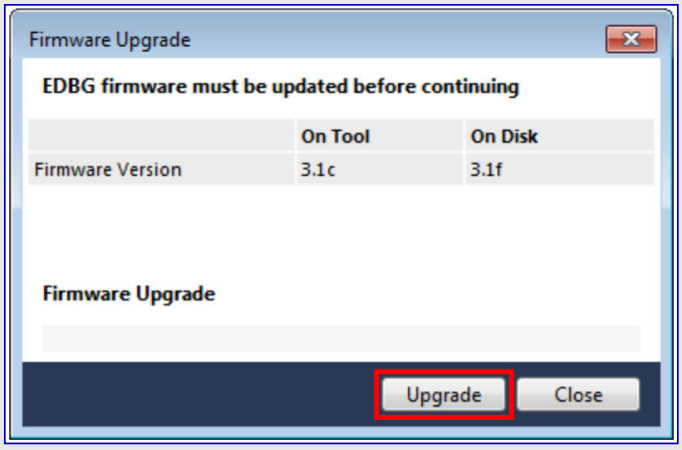


2. Atmel Studio 7がプロジェクトを構築する(Start Without Debugging(デバッグなしで開始)を押す時に自動的に行われる)と、Solution Explorer(解決策エクスプローラ)でいくつかの生成された出力ファイルが現れます。以下の出力ファイルが生成されます。
   - 2.1. EEPファイル　：デバイスに書かれるEEPROM内容
   - 2.2. ELFファイル　：プログラム、EEPROM、ヒューズを含み、デバイスに書かれる全てを含みます。
   - 2.3. HEXファイル　：デバイスに書かれるフラッシュ メモリ内容
   - 2.4. LSSファイル　：逆アセンブルしたELFファイル
   - 2.5. 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
   - 2.6. 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1-13.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)釦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)釦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。

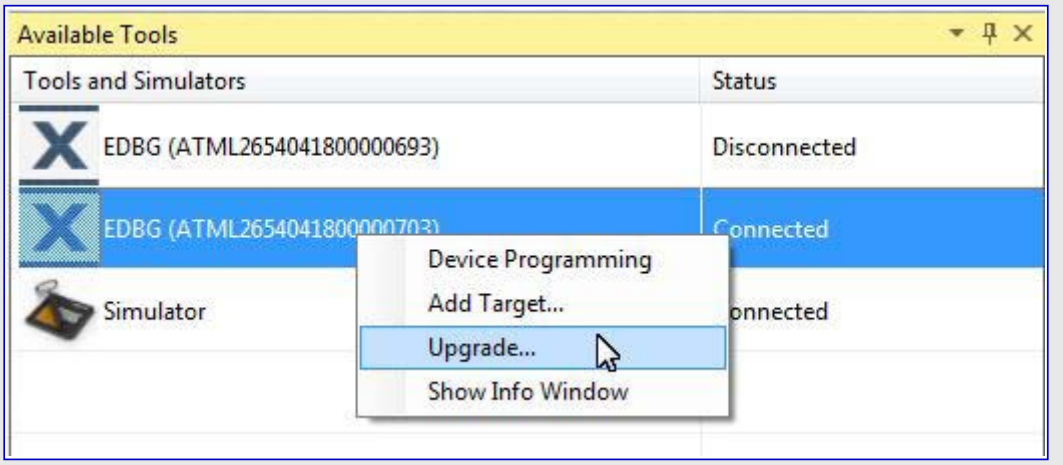
図1-13. ファームウェア更新ダイアログ



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Atmel Studio 7に特別に尋ねることができます。

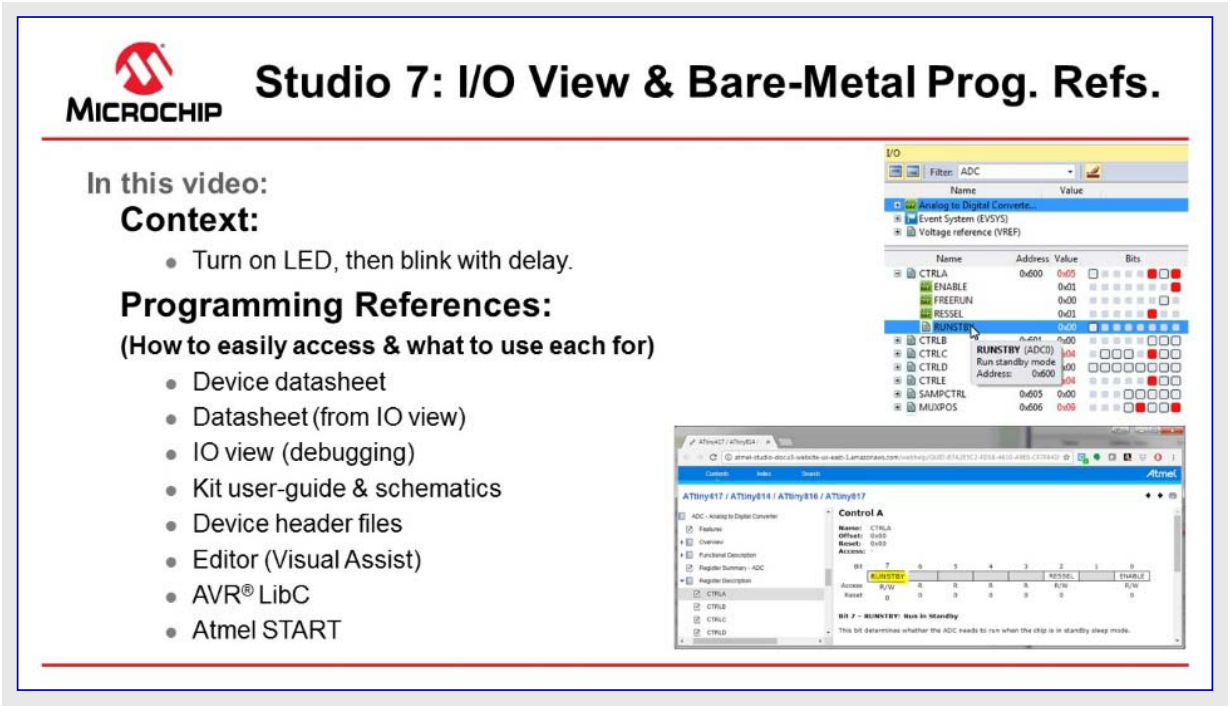
図1-14. (Viewﾒﾆｭｰでの)Atmel Studio 7で利用可能なﾂｰﾙ



1.10. I/O表示部と他の空からのﾌﾟﾛｸﾞﾗﾐﾝｸﾞ参考資料

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Atmel Studio 7で一般的にｺｰﾄﾞ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ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ参考資料で、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。ﾌﾟﾛｼﾞｪｸﾄの脈絡は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Atmel Studio 7で支援される殆どのﾃﾞﾊﾞｲｽに適用するにも関わらず、この原理はAtmel Studio 7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



映像: I/O表示部と空からのﾌﾟﾛｸﾞﾗﾐﾝｸﾞ参考資料

下の一覧は一般的に使われるﾌﾟﾛｸﾞﾗﾐﾝｸﾞ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、これは編集やﾃﾞﾊﾞｯｸﾞの時にﾃﾞｰﾀｼｰﾄのﾚｼﾞｽﾀ説明を誘導することだけでなく、ﾃﾞﾊﾞｯｸﾞ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。ﾃﾞﾊﾞｯｸﾞ時のI/O表示部のこの2つ目の使用は新しいﾚｼﾞｽﾀ構成設定の試験にも使われます。

この話題は「ﾃﾞﾊﾞｯｸﾞ3: I/O表示部、ﾒﾓﾘ表示部と監視」だけでなく「ｴﾃﾞｨﾀ: ｺｰﾄﾞの記述と整理 (Visual Assist)」の両方に対して密接に関連します。

- ﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄ
- (I/O表示部からの)ﾃﾞｰﾀｼｰﾄ
- ｷｯﾄの使用者の手引きと回路図
- I/O表示部(ﾃﾞﾊﾞｯｸﾞ)
- ｴﾃﾞｨﾀ (Visual Assist)

・ﾃﾞﾊﾞｲｽのﾍｯﾀﾞ ﾌｧｲﾙ
・AVR Libc (AVR特有)
・Atmel START: ATtiny817ﾌﾟﾛｼﾞｪｸﾄ

その過程で以下のｺｰﾄﾞが書かれます。このｺｰﾄﾞが簡単とは言え、前のﾌﾟﾛｸﾞﾗﾐﾝｸﾞ参考資料の一覧を使い、決定過程が記述されます。

```
#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。このﾍｯﾀﾞ ﾌｧｲﾙは選んだﾃﾞﾊﾞｲｽ用の正しいﾚｼﾞｽﾀ割り当てをｲﾝｸﾙｰﾄﾞし、この行なしではｺﾝﾊﾟｲﾗが上のｺｰﾄﾞで参照されたどのﾏｸﾛも認知しません。

ﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄ (PDF)

I/O表示部がﾚｼﾞｽﾀ ﾚﾍﾞﾙでﾃﾞ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。ﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄはPDF形式で、少なくて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ﾃﾞｰﾀｼｰﾄの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1-15. PDFﾃﾞｰﾀｼｰﾄからのPORT構成図

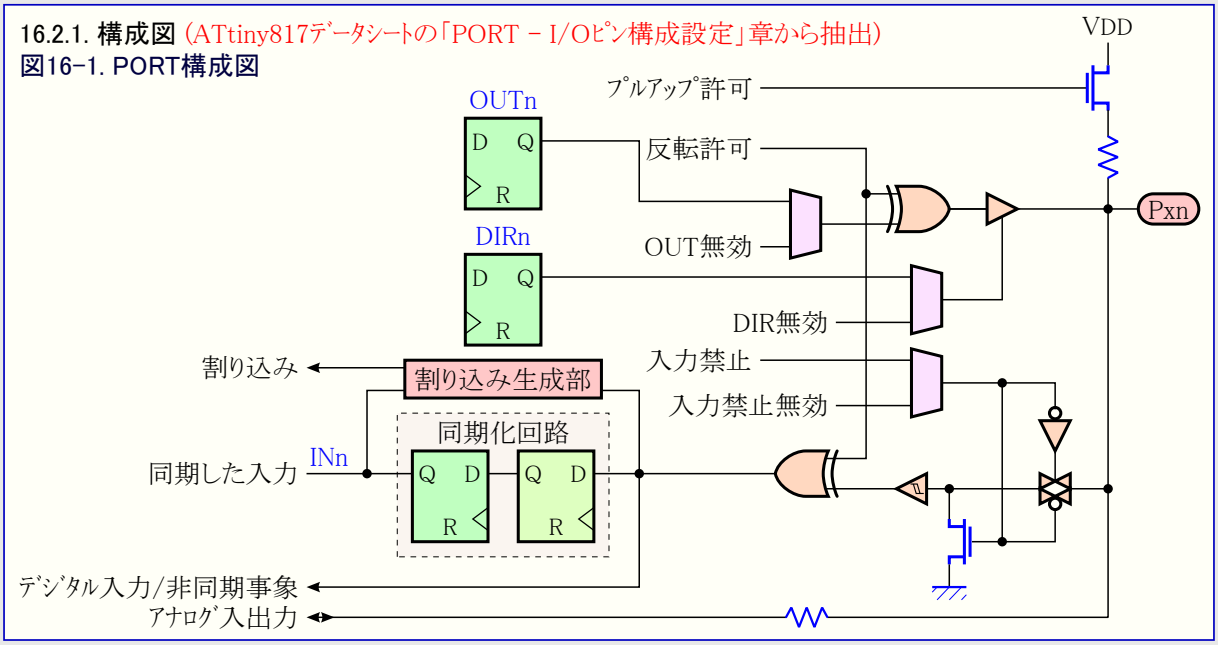


図1-16. ATtiny817のPDFデータシートからの基本機能

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(Pxn)ピンはPORTx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レジスター式を持ち、(n)ピン用のレジスター式の基準アドレスはバイト アドレスでPORT+$10です。そのレジスター式内の指標は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'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n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いました。

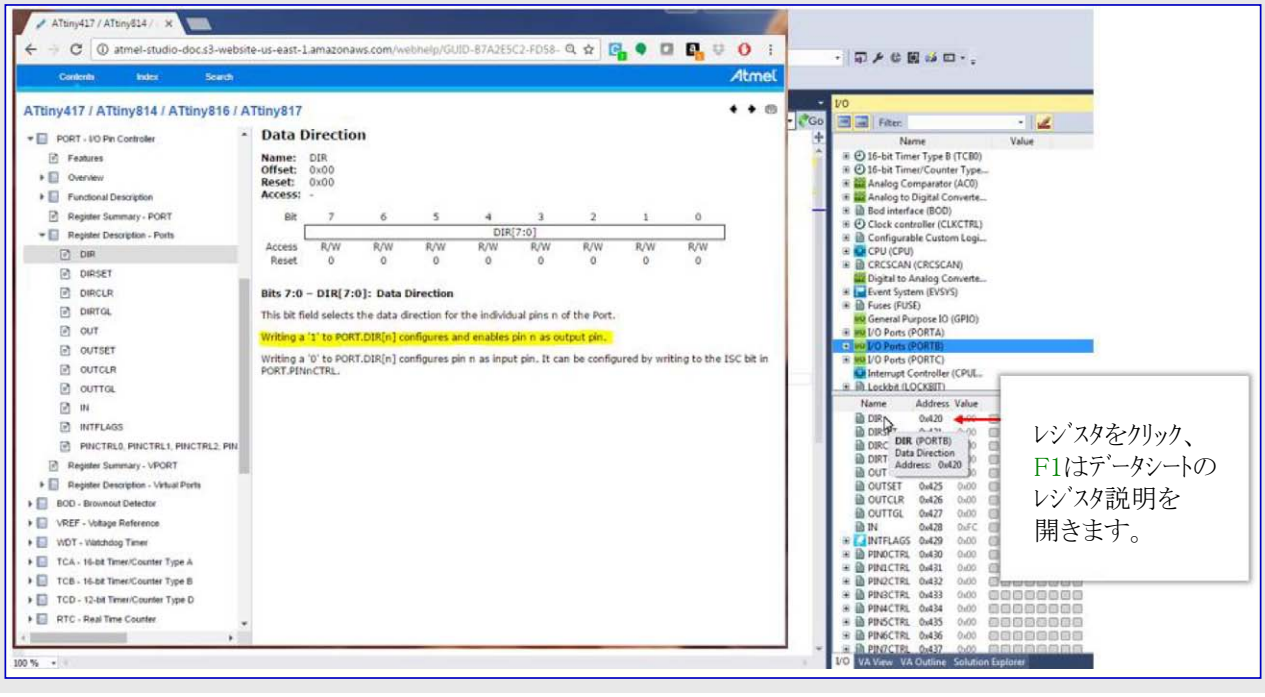
I/O表示部データシート

Atmel Studio 7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デ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.DIRn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. OUTnが'1'を書かれた場合、ピンnはLowを駆動します。

図1-17. 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「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。

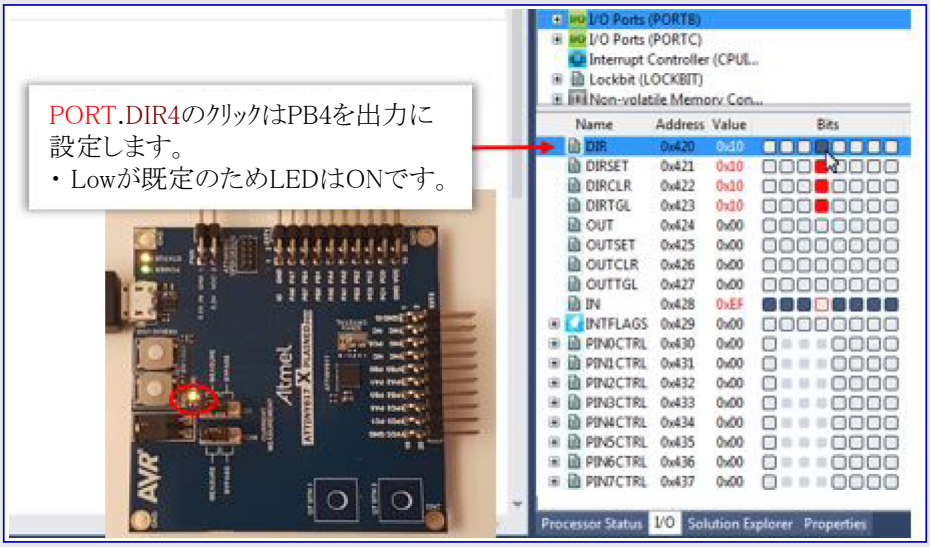
注: デバッグ時のI/O表示部は以下に使われます。

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1-2. Atmel Studio釦機能 (ﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞ作業開始)

釦	機能	ｷｰﾎﾞｰﾄﾞ ｼｮｰﾄｶｯﾄ
	ﾃﾞﾊﾞｯｸﾞを開始して中断	Alt + F5
	目的対象に取り付け	
	ﾃﾞﾊﾞｯｸﾞを開始	F5
	全て中断	Ctrl + Alt + Break
	ﾃﾞﾊﾞｯｸﾞなしで開始	Ctrl + F5

図1-18. ﾃﾞﾊﾞｯｸﾞ時にI/O表示部ﾚｼﾞｽﾀ操作を通してｷｯﾄのLEDをON/OFF切り替え



Atmel Studio7資料ﾀﾞｳﾝﾛｰﾄﾞ

ﾃﾞｰﾀｼｰﾄはAtmel Studio 7のﾍﾙﾌﾟ ｼｽﾃﾑを使うことによってもﾀﾞｳﾝﾛｰﾄﾞ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「ｵﾌﾗｲﾝ資料のﾀﾞｳﾝﾛｰﾄﾞ」で記述されます。

Atmel Studio 7 ｴﾃﾞｨﾀ (Visual Assist)

Visual Assistを装備したAtmel Studio 7のｴﾃﾞｨﾀはｺｰﾄﾞを書いて整理するだけでなく、大きなﾌﾟﾛｼﾞｪｸﾄの容易な誘導も手助けする協力な機能を持ちます。示唆機能は図1-19.で示され、一方でｺｰﾄﾞ誘導の概要は図1-20.で示されます。次の「ｴﾃﾞｨﾀ: ｺｰﾄﾞの記述と整理 (Visual Assist)」項ではｴﾃﾞｨﾀ機能がもっと詳細に網羅されます。

図1-19. ｺｰﾄﾞ書きに対するAtmel Studio 7のｴﾃﾞｨﾀでの示唆機能

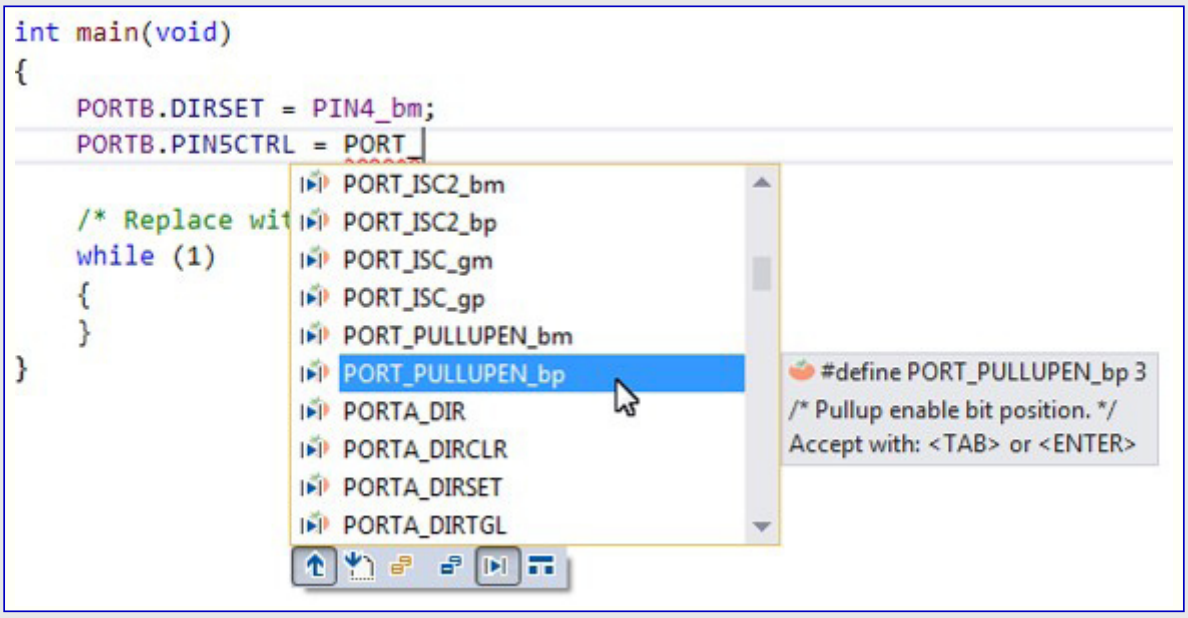


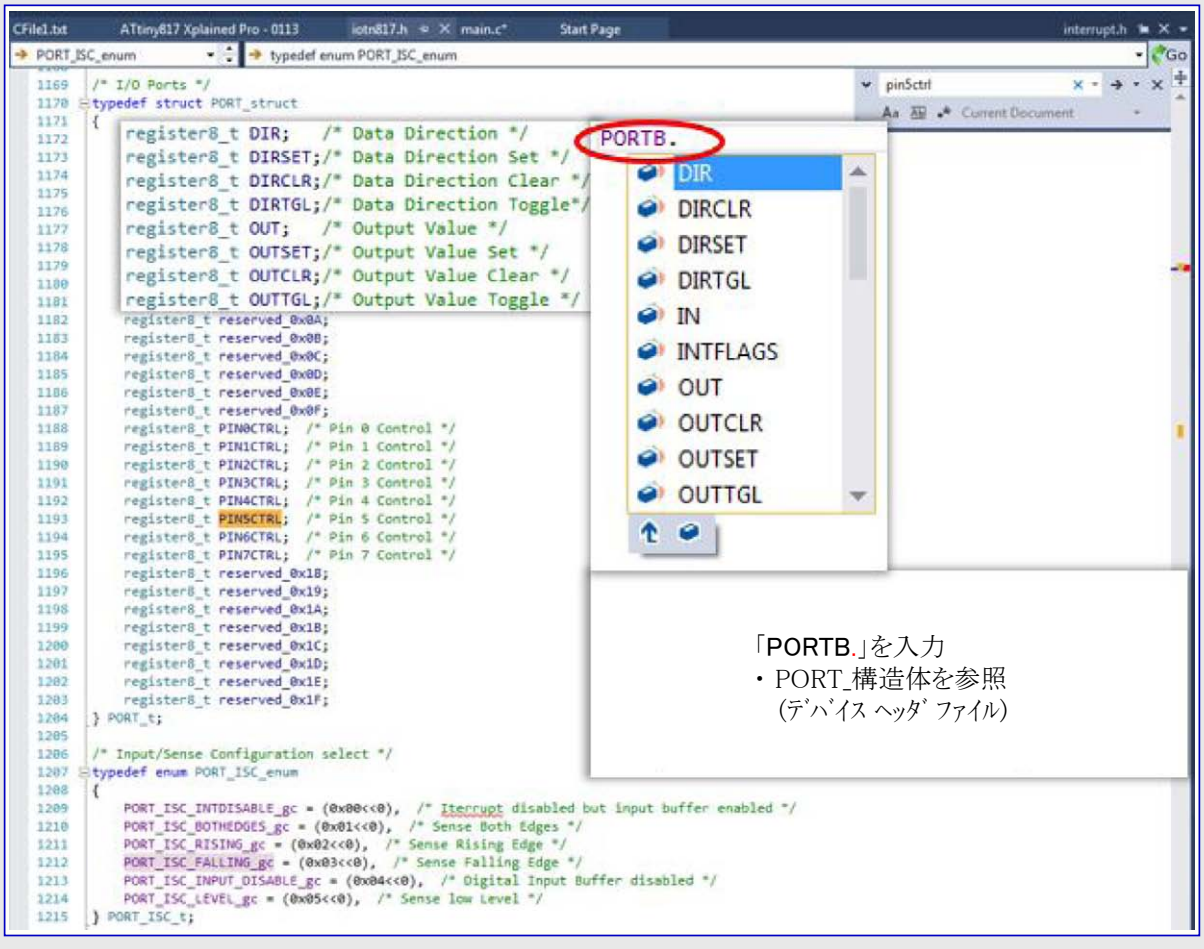
図1-20. Atmel Studio 7 ｴﾃﾞｨﾀの誘導概要

この項に関連する映像では具体的にｴﾃﾞｨﾀが以下のために使われます。

ﾃﾞﾊﾞｲｽ ﾍｯﾀﾞ ﾌｧｲﾙ

ｴﾃﾞｨﾀの移動定義機能を通して、即ち、どれかのﾚｼﾞｽﾀをｸﾘｯｸしてその後に移動(Go)釦をｸﾘｯｸするか、Alt+Gを入力することにより、MCUﾃﾞﾊﾞｲｽ ﾍｯﾀﾞ ﾌｧｲﾙを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ﾃﾞﾊﾞｲｽ ﾍｯﾀﾞ ﾌｧｲﾙ」で示される、PORT構造体から潜在的なﾚｼﾞｽﾀの示唆一覧を与えます。AVRﾍｯﾀﾞ 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1-21. 示唆一覧とMCUﾃﾞﾊﾞｲｽ ﾍｯﾀﾞ 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ﾋﾟﾝ接続を理解するのに有用です。完全な回路図とｶﾞｰﾊﾞ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1-22. 特定開発基板用回路図の探し方

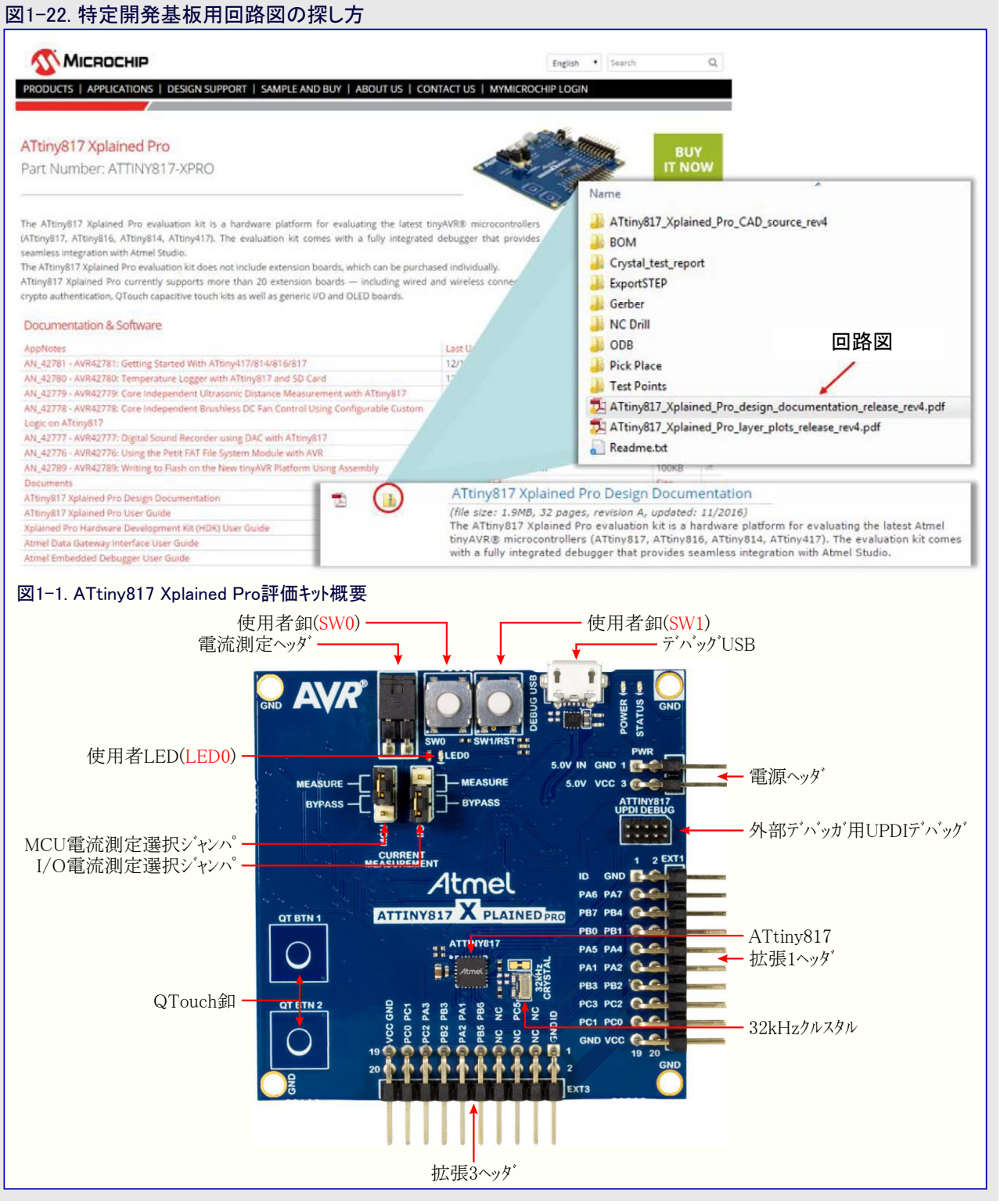


図1-1. ATtiny817 Xplained Pro評価ｷｯﾄ概要

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

表1-3. ATtiny817 Xplained Pro GPIO接続

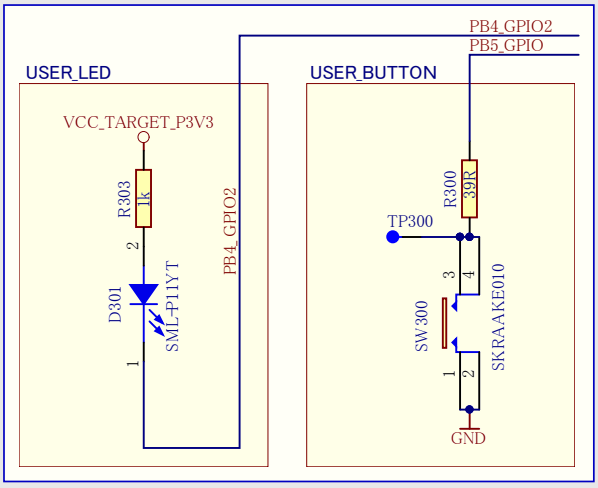
ｼﾙｸ文字	ATtiny817 GPIOﾋﾟﾝ
LED0	PB4
SW0	PB5

ATtiny817 Xplained Pro設計資料回路図は右図でのようにLEDと釦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は外部ﾌﾟﾙｱｯﾌﾟ抵抗を持ちません。
- SW0はATtiny817の内部ﾌﾟﾙｱｯﾌﾟ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1-23. 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ﾗｲﾌﾞ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)、ﾃﾞﾊﾞｯｸﾞ(avr-gdb,AVaRICE)の事業も伴います。

図1-24.で示されるように、ﾗｲﾌﾞﾗﾘ参考基準(Library Reference)は通常、AVR Libcへの迅速な遣り取り部です。関連ﾗｲﾌﾞﾗﾘに対して頁を迅速に検索することができます。ﾌﾟﾛｼﾞｪｸﾄに追加されるべき関連ﾍｯﾀﾞ 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、関連ｲﾝｸﾙｰﾄﾞは#include <avr¥interrupt.h>でしょう。単位部内でｸﾘｯｸすると、図1-25.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1-24. AVR Libcﾗｲﾌﾞﾗﾘ参照基準

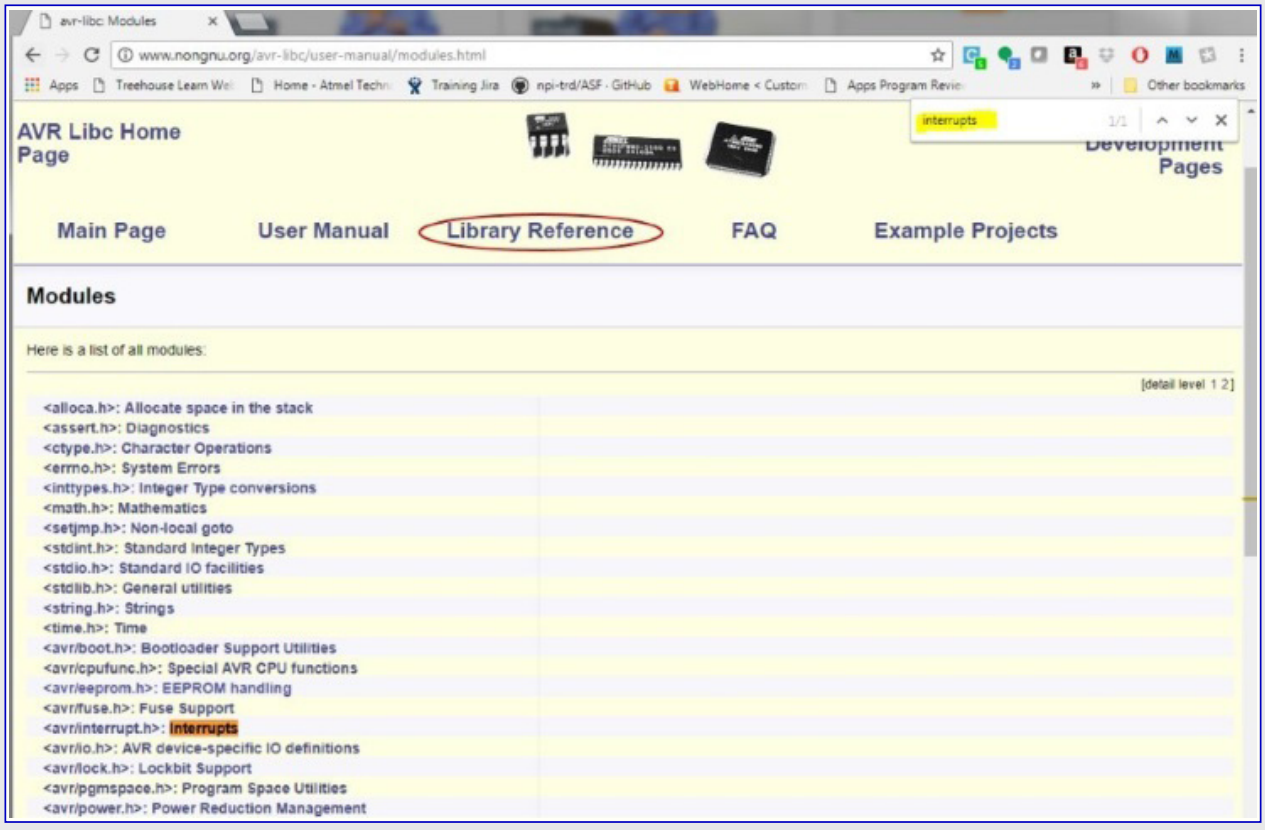


図1-25. AVR Libcでの割り込みの使い方

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ﾌﾞに基づくｿﾌﾄｳｪｱ構成ﾂｰﾙです。新しいﾌﾟﾛｼﾞｪｸﾄまたは例ﾌﾟﾛｼﾞ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誂えるためにﾄﾞﾗｲﾊﾞやﾐﾄﾞ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、生成されたｺｰﾄﾞ ﾌﾟﾛｼﾞｪｸﾄをﾀﾞｳﾝﾛｰﾄﾞしてAtmel Studio 7、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 Studio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ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ参照基準の一覧を使って0からｺｰﾄﾞ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ﾌﾟﾛｼﾞ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表示部はﾃﾞﾊﾞ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、構成ｺｰﾄﾞを見ることで、有用な部分をあなたのﾌﾟﾛｼﾞｪｸﾄに貼り付けることができます。例えば、図1-28.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1-26. 関連基板用新規ﾌﾟﾛｼﾞｪｸﾄ作成にAtmel STARTを使用

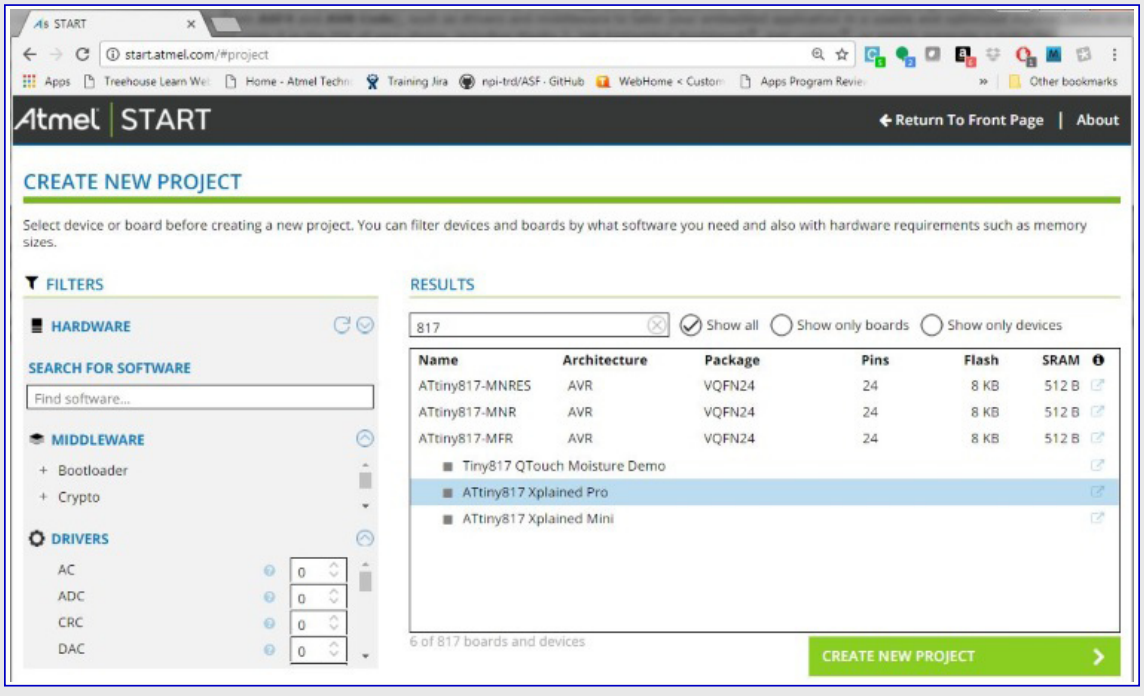


図1-27. キット回路図に対する代替としてAtmel STARTで基板分類表示

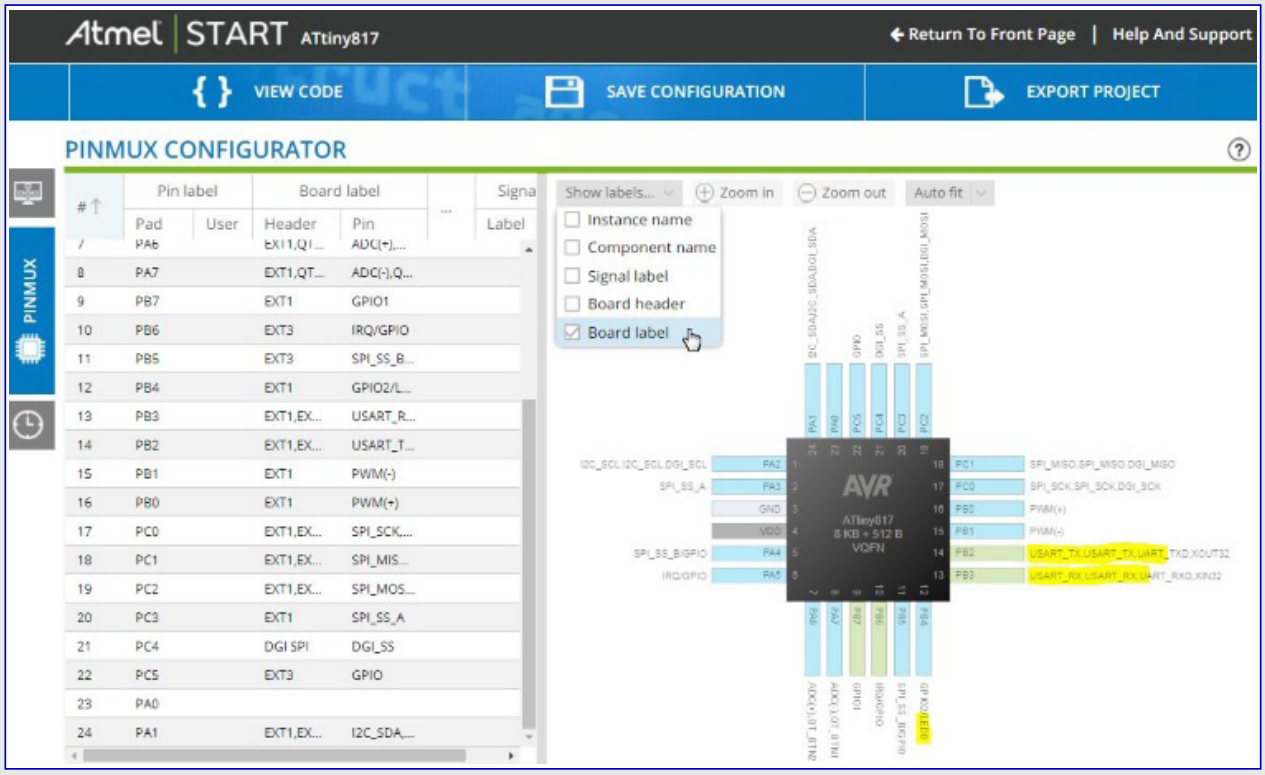
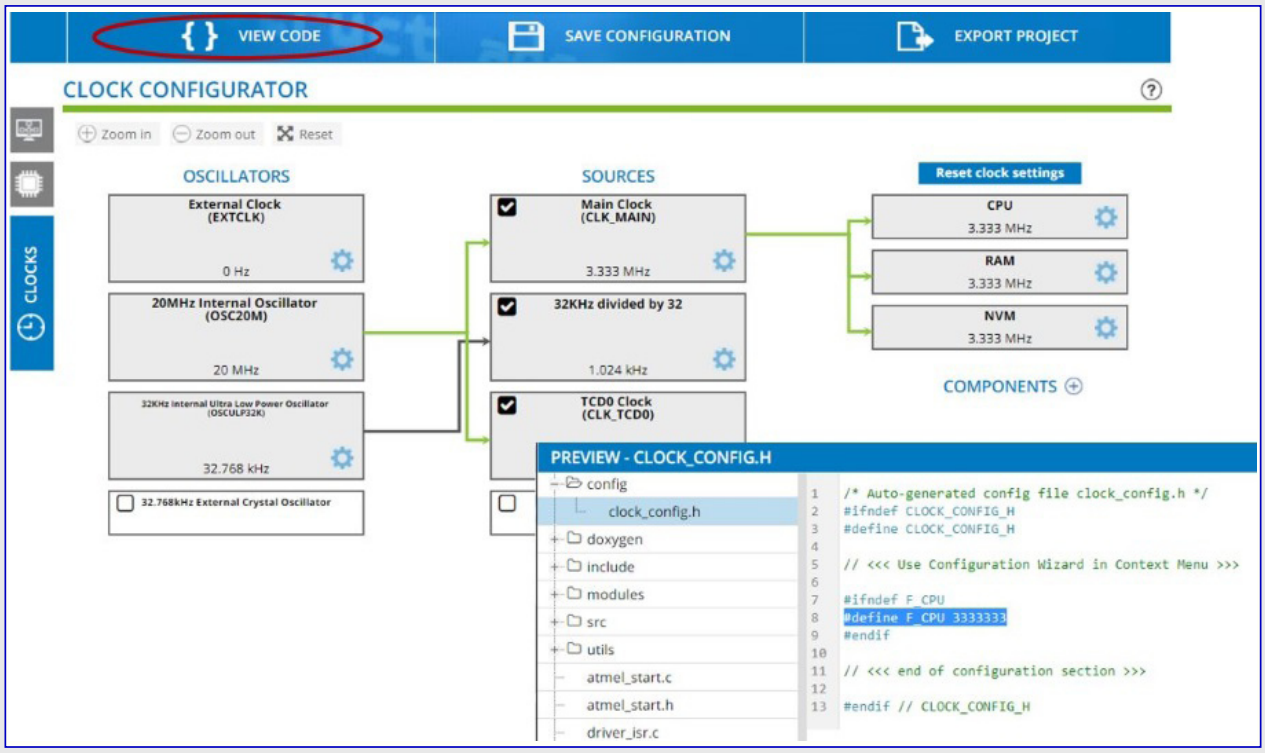


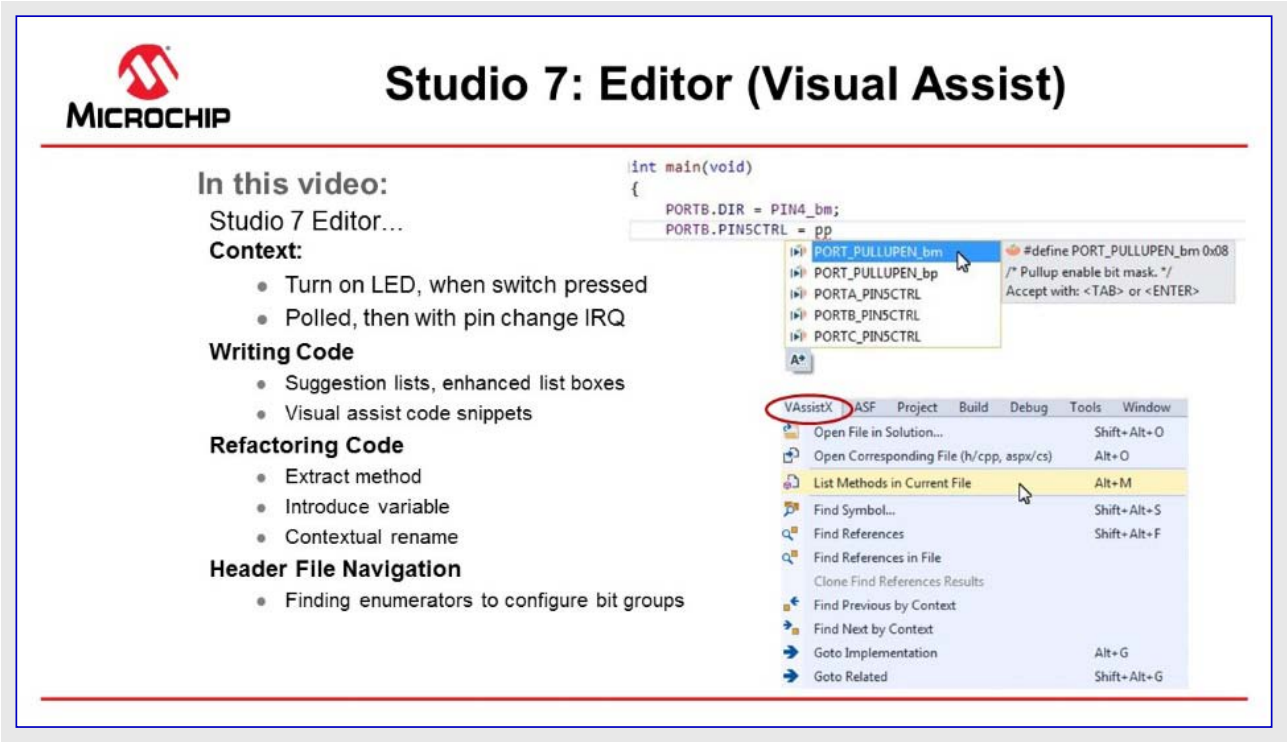
図1-28. 既定クロック構成を調べてF_CPU定義を見つけるのにVIEW CODEを使用



1.11. ｴﾃﾞｨﾀ: ｺｰﾄﾞの記述と整理 (Visual Assist)

Atmel Studio 7のｴﾃﾞｨﾀはCとC++のｺｰﾄﾞ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



映像: Atmel Studio 7 ｴﾃﾞｨﾀ (Visual Assist)

1. 「I/O表示部と他の空からのﾌﾟﾛｸﾞﾗﾐﾝｸﾞ参考資料」から基本的な機能で開始すると、main.cは以下のｺｰﾄﾞを持ちます。

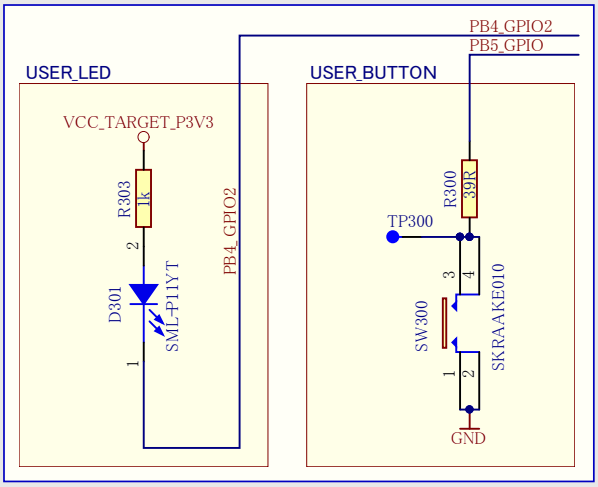
```
#include <avr/io.h>

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

    while (1)
    {
    }
}
```

ATtiny817 Xplained Pro設計資料回路図は右図でのようにLEDと釦に対する接続を示します。

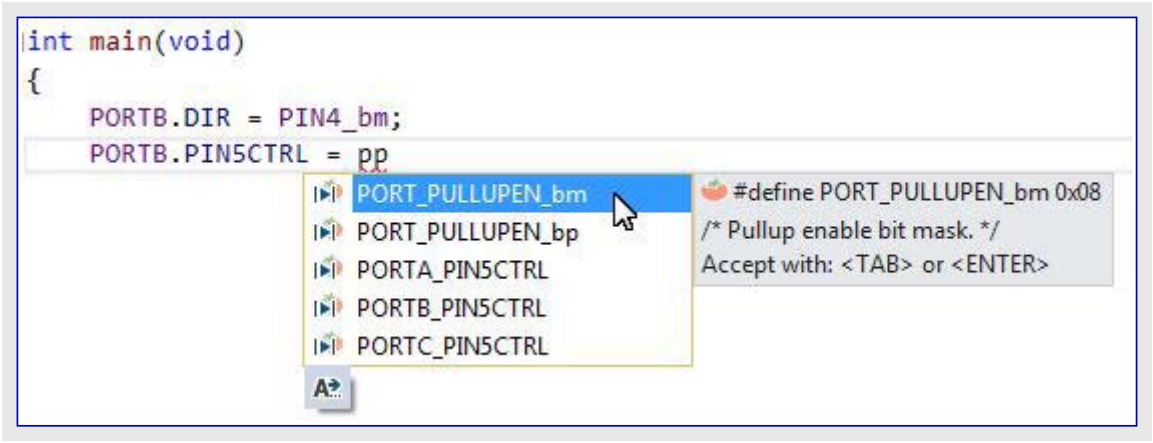
この回路図からは以下が断定されます。

- 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は外部ﾌﾟﾙｱｯﾌﾟ抵抗を持ちません。
- SW0はATtiny817の内部ﾌﾟﾙｱｯﾌﾟ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1-29. ATtiny817 Xplained Pro GPIO接続回路図

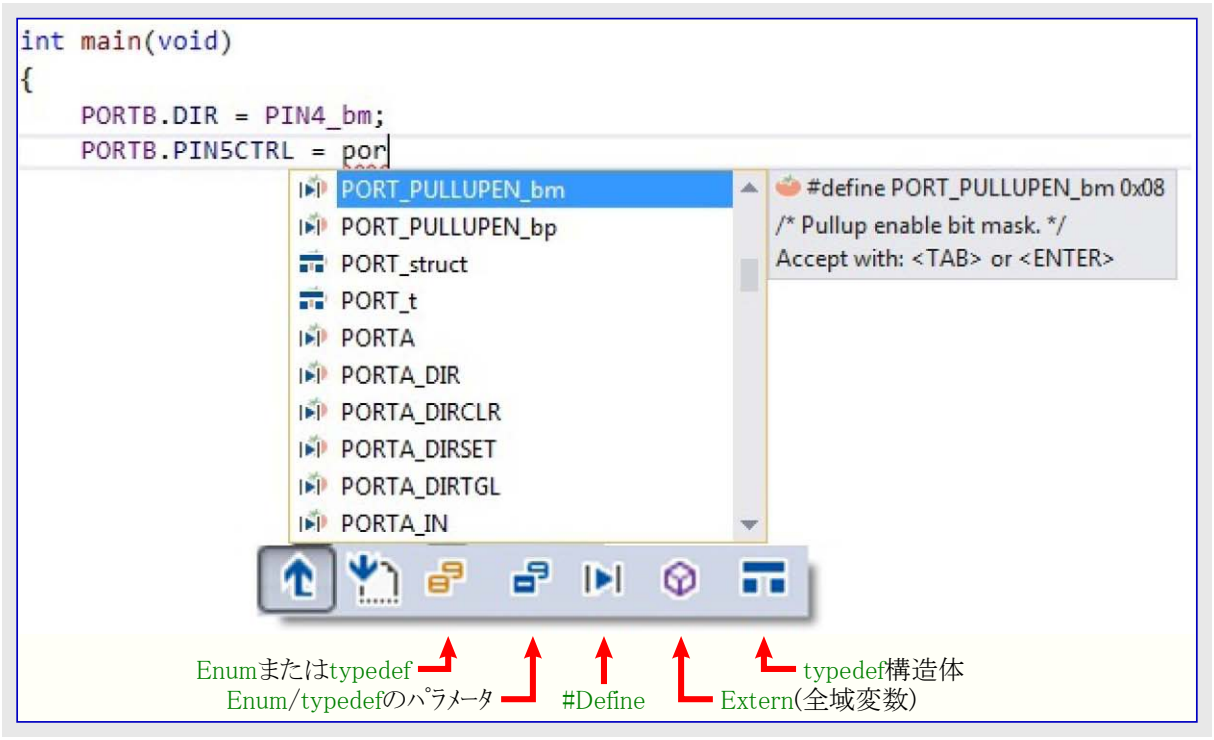


2. 示唆一覧と強化された一覧枠を用いてPORTB5のﾌﾟﾙｱｯﾌﾟを許可してください。示唆一覧は頭字語を支援し、故に”pp”と入力すると、PORT_PULLUPENが先頭示唆になることに注意してください。

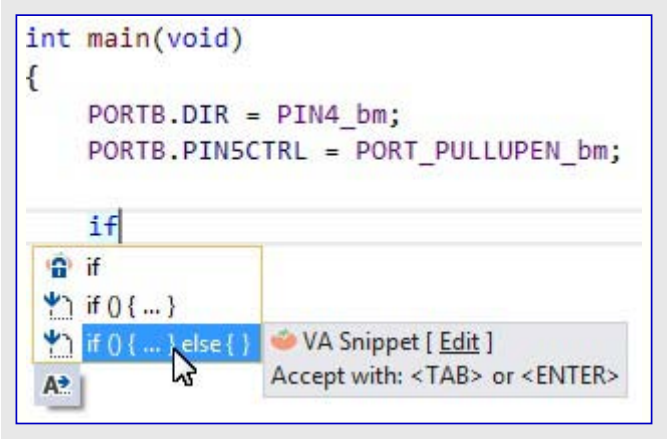


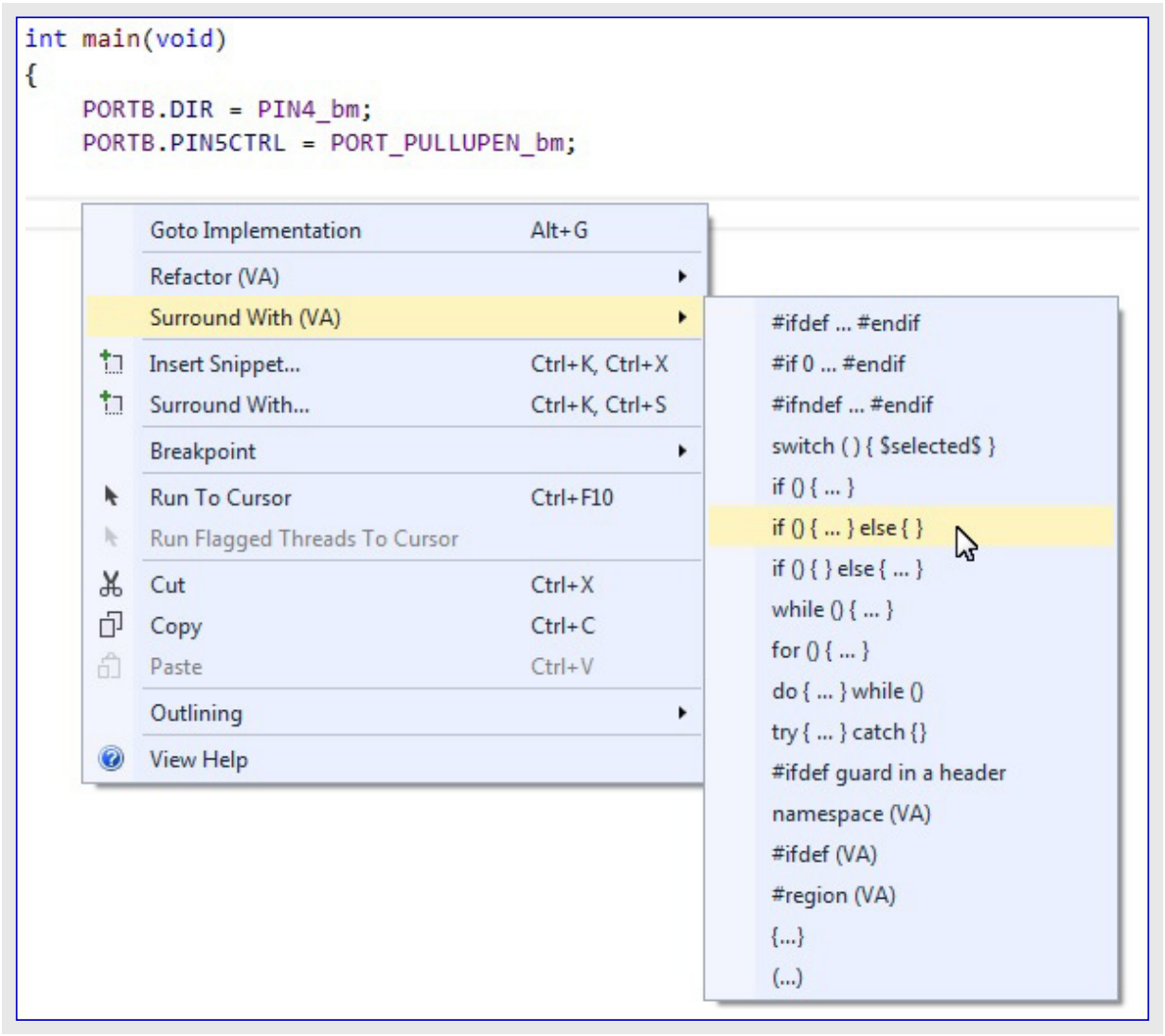
3. けれども、Enterを打つ前に、最初に”POR”を入力してその後にCtrl+Spaceを打ってください。これは可能な全ての任意選択を持つ強化された一覧枠を提示します。

   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。



4. if(){...}else{...} Visual assistｺｰﾄﾞ断片を用いてSW0が押されたかを検査してください。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。



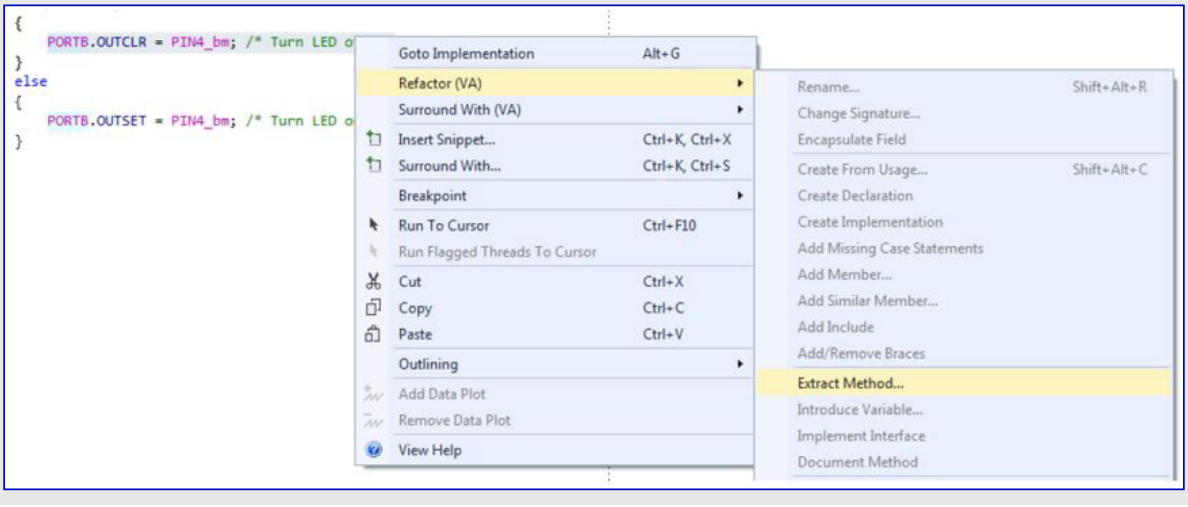


5. if(){...}else{...}条件としてｽｲｯﾁが押されたかを検査し、押された場合にLEDをONにそうでない場合にOFFにしてください。main.c は今や次のように見えるべきです。

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

int main(void)
{
    PORTB.DIRSET   = PIN4_bm;              /* LEDﾋﾟﾝを出力として構成設定 */
    PORTB.PIN5CTRL = PORT_PULLUPEN_bm;     /* SW0ﾋﾟﾝに対してﾌﾟﾙｱｯﾌﾟを許可  */

    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 */
        }
    }
}
```

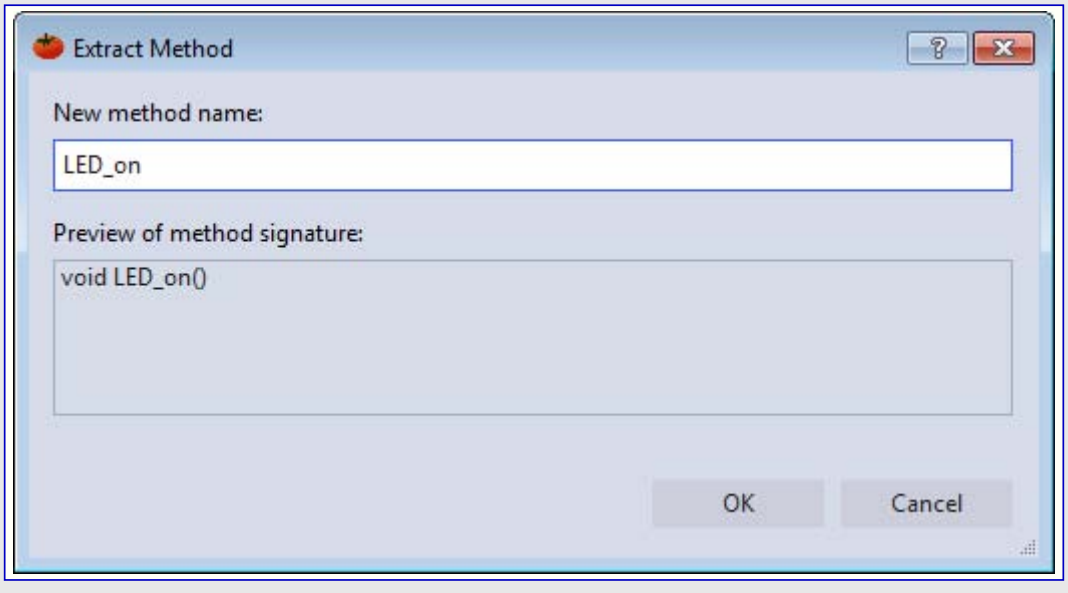
6. SW0押下時にLED0が点灯することを確認してください。ATtiny817 Xplained ProキットでSW0押下時にLED0が点灯することを確認するため、Start Without Debugging(ﾃﾞﾊﾞｯｸﾞなしで開始) (Ctrl+Alt+F5)をｸﾘｯｸすることによってｺｰﾄﾞを走らせてください。
   今や基本的な機能が整ったので、もっと読み易くするためにｺｰﾄﾞを整理しましょう。
7. Refactor(整理)⇒Extract Method(方式(ﾒｿｯﾄﾞ)抽出)を使ってLED_in()とLED_off()の関数を作成してください。LEDをONにするｺｰﾄﾞの行はSW0が押された時に実行されます。ｺｰﾄﾞのこの行を(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1-30. 方式抽出



Extract Method(方式抽出)ﾀﾞｲｱﾛｸﾞが現れます。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1-31. 方式抽出ﾀﾞｲｱﾛｸﾞ



OKをｸﾘｯｸし、ｺｰﾄﾞが変更されるべきです。使われるべきｺｰﾄﾞ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 Variablle(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ことが必要です。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1-32. 変数導入

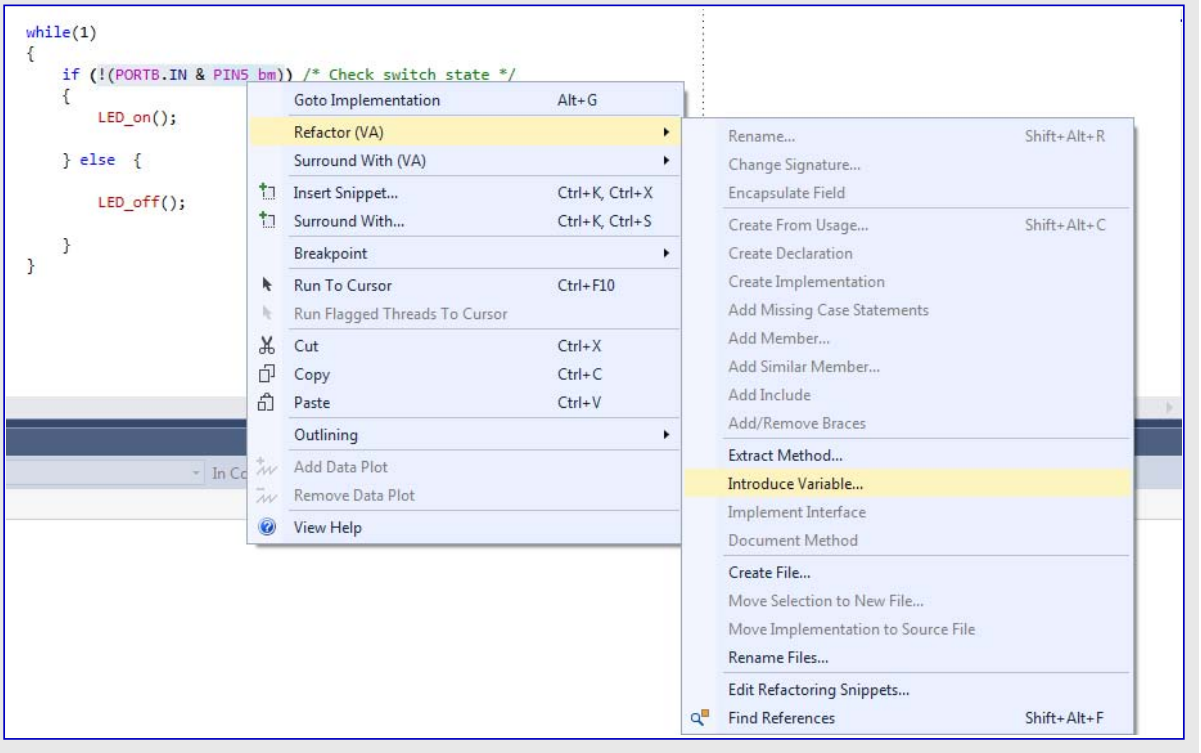
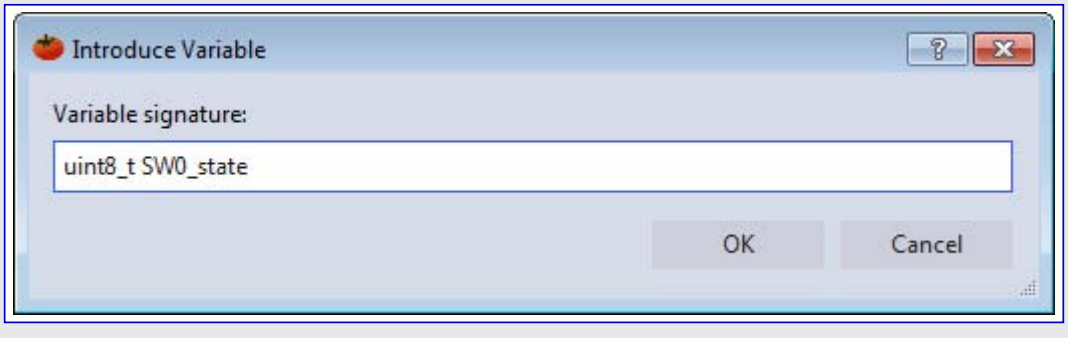


図1-33.で描かれるように、Introduce Variable(変数導入)ﾀﾞｲｱﾛｸﾞが現れます。変数を"uint8_t SW0_state"と名付けてください。

図1-33. 変数導入ﾀﾞｲｱﾛｸﾞ



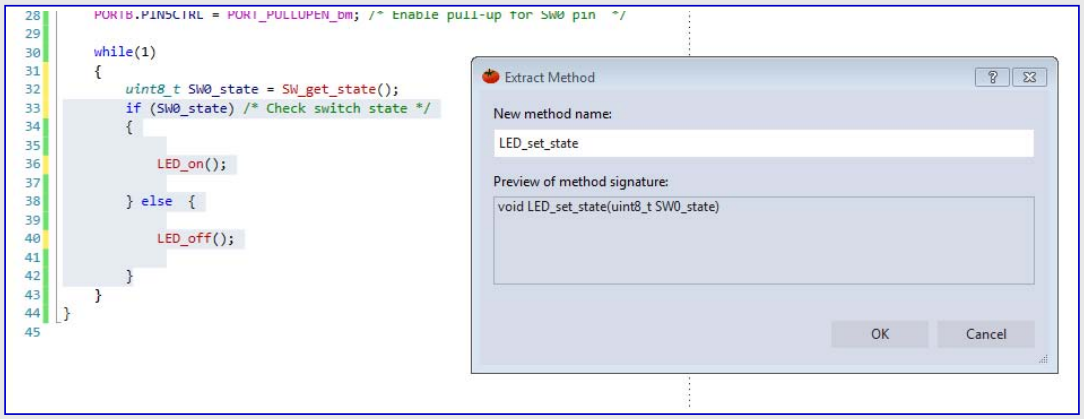
助言: ﾌﾞｰﾙ値で扱うのに追加のﾍｯﾀﾞをｲﾝｸﾙｰﾄﾞするのを避けるため、uint8_tに対する戻り値を自動的に生成したboolに変更してください。

OKをｸﾘｯｸし、ｺｰﾄﾞが変更されるべきです。下のｺｰﾄﾞ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(ﾒｿｯﾄﾞ)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1-34.**で示されるように、Atmel Studioは引数SW0_stateを検出します。

図1-34. 引数と共に方式を抽出



OKをｸﾘｯｸし、ｺｰﾄﾞ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(ﾒｿｯﾄﾞ)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。Atmel 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ために使うことができます。**図1-35.**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1-35. 脈絡での改名

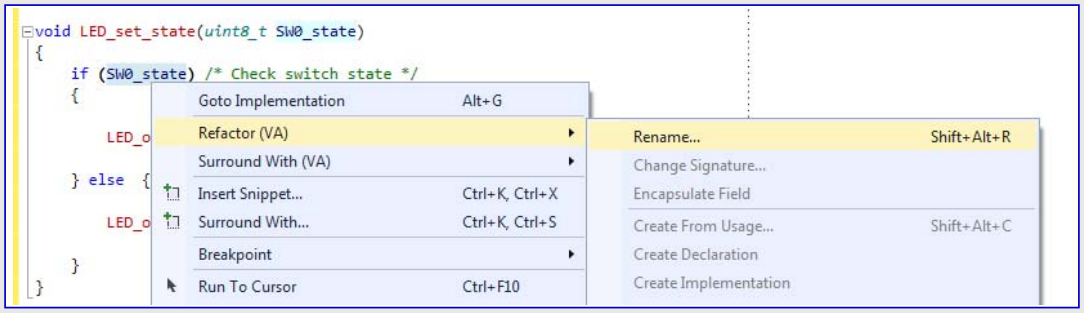
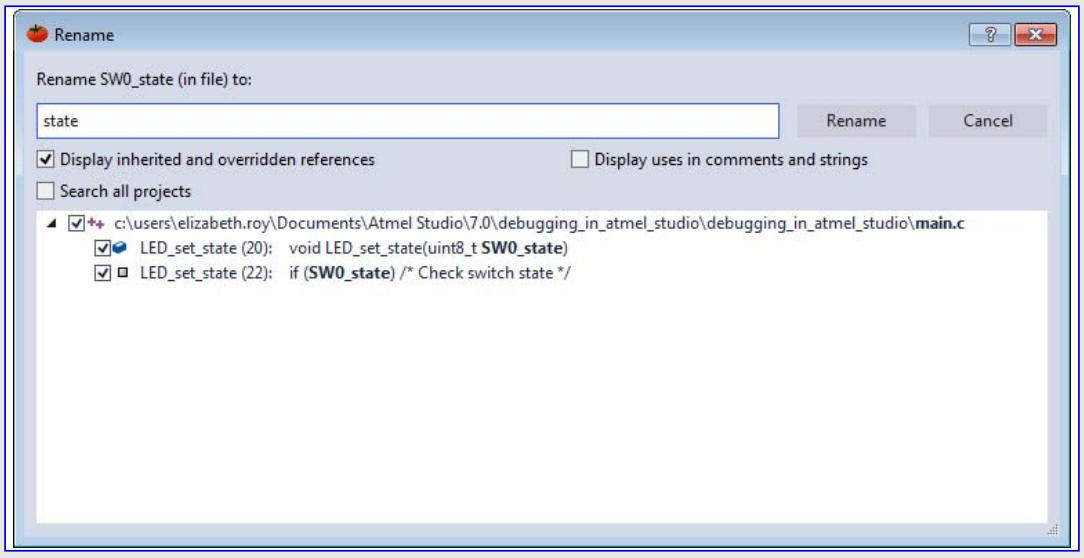


図1-36.で描かれるように、Rename(改名)ﾀﾞｲｱﾛｸﾞが現れます。SW0_state変数を"state"に改名してください。Atmel Studioは選択されたものと同じ脈絡を持つ変数の全ての存在を検出し、それが一覧で提示され、個別に選択または解除を行うことができます。

図1-36. 脈絡での改名ﾀﾞｲｱﾛｸﾞ



Rename(改名)をｸﾘｯｸし、ｺｰﾄﾞが変更されるべきです。LED_se t_state()の引数と関数内のそれ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。

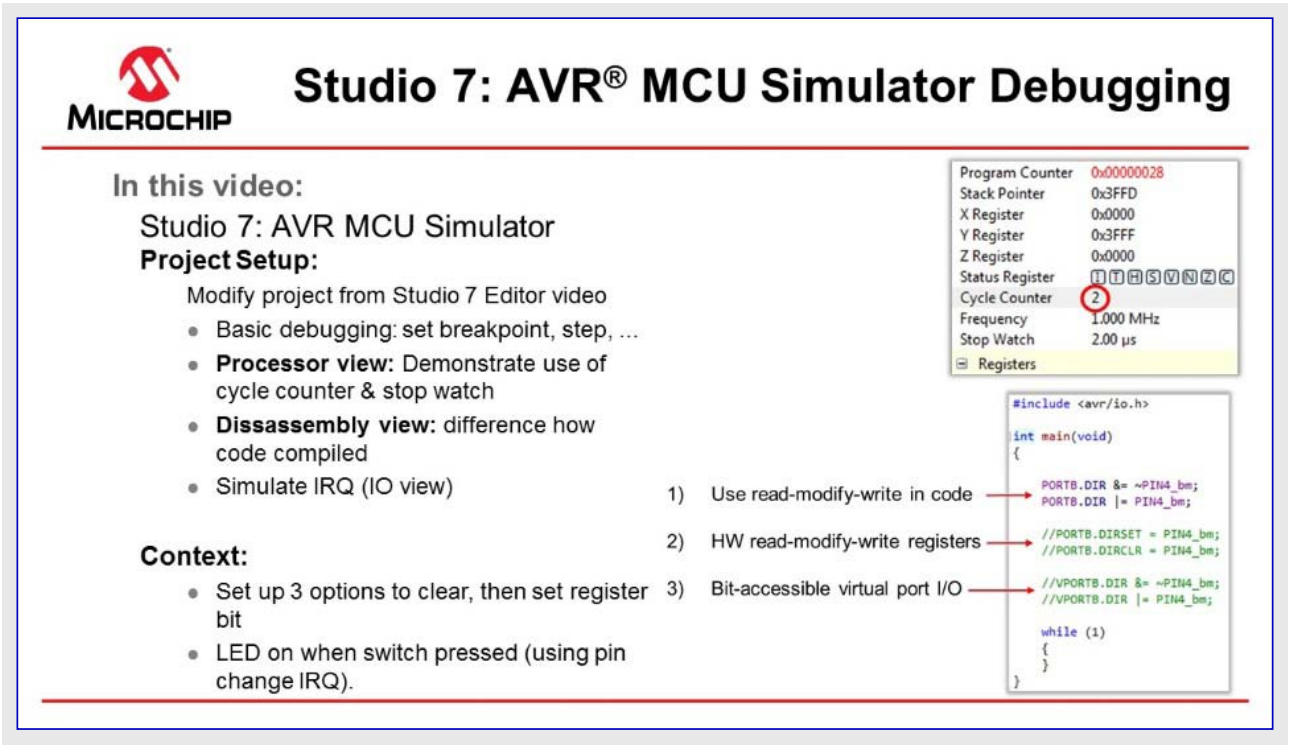
12. 作成した関数をmain()の下に移動して関数定義を作成してください。main.cは今や次のように見えるべきです。

```
#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ﾋﾟﾝを出力として構成設定 */
    PORTB.PIN5CTRL = PORT_PULLUPEN_bm;          /* SW0ﾋﾟﾝに対してﾌﾟﾙｱｯﾌﾟ許可 */

    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();
    }
}
```

1.12. 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)設定とコードを通した段階実行)を実演します。割り込みを模倣する方法も示します。

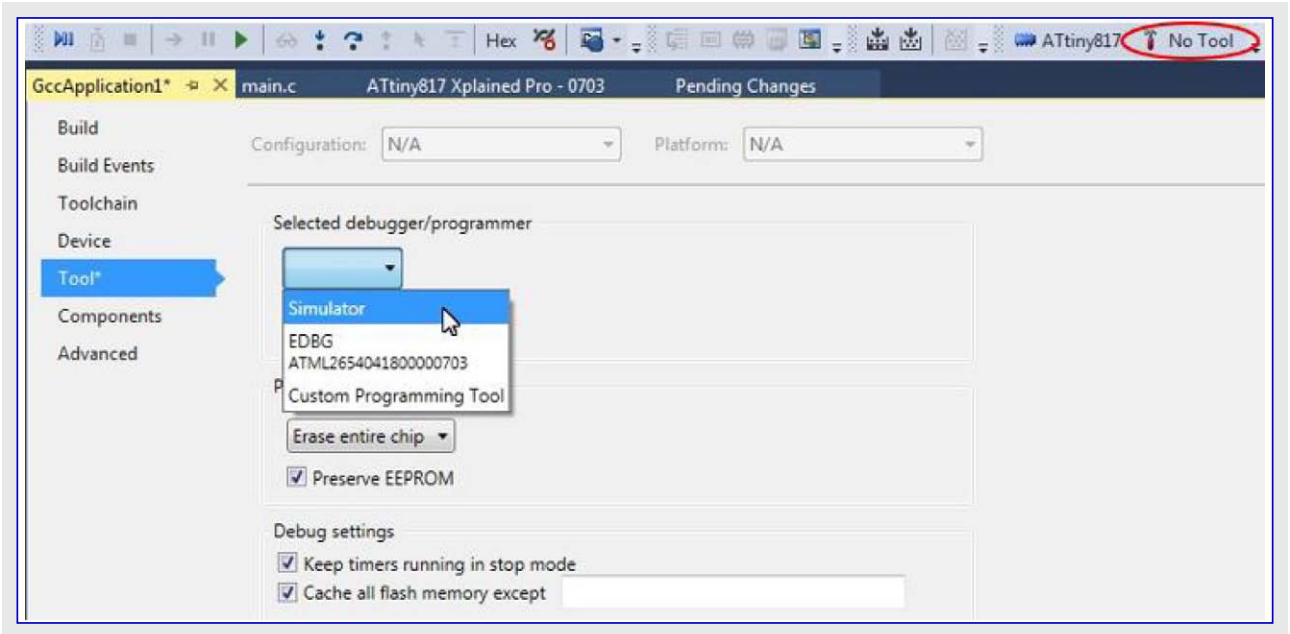
開始に際しての話題



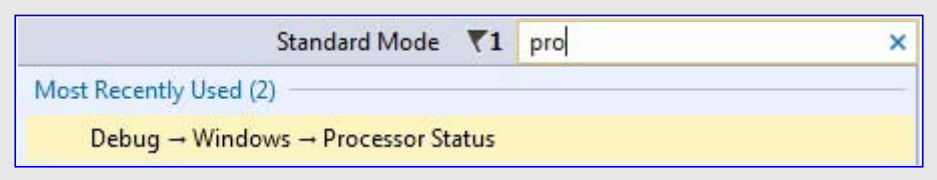
映像: AVRシミュレータ デバッグ

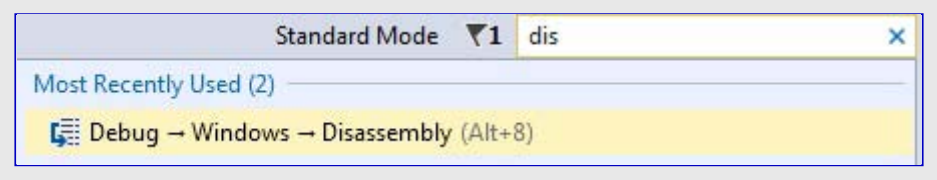
上の映像で使われたコードは「エディタ: コードの記述と整理 (Visual Assist)」の映像で書かれました。

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。

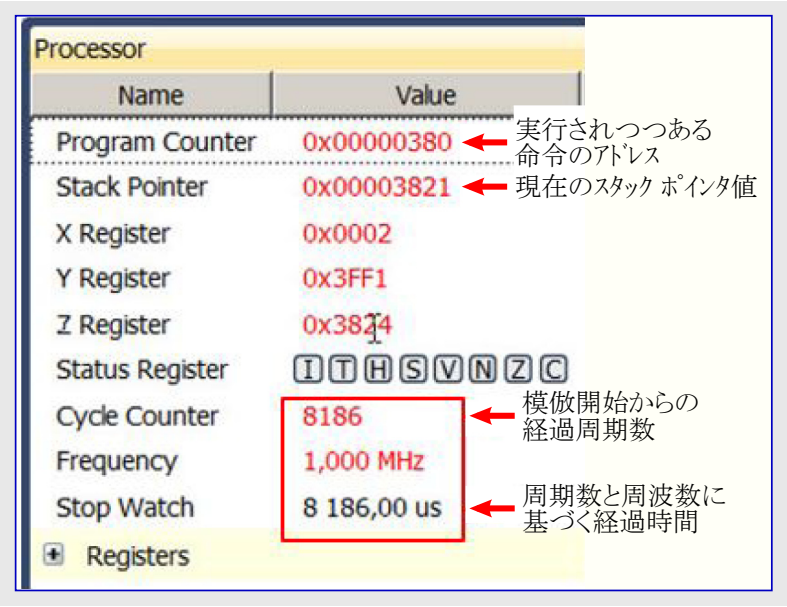


Cycle Counter(周期計数器)とStopwatch(ストップウォッチ)はシミュレータでだけ利用可能です。これらを使うには最初にデバッグ作業を開始するためにStart Debugging and Break(デバッグを開始して中断) をクリックし、その後に敏速起動バーに"Processor"を入力してEnterを打つことによってProcessor Status(プロセッサ状態)ウィンドウを開いてください(また、これはDebug(デバッグ)⇒Windows(ウィンドウ)⇒Processor Status(プロセッサ状態)下で見つけることができます)。同様に、Disassembly(逆アセ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

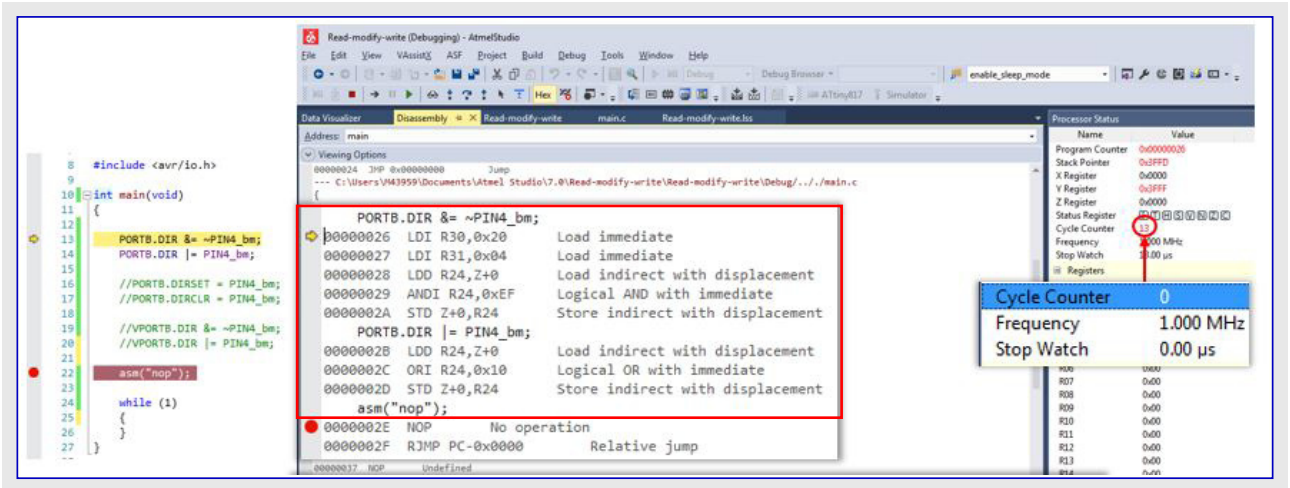




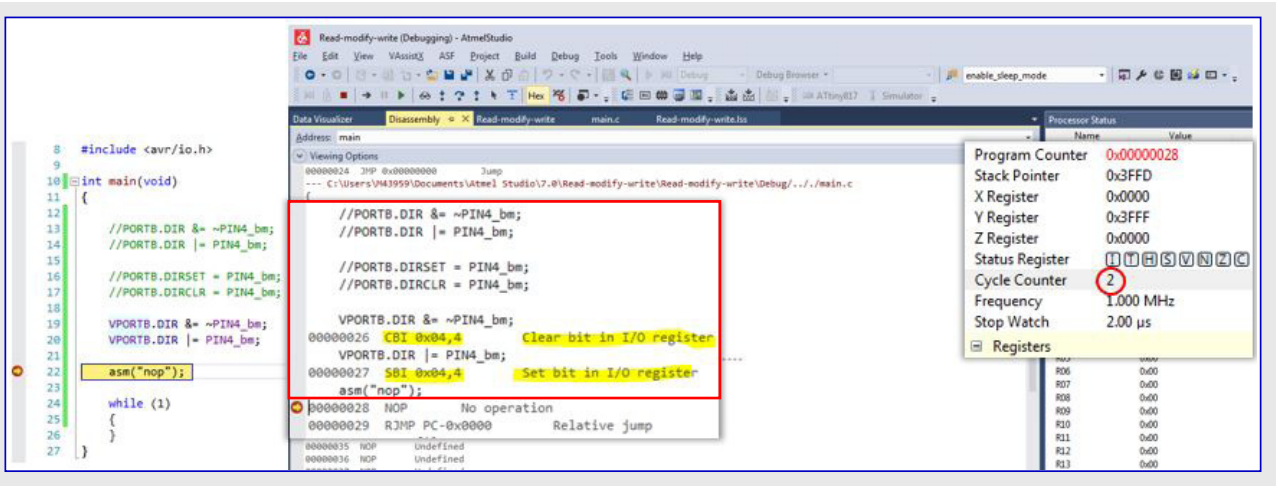
AVRシミュレータは実デバイスを作るのに使われるのと同じRTLコードに基づく模式を使っています。これはバグと遅延の両方に対してCycle Counter(周期計数器)を正確にします。Stopwatch(ストップウォッチ)はFrequency(周波数)に関連され、それは値上のダブル クリックと使いたいクロック周波数を入力することによって設定することができることに注意してください。



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。



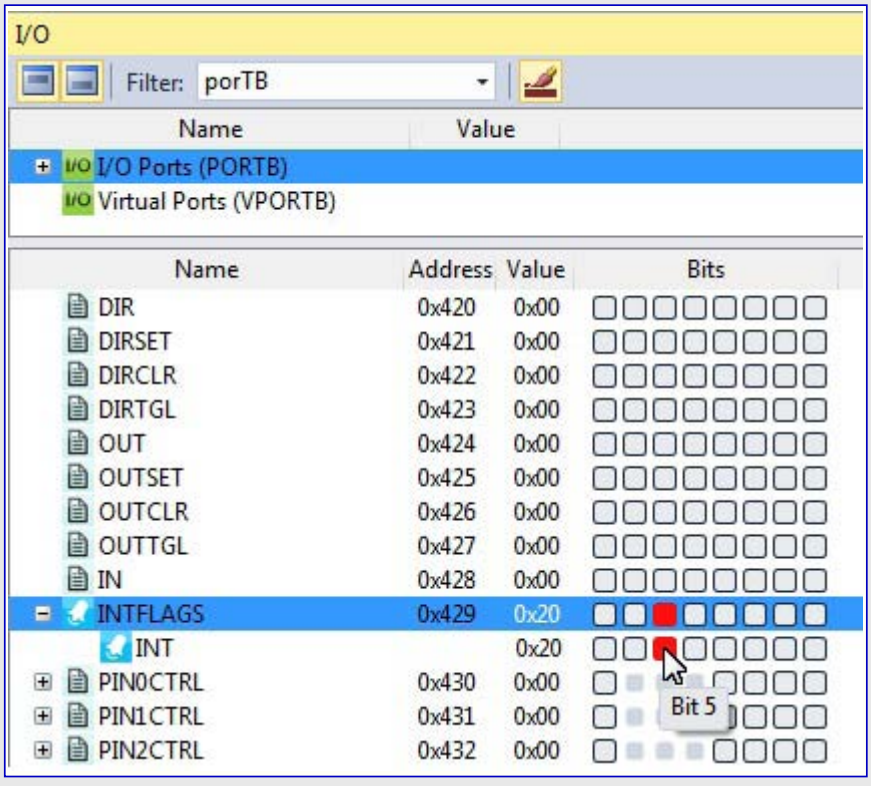
(訳補:周期計数器の違いとなる)仮想ポート レジスタでのソフトウェア読み−変更−書き(前図)とビット操作命令(次図)間で生成されたアセンブリコードでの違いに気付いてください。



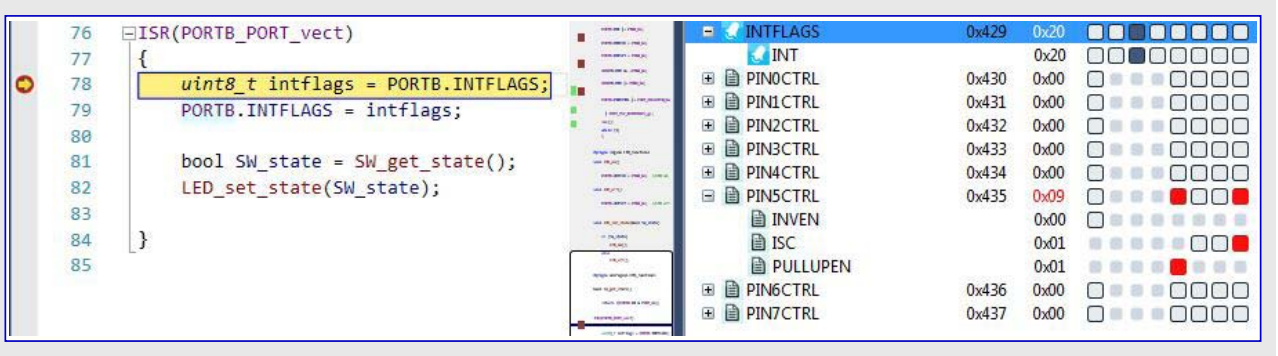
これら3つの方法を比べた結果が右表で要約されます。

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。



ﾋﾟﾝ変化割り込みはI/O表示部でﾎﾟｰﾄ入力ﾚｼﾞｽﾀに書くことによって起動することもできます。ﾎﾟｰﾄ入力ﾚｼﾞｽﾀへのﾋﾞｯﾄ書き込みはﾃﾞﾊﾞｲｽ外囲器の物理ﾋﾟﾝへその値を印加するのと同じです。内部ﾎﾟ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。

Atmel Studio 7の標準ﾃﾞﾊﾞｯｸﾞ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ﾌﾟ上ﾃﾞﾊﾞｯｸﾞ能力が欠けていてﾊｰﾄﾞｳｪｱ ﾃﾞﾊﾞｯｶﾞを用いてﾃﾞﾊﾞｯｸﾞすることができないﾃﾞﾊﾞｲｽでも利用可能です。この開始の手引きのﾃﾞﾊﾞｯｸﾞ章をご覧ください。

(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ﾄﾞ

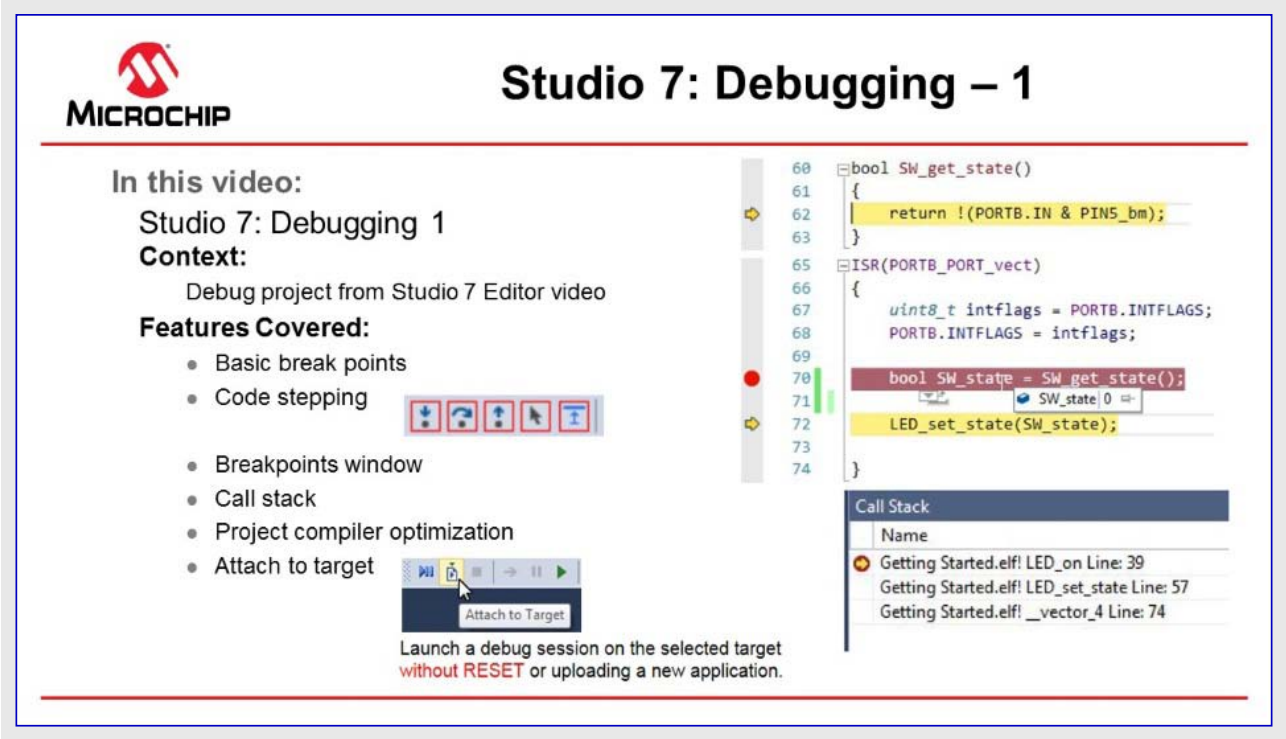
```
#include <avr/io.h>
#include <stdbool.h>
#include <avr/interrupt.h>

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 LED_functions

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);
}
```

1.13. ﾃﾞﾊﾞｯｸﾞ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Atmel Studio 7のﾃﾞﾊﾞｯｸﾞ能力を紹介します。主な話題は中断点(ﾌﾞﾚｰｸ ﾎﾟｲﾝﾄ)、中断点を使う基本的なｺｰﾄﾞ段階実行、呼び出しｽﾀｯｸ ｳｨﾝﾄﾞｳだけでなく、更にｺﾝﾊﾟｲﾗ最適化設定調整もです。

開始に際しての話題



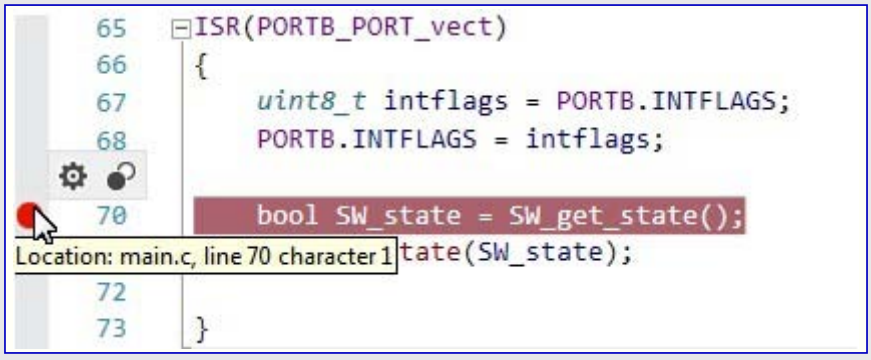
映像: Atmel Studio 7 ﾃﾞﾊﾞｯｸﾞ-1

「ｴﾃﾞｨﾀ: ｺｰﾄﾞの記述と整理 (Visual Assist)」項で作成されたのと同じｺｰﾄﾞが使われます。

すべきこと: 中断点を配置してﾌﾟﾛｼﾞｪｸﾄ内の全ての中断点の一覧を調べてください。

1. 図1-37.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1-37. 中断点を配置

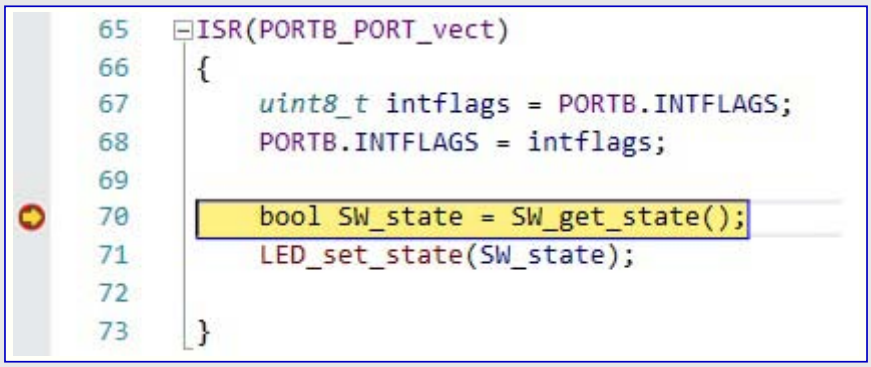


情報: 中断点は以下によってｺｰﾄﾞの行に配置することができます。

- ｴﾃﾞｨﾀ ｳｨﾝﾄﾞｳ左端の灰色ﾊﾞｰをｸﾘｯｸ。
- 最上部のﾒﾆｭｰ ﾊﾞｰで、Debug(ﾃﾞﾊﾞｯｸﾞ)⇒Toggle Breakpoint(中断点ON/OFF)へ行く。
- ｷｰﾎﾞｰﾄﾞでのF9押下による。

2. ▶でﾃﾞﾊﾞｯｸﾞ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ﾄﾞの行が正に実行しようとしていることを示します。図1-38.をご覧ください。

図1-38. 中断点に的中した時の実行停止



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、Atmel Studioは一時区画でﾌｧｲﾙを開きます。ﾃﾞﾊﾞｯｸﾞ作業で的中した中断点を含むﾌｧｲﾙは常にﾌｧｰｶｽを持ちます。

3. ﾌﾟﾛｸﾞ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、今やﾌﾟﾛｸﾞ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。

ｺｰﾄﾞ段階実行はこの過程を調べるのに使うことができます。ｺｰﾄﾞ段階実行に使われるｷｰ釦は下表で説明され、最上部のﾒﾆｭｰﾊﾞｰまたはDebug(ﾃﾞﾊﾞｯｸﾞ)ﾒﾆｭｰで見つけてください。対応する機能とｷｰﾎﾞｰﾄﾞ ｼｮｰﾄｶｯﾄは右図で略述されます。

図1-39. ｺｰﾄﾞ段階事項用のAtmel Studioの釦

表1-4. Atmel Studio釦機能 (ｺｰﾄﾞ段階実行)

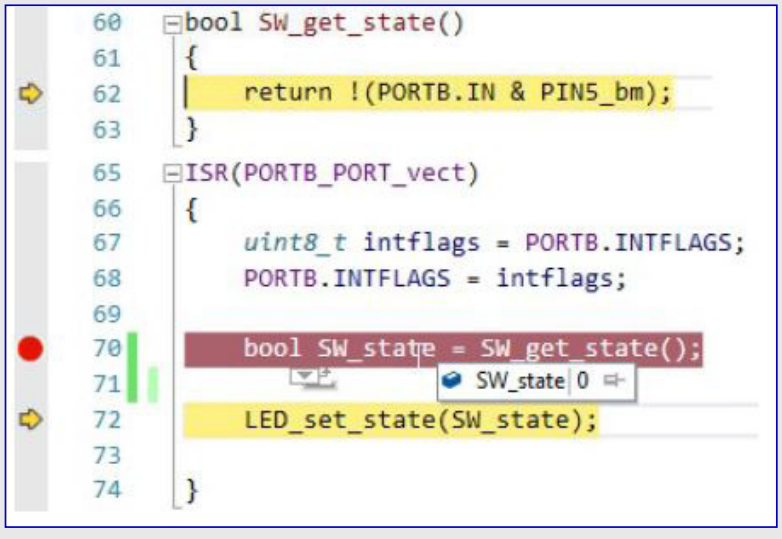
釦	機能	ｷｰﾎﾞｰﾄﾞ ｼｮｰﾄｶｯﾄ
	関数呼び出し内段階実行	F11
	外側段階実行	F10
	関数外へ段階実行	Shift + F11
	ｶｰｿﾙまで走行	Ctrl + F10
	ｼｽﾃﾑ ﾘｾｯﾄ発行	

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にされか否かを決めるのにLED_set_state(SW_state)関数内を更に段階実行することができる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。

情報: ｽｲｯﾁ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。

図1-40. ﾏｳｽ浮かしを使うSW_stateの値調査



1. ﾌﾟﾛｸﾞﾗﾑ上の中断点を把握するためのｳｨﾝﾄﾞｳまたは表示部が必要とされます。敏速起動ﾊﾞｰはAtmel Studio 7使用者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1-41.**と**図1-42.**の2つの図を比べることにより、下で実演されます。敏速起動ﾊﾞｰでの各的中がDebug(ﾃﾞﾊﾞｯｸﾞ)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1-41. 敏速起動ﾊﾞｰでの”break”検索

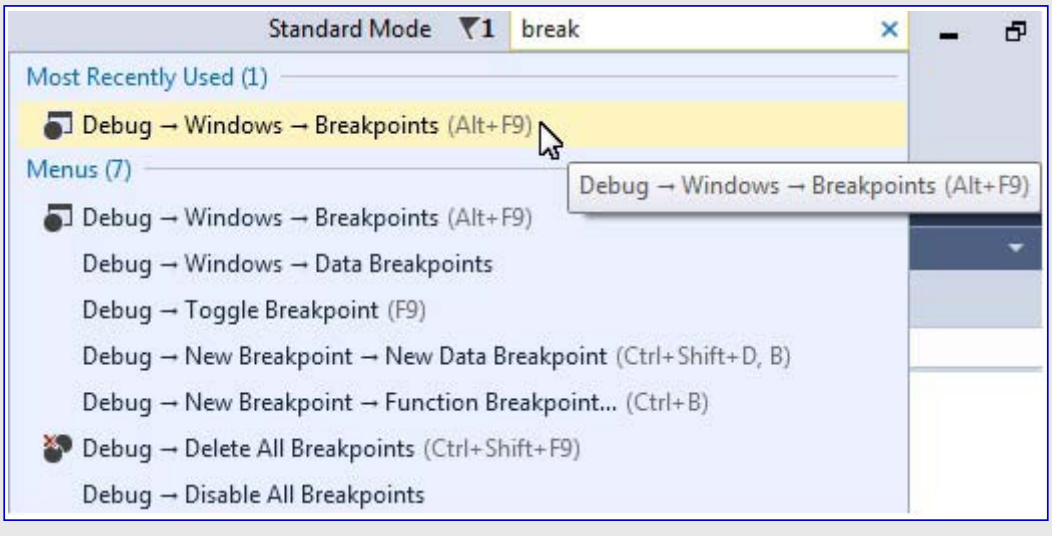
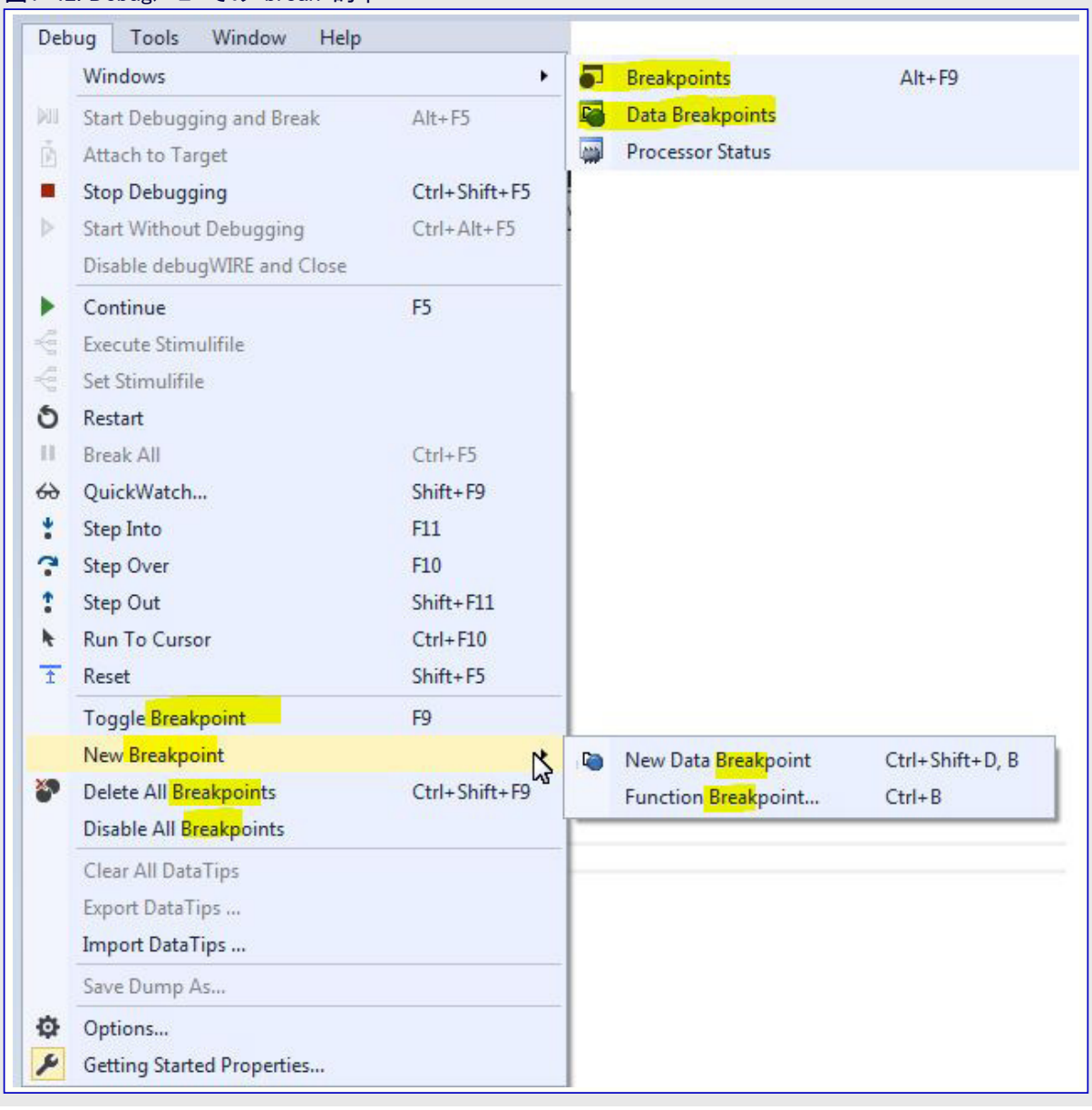


図1-42. Debugﾒﾆｭｰでの”break”的中



先頭の結果(Debug(ﾃﾞﾊﾞｯｸﾞ)⇒Windows(ｳｨﾝﾄﾞｳ)⇒Breakpoints(中断点))でのｸﾘｯｸによってBreakpoints(中断点)ｳｨﾝﾄﾞｳを開いてください。図1-43.で描かれるように、中断点ｳｨﾝﾄﾞｳは現在の的中数と共にﾌﾟﾛｼﾞ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1-44.部で実演されるように、逆ｱｾﾝﾌﾞﾘ表示部はｿｰｽ ｺｰﾄﾞと一緒に便利に表示することができます。

図1-43. Breakpoints(中断点)ｳｨﾝﾄﾞｳ

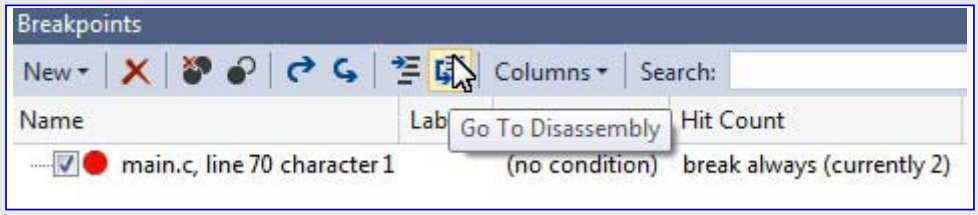
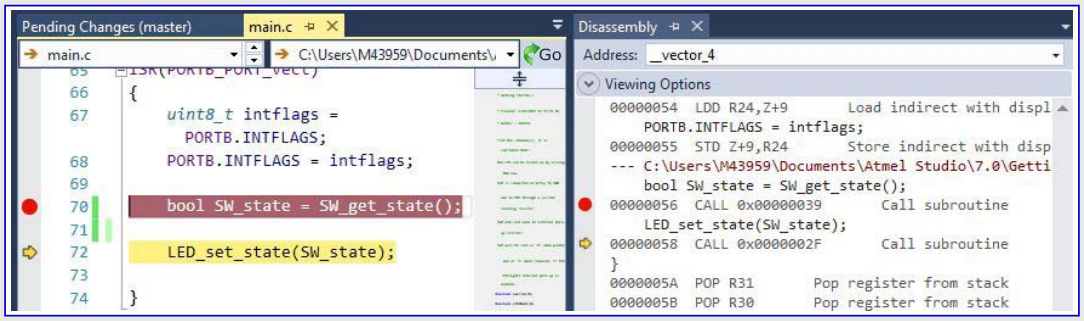


図1-44. 逆ｱｾﾝﾌﾞﾘ表示部

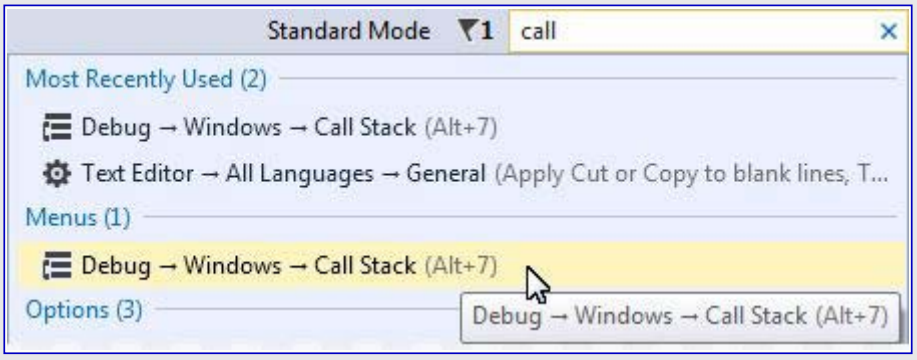


すべきこと: 呼び出しｽﾀｯｸと最適化が禁止された時のそれの影響を調べてください。

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1-45.で表されるように、敏速起動ﾊﾞｰで"call"を入力してDebug(ﾃﾞﾊﾞｯｸﾞ)⇒Windows(ｳｨﾝﾄﾞｳ)⇒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)ｳｨﾝﾄﾞｳを開いてください。

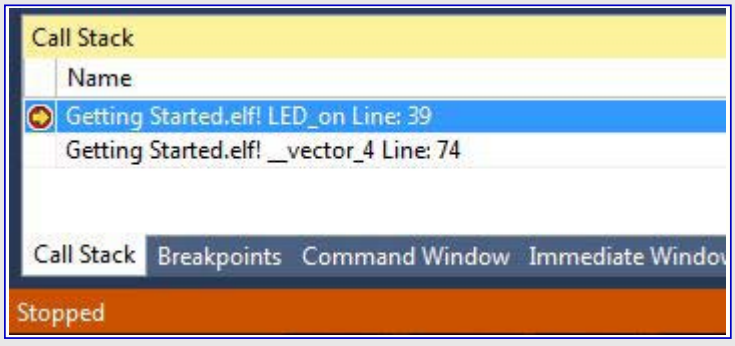
   注: このｳｨﾝﾄﾞｳを開くにはﾃﾞﾊﾞｯｸﾞ作業を活性(有効)にする必要があります。

図1-45. Call Stack(呼び出しｽﾀｯｸ)を開く



3. それはｺｰﾄﾞ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 ｳｨﾝﾄﾞｳでは(図1-46.でのように)呼び出し元として_vector_4が一覧にされ、これはｺﾝﾊﾟｲﾗの最適化のためです。

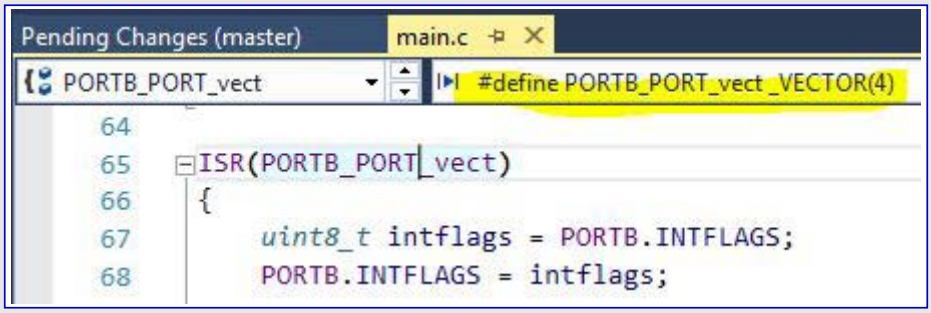
図1-46. 最適化での呼び出しｽﾀｯｸ



情報: 呼び出し指示はｺﾝﾊﾟｲﾗ最適化のために異なります。このｺｰﾄﾞは理解するのが比較的簡単で、例えｺﾝﾊﾟｲﾗで最適化され、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。もっと複雑なﾌﾟﾛｼﾞｪｸﾄではﾊﾞｸﾞを探し出すのに時にはｺﾝﾊﾟｲﾗの最適化を禁止することが役立つかもしれません。

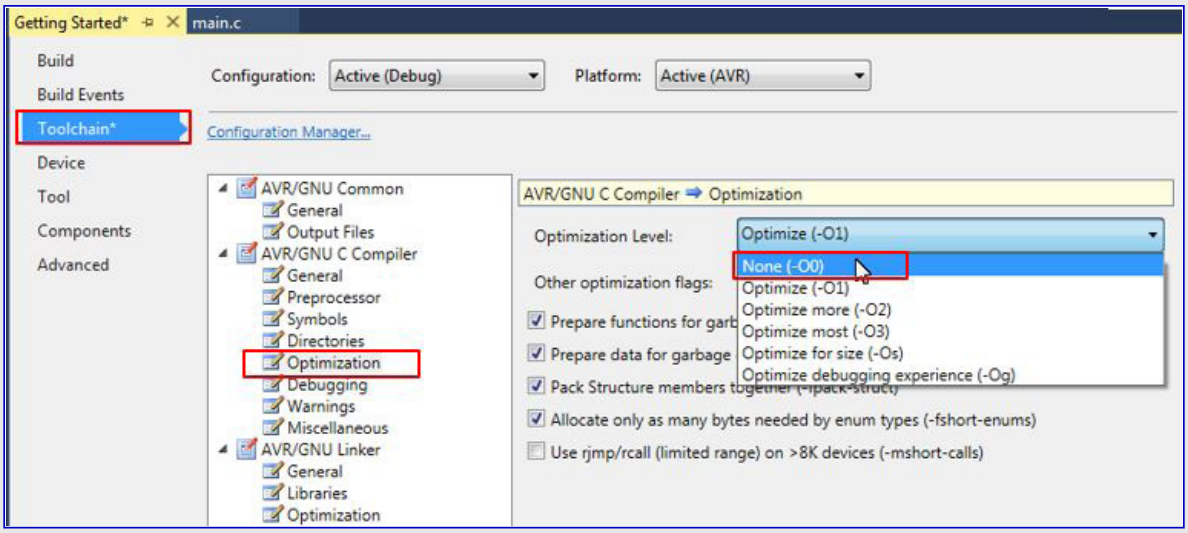
注: 呼び出しｽﾀｯｸが最初に何故_vector_4から来ることを示すのかを知るため、**図1-47.**で示されるように、PORTB_PORT_vectをｸﾘｯｸして定義に対する前後関係領域を覗いて見てください。

図1-47. _vector_4はPORTB割り込みﾍﾞｸﾀ



4. Stop Debugging(ﾃﾞﾊﾞｯｸﾞ停止) ■ 釦をｸﾘｯｸ、またはShift+F5を押すことによってﾃﾞﾊﾞｯｸﾞを停止してください。
5. Project(ﾌﾟﾛｼﾞｪｸﾄ)⇒<ﾌﾟﾛｼﾞｪｸﾄ名> properties(ﾌﾟﾛﾊﾟﾃｨ)へ行くか、またはAlt+F7を押すことによってﾌﾟﾛｼﾞｪｸﾄ設定を開いてください。**図1-48.**でのように、左ﾒﾆｭｰでToolchain(ﾂｰﾙﾁｪｰﾝ)ﾀﾌﾞへ行ってください。
6. AVR/GNU C Compiler(ｺﾝﾊﾟｲﾗ)⇒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。

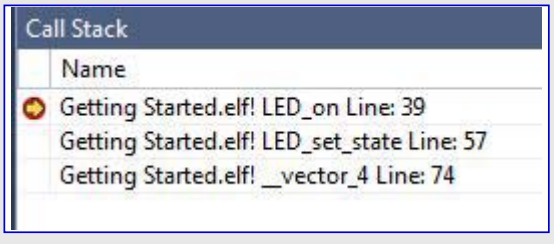
図1-48. ｺﾝﾊﾟｲﾗ最適化禁止



警告 ｺﾝﾊﾟｲﾗ最適化禁止はﾒﾓﾘ消費増加に帰着し、実行ﾀｲﾐﾝｸﾞでの変化に帰着し得ます。これはﾃﾞﾊﾞｯｸﾞ時間が重要なｺｰﾄﾞの時に考慮されることが重要で有り得ます。

7. 新しいﾃﾞﾊﾞｯｸﾞ作業を開始してLED_on()内でｺｰﾄﾞ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1-49.**で示されるように、ｺｰﾄﾞ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()を一覧にするでしょう。

図1-49. 最適化なしでの呼び出しｽﾀｯｸ



助言: Atmel Studioはｺﾝﾊﾟｲﾙされたｺｰﾄﾞをできるだけｿｰｽ ｺｰﾄﾞに繋げようとしますが、ｺﾝﾊﾟｲﾗの最適化はこれを困難にします。ｺﾝﾊﾟｲﾗの最適化禁止は、中断点がﾃﾞﾊﾞｯｸﾞ中に無視されたように思えたり、実行の流れがｺｰﾄﾞ段階実行中に従うのが難しい場合に役立ち得ます。

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。

ﾃﾞﾊﾞｯｸﾞ1に使われるｺｰﾄﾞ

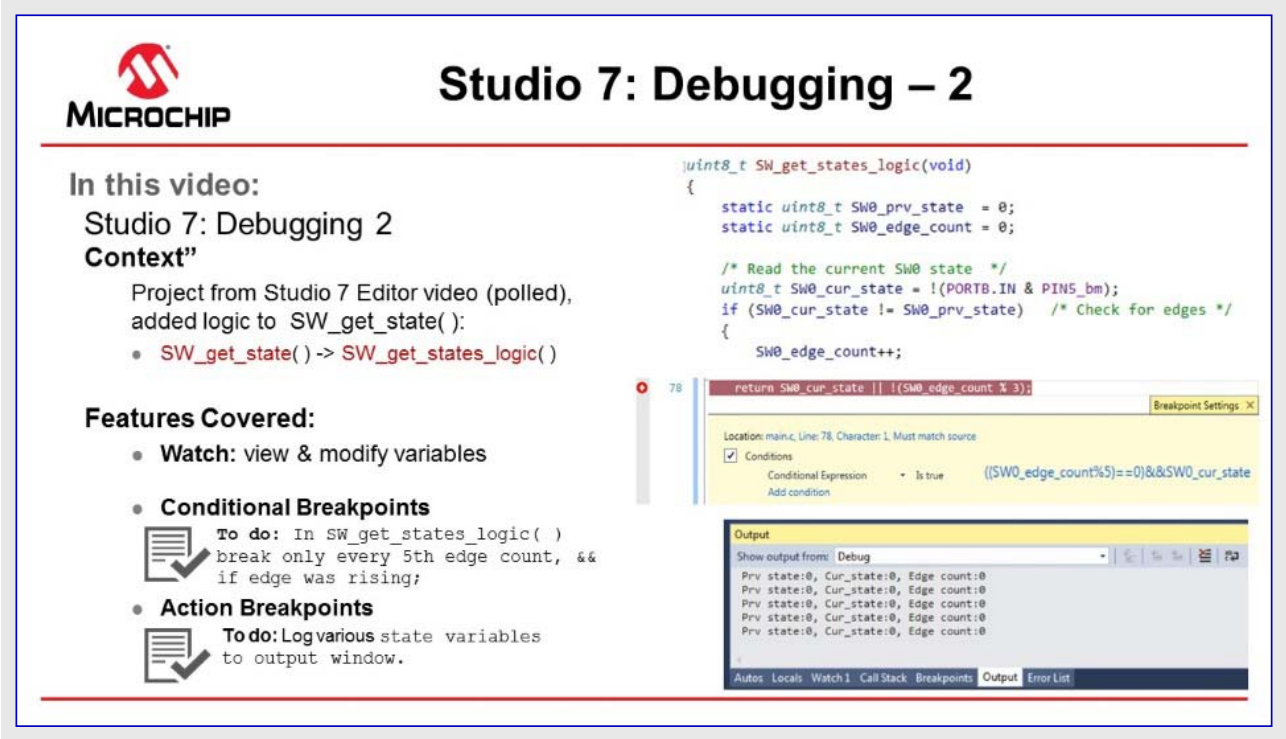
```
/*
 LEDはｽｲｯﾁが押された時にONにされ、LEDは(ﾋﾟﾝ変化割り込み経由で)ONにされます。予期せぬﾌﾟﾛｼﾞｪｸﾄ中断を避けるため、
 目的対象へ取り付け実演するために書かれたMY_mistake()が注釈にされます。

 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は外部ﾌﾟﾙｱｯﾌﾟ抵抗を持ちません。
 SW0はATtiny817の内部ﾌﾟﾙｱｯﾌﾟ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 LED_functions

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);
}
```

1.14. ﾃﾞﾊﾞｯｸﾞ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Atmel Studio 7でのもっと高度なﾃﾞﾊﾞｯｸﾞの話題を網羅します。主な話題はｺｰﾄﾞ内の変数の変更方法、条件付きと活動付きの中断点(ﾌﾞﾚｰｸ ﾎﾟｲﾝﾄ)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



映像: ﾃﾞﾊﾞｯｸﾞ-2

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

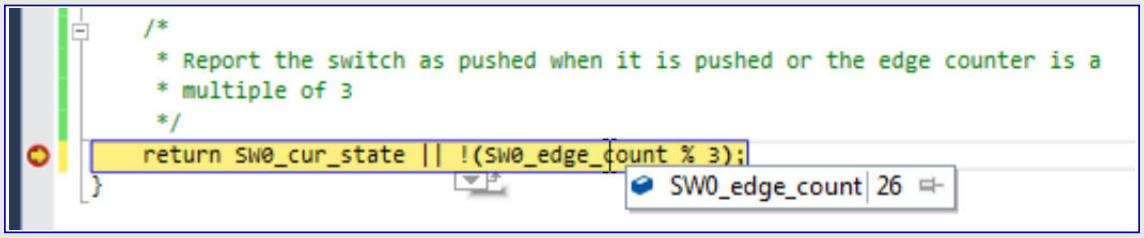
1. 使われるｺｰﾄﾞ(下参照)は「ｴﾃﾞｨﾀ: ｺｰﾄﾞの記述と整理 (Visual Assist)」項で開発されたものと同じです。SW_get_state()関数が以下のｺｰﾄﾞで置き換えられているだけです(戻り値の型が変わっていることにも注意してください)。

```
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: このｺｰﾄﾞは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(ﾌﾞﾚｰｸ ﾎﾟｲﾝﾄ)を禁止するためにDebug(ﾃﾞﾊﾞｯｸﾞ)⇒Disable All Breakpoints(全中断点禁止)へ行ってください。これはBreakpoints(中断点)ｳｨﾝﾄﾞ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(ﾃﾞﾊﾞｯｸﾞ開始) ▶ 釦をｸﾘｯｸすることによって新しいﾃﾞﾊﾞｯｸﾞ作業を開始してください。
4. ｷｯﾄ上のSW0を数回押して、ｺｰﾄﾞへの変更が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1-50.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。

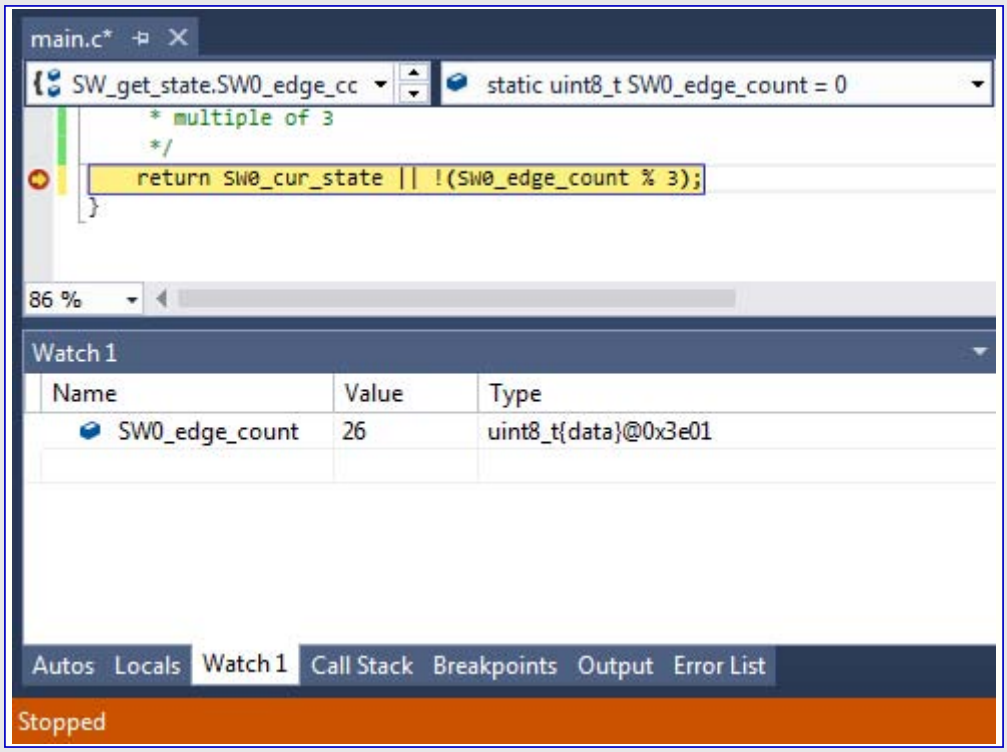
図1-50. 現在値を見るために変数上にｶｰｿﾙをかざす。



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、Atmel Studioはﾎﾟｯﾌﾟｱｯﾌﾟでその変数の内容を提示します。

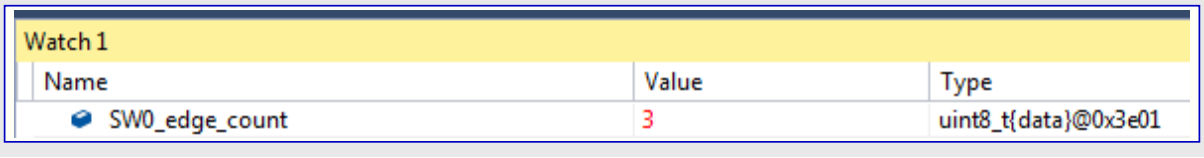
7. 変数をﾃﾞｰﾀ監視ｳｨﾝﾄﾞｳに追加するために、SW0_edge_count変数を右ｸﾘｯｸして脈絡ﾒﾆｭｰからAdd Watch(監視に追加)を選んでください。**図1-51.**でのように、変数値、ﾃﾞｰﾀ型、ﾒﾓﾘ ｱﾄﾞﾚｽと共に一覧にされたSW0_edge_count変数を持つWatch(監視)ｳｨﾝﾄﾞｳが現れるでしょう。

図1-51. Watch(監視)ｳｨﾝﾄﾞｳへ変数追加



8. 下で記述される手順を使って、Watch(監視)ｳｨﾝﾄﾞ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1-52.**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)ｳｨﾝﾄﾞｳで変数値をﾀﾞﾌﾞﾙ 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1-52. Watch(監視)ｳｨﾝﾄﾞｳで新しく更新された変数値



情報: Watch(監視)ｳｨﾝﾄﾞｳのValue(値)列は監視ｳｨﾝﾄﾞｳで右ｸﾘｯｸして脈絡ﾒﾆｭｰからHexadecial 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. ﾃﾞﾊﾞｲｽに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ﾃﾞﾊﾞｯｸﾞ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)ｳｨﾝﾄﾞ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ﾄﾞｳでのより良い可読性のために、変数を違うﾃﾞｰﾀ型にｷｬｽﾄ(型変換)することさえ可能です。これは特にﾎﾟ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、それはﾎﾟｲﾝﾀとして関数に渡されます。これはAtmel 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、監視ｳｨﾝﾄﾞ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ﾎﾟ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。

これは監視ｳｨﾝﾄﾞｳで空の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
```

変数へのﾎﾟｲﾝﾀを得るためにこの場合は'&'ｼﾝﾎﾞ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: Atmel Studioは今やｺｰﾄﾞ内の変数の内容を調べて変更するのに使われています。

1.14.1. 条件付き中断点

本項は条件付き中断点を配置するのにAtmel 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ﾄﾞ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ﾄﾞ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端計数毎にﾃﾞﾊﾞｯｸﾞ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)ｳｨﾝﾄﾞｳを使ってﾌﾟﾛｼﾞｪｸﾄから全ての中断点を解消してください。
2. 図1-53.でのように、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1-53. 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. 釦をｸﾘｯｸするか、またはF5を押すことによってﾃﾞﾊﾞｯｸﾞを続けるか、新しいﾃﾞﾊﾞｯｸﾞ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ﾄﾞ実行がどう停止されるかを観察してください。
8. Watch(監視)ｳｨﾝﾄﾞ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ﾄﾞ実行が完全に停止されるとは言え、Atmel 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に的中する毎にｺｰﾄﾞ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、実行ﾀｲﾐﾝｸﾞ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: Atmel 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。

1.14.2. 活動付き中断点

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

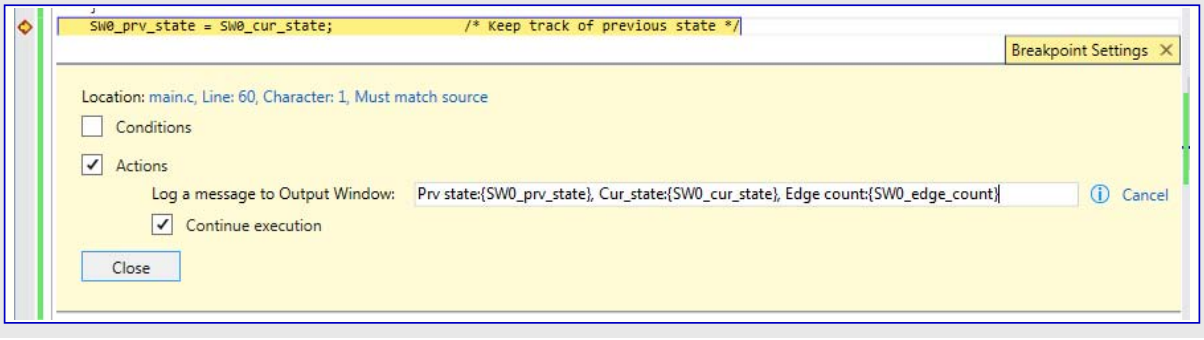
活動付き中断点はｺｰﾄﾞ実行を停止して手動で必要とするﾃﾞｰﾀを記録することなしに変数内容や実行の流れが記録されることを必要な場合に有用で有り得ます。

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. 進行中のﾃﾞﾊﾞｯｸﾞ作業を停止してBreakpoints(中断点)ｳｨﾝﾄﾞｳから全ての中断点を解消ください。
2. 図1-54.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 messege to Output Window(出力ｳｨﾝﾄﾞｳにﾒｯｾｰｼﾞ記録)ﾃｷｽﾄ枠に以下を入力してください。

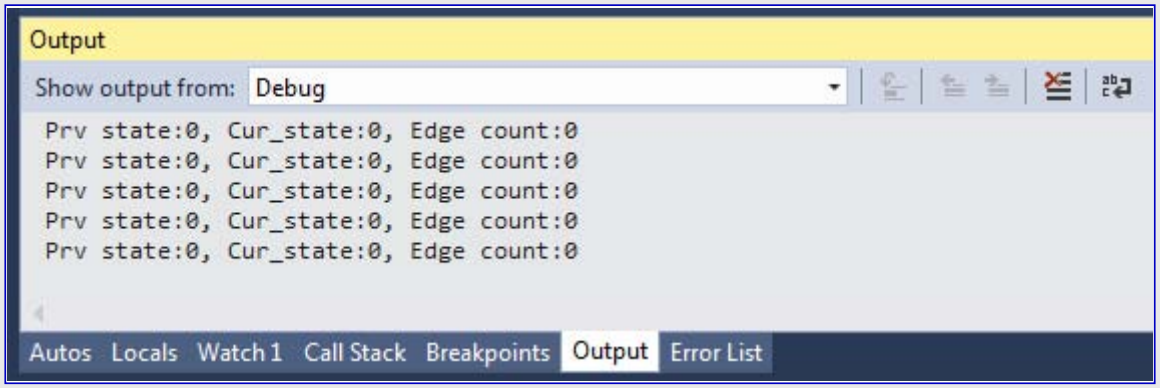
```
Prv state:{SW0_prv_state}, Cur_state:{SW0_cur_state}, Edge count:{SW0_edge_count}
```

図1-54. 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. ﾃﾞﾊﾞｯｸﾞ作業を開始してください。
7. Debug(ﾃﾞﾊﾞｯｸﾞ)⇒Windows(ｳｨﾝﾄﾞｳ)⇒Output(出力)へ行くことによってﾃﾞﾊﾞｯｸﾞ出力ｳｨﾝﾄﾞｳを開いてください。図1-55.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1-55. 変数内容を示すﾃﾞﾊﾞｯｸﾞ出力ｳｨﾝﾄﾞｳ



警告 活動付き中断点使用時、Atmel Studioは変数内容を読み出すためにｺｰﾄﾞ実行を一時的に停止します。結果として、実行ﾀｲﾐﾝｸﾞ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、ｺｰﾄﾞの1行によって遅らされるﾃﾞﾊﾞｯｸﾞ ｳｨﾝﾄﾞ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ﾃﾞ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: Atmel Studioは活動付き中断点を使って変数ﾃﾞｰﾀを記録するのに使われています。

1.14.3. (ATtiny817 Xplained Pro用) 使用ｺｰﾄﾞ

条件付きと活動付きの中断点に使われるｺｰﾄﾞ

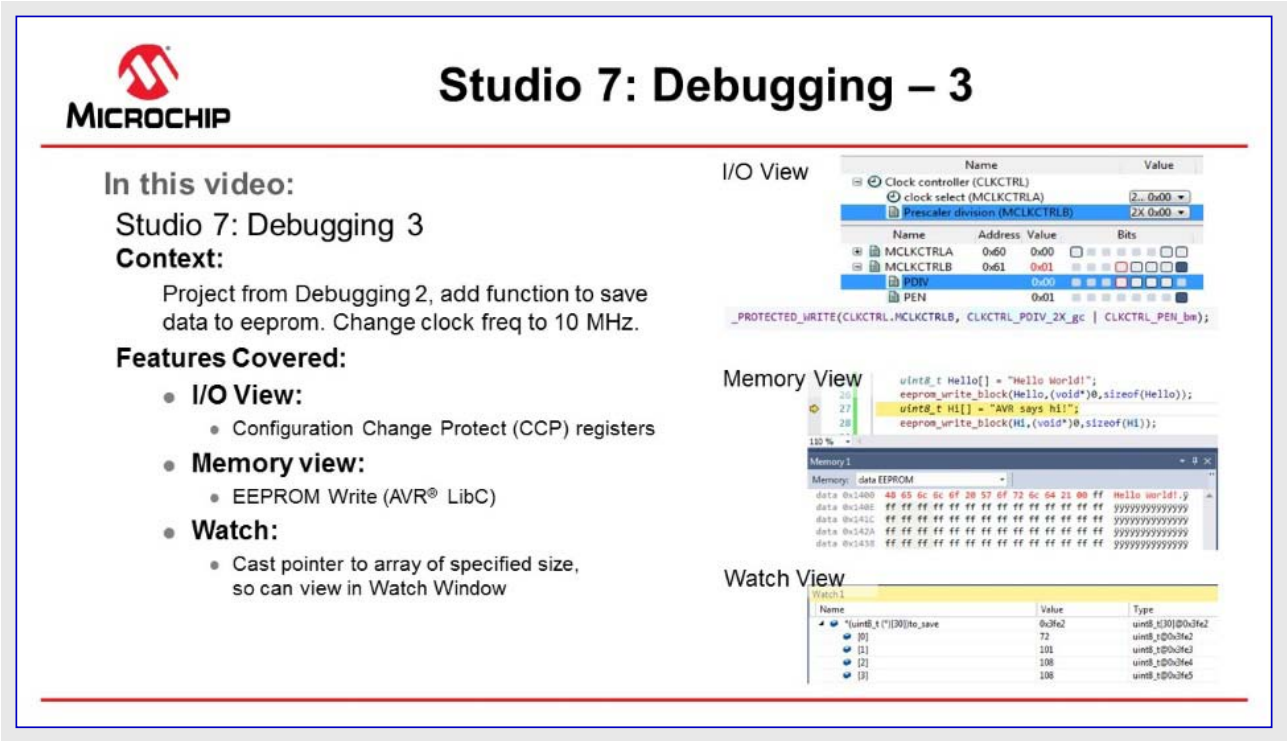
```
#include <avr/io.h>
#include <avr/interrupt.h>

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);
}
```

1.15. ﾃﾞﾊﾞｯｸﾞ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Atmel Studio 7でのもっと高度なﾃﾞﾊﾞｯｸﾞ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)ﾚｼﾞ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、ﾎﾟｲﾝﾀを配列としてｷｬｽﾄ(型変換)するためのWatch(監視)ｳｨﾝﾄﾞｳの使い方もです。

開始に際しての話題



映像: ﾃﾞﾊﾞｯｸﾞ-3

1.15.1. I/O表示部

I/O表示部はﾌﾟﾛｼﾞｪｸﾄと関連するﾃﾞﾊﾞｲｽのI/Oﾒﾓﾘ配置の図画的表示を提供します。このﾃﾞﾊﾞｯｸﾞ ツールはﾃﾞﾊﾞｯｸﾞ時に実際のﾚｼﾞｽﾀ内容を表示し、周辺機能構成設定の確認を許します。これは再ｺﾝﾊﾟｲﾙする必要なしにﾚｼﾞ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。

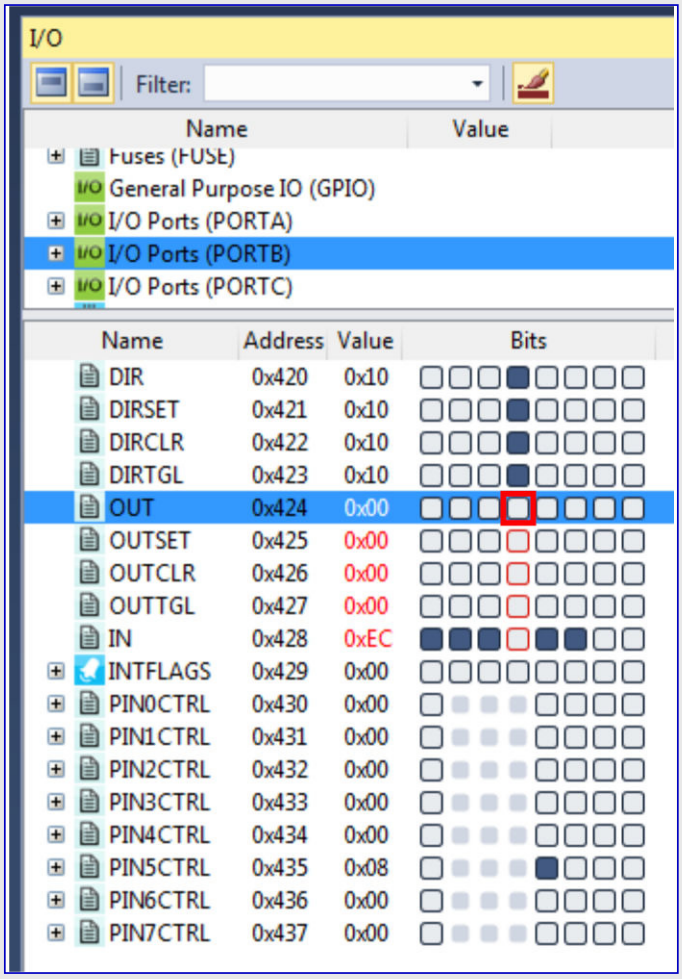
- ﾃﾞﾊﾞｲｽのﾒﾓﾘ配置の概要を取得
- 現在の周辺機能構成設定を調査
- 周辺機能構成設定を変更
- 構成設定変更を確認

1. 全ての中断点を取り去って新しいﾃﾞﾊﾞｯｸﾞ作業を開始してください。
2. Break All(全て中断 ⏸ 釦を押すことによってｺｰﾄﾞ実行を中断してください。
3. 上部ﾒﾆｭｰ ﾊﾞｰからDebug(ﾃﾞﾊﾞｯｸﾞ)⇒Windows(ｳｨﾝﾄﾞｳ)⇒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1-56.**で描かれるように、OUTﾚｼﾞｽﾀを見つけて対応する四角が色を変えるようにBits(ﾋﾞｯﾄ)列でﾋﾞｯﾄ4をｸﾘｯｸしてください。PORTB.OUTﾚｼﾞｽﾀでﾋﾞｯﾄ4をｸﾘｯｸすることはGPIOのPB4ﾋﾟﾝでの出力ﾚﾍﾞﾙを切り替え、これはATtiny817 Xplained Pro上のLEDを制御します。

情報: I/O表示部は何れかのﾚｼﾞ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)領域をﾀﾞﾌﾞﾙ ｸﾘｯｸしてﾚｼﾞ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ﾋﾞｯﾄを同時に変更することができます。

図1-56. I/O表示部を使うﾚｼﾞｽﾀ内のﾋﾞｯﾄ値操作



5. I/Oｳｨﾝﾄﾞ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。

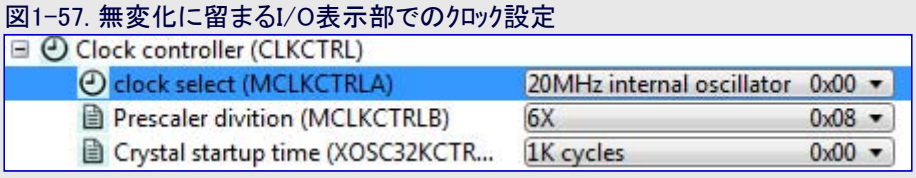
情報: 既定ｸﾛｯｸ構成設定はﾃﾞﾊﾞｲｽが1.8～5.5Vの支援される動作電圧範囲全体に渡ってｺｰﾄﾞを確実に実行することを保証します。Xplained ProｷｯﾄはATtiny817に3.3Vで給電します。ﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄの「全般動作定格」に従って、ﾃﾞﾊﾞｲｽは3.3V供給、10MHzで安全に走行することができます。

6. ｺｰﾄﾞは今や10MHzでATtiny817を走らせるように変更されています。下のように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. ﾌﾟﾛｼﾞｪｸﾄを再ｺﾝﾊﾟｲﾙしてﾃﾞﾊﾞｲｽに書き込むために新しいﾃﾞﾊﾞｯｸﾞ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ﾄﾞ実行を停止してください。**図1-57.**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。

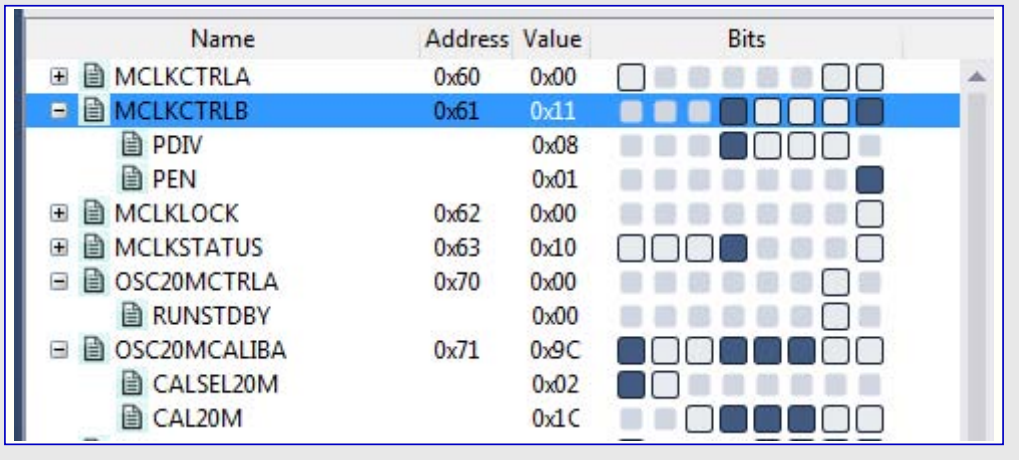
図1-57. 無変化に留まるI/O表示部でのｸﾛｯｸ設定



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1-58.で示されるように、I/O表示部でMCLKCTRLBﾚｼﾞｽﾀを選択してください。

図1-58. I/O表示部でのMCLKCTRLB選択



10. ｳｪﾌﾞに基づくﾚｼﾞｽﾀ説明を提示するためにｷｰﾎﾞｰﾄﾞでF1を押してください。

情報: ｳｪﾌﾞに基づくﾚｼﾞ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 ﾃﾞｰﾀｼｰﾄのｵﾌﾗｲﾝ版を参照してください。

11. MCLKCTRLBﾚｼﾞ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: このﾚｼﾞ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。重要なﾚｼﾞ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。これらのﾚｼﾞｽﾀはﾃﾞ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ﾄﾞ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);
```

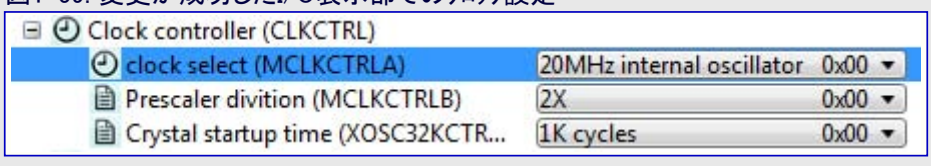
情報: _PROTECTED_WRITE()は保護されたﾚｼﾞｽﾀを解錠するためのﾀｲﾐﾝｸﾞ要件が合致することを保証するｱｾﾝﾌﾞﾘ ﾏｸﾛです。保護されたﾚｼﾞｽﾀを変更する時にこのﾏｸﾛを使うことが推奨されます。

助言: ﾏｸﾛの実装へ誘導するにはｺｰﾄﾞ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。これはｺｰﾄﾞのﾏｸﾛ名にｶｰｿﾙを置いてｷｰﾎﾞｰﾄﾞで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. 変更と共にﾃﾞﾊﾞｲｽを書き込むために以前のﾃﾞﾊﾞｯｸﾞ作業を停止して新しい(ﾃﾞﾊﾞｯｸﾞ)作業を開始してください。

14. 図1-59.で示されるように、ｺｰﾄﾞ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。

図1-59. 変更が成功したI/O表示部でのｸﾛｯｸ設定



助言: Processor Status(ﾌﾟﾛｾｯｻ状態)ｳｨﾝﾄﾞｳはAVRｺｱに対するﾚｼﾞｽﾀ表示ﾂｰﾙです。このﾂｰﾙは上部ﾒﾆｭｰ ﾊﾞｰからDebug(ﾃﾞﾊﾞｯｸﾞ)⇒Windows(ｳｨﾝﾄﾞｳ)⇒Processor Status(ﾌﾟﾛｾｯｻ状態)へ行くことによって開くことができます。このｳｨﾝﾄﾞｳは内部AVRｺｱ ﾚｼﾞ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(ｽﾃｰﾀｽﾚｼﾞｽﾀのIﾋﾞｯﾄを見る)のに使うことができます。

結果: I/O表示部の能力はﾌﾟﾛｼﾞｪｸﾄのﾊﾞｸﾞを見つけて修正するのに使われています。

1.15.2. メモリ表示部

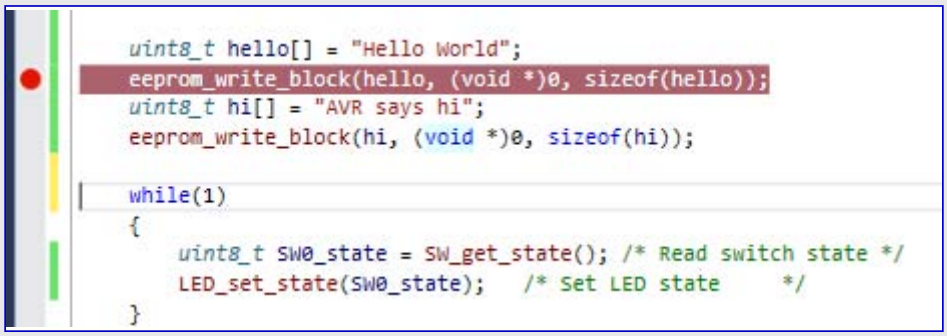
すべきこと: ATtiny817のEEPROM先頭に2つの文字列を書き、EEPROM内容を確認するためにメモリ表示部を使ってください。

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));
```

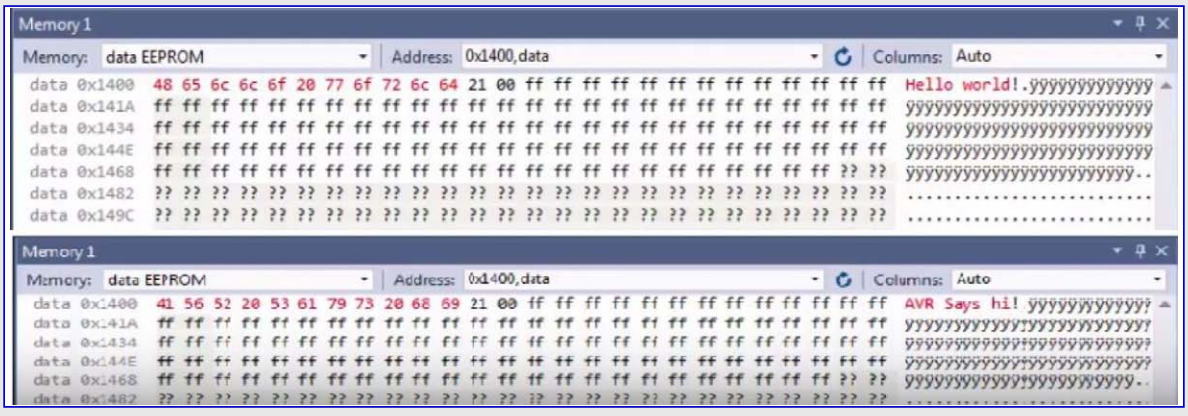
3. 図1-60.でのように最初のeeprom_write_block()呼び出しの隣に中断点を配置してください。

図1-60. EEPROM調査用停止に対する中断点



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(メモリ)⇒Memory 1(メモリ1)へ行くことによりT、メモリ 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. メモリ表示部を使って(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1-61.でのように現れるべきです。

図1-61. EEPROM書き込み後のメモリ表示部更新



助言: メモリ表示部ツールはプログラム メモリを含む他の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書き込み後のそれの内容がメモリ表示部を使って確認されました。

1.15.3. 監視ｳｨﾝﾄﾞｳ

これは「ﾃﾞﾊﾞｯｸﾞ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)ｳｨﾝﾄﾞｳでﾎﾟ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)ｳｨﾝﾄﾞ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ﾄﾞｳでのより良い可読性のために、変数を違うﾃﾞｰﾀ型にｷｬｽﾄ(型変換)することさえ可能です。これは特にﾎﾟ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、それはﾎﾟｲﾝﾀとして関数に渡されます。これはAtmel 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、監視ｳｨﾝﾄﾞ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ﾎﾟ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。

これは監視ｳｨﾝﾄﾞｳで空の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
```

変数へのﾎﾟｲﾝﾀを得るためにこの場合は'&'ｼﾝﾎﾞ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: Atmel Studioは今やｺｰﾄﾞ内の変数の内容を調べて変更するのに使われています。

ﾃﾞﾊﾞｯｸﾞ3に使うｺｰﾄﾞ

```
#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ﾋﾟﾝを出力として構成設定 */
    PORTB.PIN5CTRL = PORT_PULLUPEN_bm;                  /* SW0ﾋﾟﾝに対してﾌﾟﾙｱｯﾌﾟ許可 */

    _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();
    }
}
```

2. 改訂履歴

資料改訂	日付	注釈
A	2018年1月	初版資料公開

Microchipウェブ サイト

Microchipはhttp://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手段として使われます。お気に入りのインターネット ブラウザを用いてアクセスすることができ、ウェブ サイトは以下の情報を含みます。

- **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
- **全般的な技術支援** – 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相談役プログラム員一覧
- **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

お客様への変更通知サービス

Microchipのお客様通知サービスは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http://www.microchip.com/でMicrochipのウェブ サイトをアクセスしてください。”Support”下で”Customer Change Notification”をクリックし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
- 現場応用技術者(FAE:Field Aplication 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、または現場応用技術者(FAE)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http://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デバイス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、その製品系統が今日の市場でその種類の最も安全な系統の1つであると考えます。
- コード保護機能を破るのに使われる不正でおそらく違法な方法があります。当社の知る限りこれらの方法の全てはMicrochipのデータシートに含まれた動作仕様外の方法でMicrochip製品を使うことが必要です。おそらく、それを行う人は知的財産の窃盗に関与しています。
- Microchipはそれらのコードの完全性について心配されているお客様と共に働きたいと思います。
- Microchipや他のどの半導体製造業者もそれらのコードの安全を保証することはできません。コード保護は当社が製品を”破ることができない”として保証すると言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。Microchipのコード保護機能を破る試みはデジタル ミレニアム著作権法に違反するかもしれません。そのような行為があなたのソフトウェアや他の著作物に不正なアクセスを許す場合、その法律下の救済のために訴権を持つかもしれません。

法的通知

デバイス応用などに関してこの刊行物に含まれ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。**Microchipはその条件、品質、性能、商品性、目的適合性を含め、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。**Microchipはこの情報とそれの使用から生じる全責任を否認します。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

DNVによって認証された品質管理ｼｽﾃﾑ

ISO/TS 16949

Microchipはその世界的な本社、ｱﾘｿﾞﾅ州のﾁｬﾝﾄﾞﾗｰとﾃﾝﾍﾟ、ｵﾚｺﾞﾝ州ｸﾞﾗｼｬﾑの設計とｳｪﾊｰ製造設備とｶﾘﾌｫﾙﾆｱとｲﾝﾄﾞの設計ｾﾝﾀｰに対してISO/TS-16949:2009認証を取得しました。当社の品質ｼｽﾃﾑの処理と手続きはPIC® MCUとdsPIC® DSC、KEELOQ符号飛び回りﾃﾞﾊﾞｲｽ、直列EEPROM、ﾏｲｸﾛ周辺機能、不揮発性ﾒﾓﾘ、ｱﾅﾛｸﾞ製品用です。加えて、開発ｼｽﾃﾑの設計と製造のためのMicrochipの品質ｼｽﾃﾑはISO 9001:2000認証取得です。

日本語© *HERO* 2020.

本使用者の手引きはMicrochipのAtmel Studio 7とでの開始に際して(DS50002712A-2018年1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訳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: http://www.microchip.com/support ｳｪﾌﾞｱﾄﾞﾚｽ: www.microchip.com **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 **ﾎﾞｽﾄﾝ** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **ｼｶｺﾞ** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **ﾀﾞﾗｽ** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **ﾃﾞﾄﾛｲﾄ** Novi, MI Tel: 248-848-4000 **ﾋｭｰｽﾄﾝ** TX Tel: 281-894-5983 **ｲﾝﾃﾞｱﾅﾎﾟﾘｽ** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **ﾛｻﾝｾﾞﾙｽ** 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 **ｶﾅﾀﾞ - ﾄﾛﾝﾄ** Tel: 905-695-1980 Fax: 905-695-2078	**ｵｰｽﾄﾗﾘｱ - ｼﾄﾞﾆｰ** 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 - 香港特別行政区** 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	**ｲﾝﾄﾞ - ﾊﾝｶﾞﾛｰﾙ** Tel: 91-80-3090-4444 **ｲﾝﾄﾞ - ﾆｭｰﾃﾞﾘｰ** Tel: 91-11-4160-8631 **ｲﾝﾄﾞ - ﾌﾟﾈｰ** 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 - ｸｱﾗﾙﾝﾌﾟｰﾙ** Tel: 60-3-7651-7906 **ﾏﾚｰｼｱ - ﾍﾟﾅﾝ** Tel: 60-4-227-8870 **ﾌｨﾘﾋﾟﾝ - ﾏﾆﾗ** Tel: 63-2-634-9065 **ｼﾝｶﾞﾎﾟｰﾙ** 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 - ﾊﾞﾝｺｸ** Tel: 66-2-694-1351 **ﾍﾞﾄﾅﾑ - ﾎｰﾁﾐﾝ** Tel: 84-28-5448-2100	**ｵｰｽﾄﾘｱ - ｳﾞｪﾙｽ** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **ﾃﾞﾝﾏｰｸ - ｺﾍﾟﾝﾊｰｹﾞﾝ** Tel: 45-4450-2828 Fax: 45-4485-2829 **ﾌｨﾝﾗﾝﾄﾞ - ｴｽﾎﾟｰ** Tel: 358-9-4520-820 **ﾌﾗﾝｽ - ﾊﾟﾘ** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **ﾄﾞｲﾂ - ｶﾞﾙﾋﾝｸﾞ** Tel: 49-8931-9700 **ﾄﾞｲﾂ - ﾊｰﾝ** Tel: 49-2129-3766400 **ﾄﾞｲﾂ - ﾊｲﾙﾌﾞﾛﾝ** Tel: 49-7131-67-3636 **ﾄﾞｲﾂ - ｶｰﾙｽﾙｰｴ** Tel: 49-721-625370 **ﾄﾞｲﾂ - ﾐｭﾝﾍﾝ** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **ﾄﾞｲﾂ - ﾛｰｾﾞﾝﾊｲﾑ** 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 - ﾊﾟﾄﾞﾊﾞ** Tel: 39-049-7625286 **ｵﾗﾝﾀﾞ - ﾃﾞﾙｰﾈﾝ** Tel: 31-416-690399 Fax: 31-416-690340 **ﾉﾙｳｪｰ - ﾄﾛﾝﾊｲﾑ** Tel: 47-7289-7561 **ﾎﾟｰﾗﾝﾄﾞ - ﾜﾙｼｬﾜ** Tel: 48-22-3325737 **ﾙｰﾏﾆｱ - ﾌﾞｶﾚｽﾄ** Tel: 40-21-407-87-50 **ｽﾍﾟｲﾝ - ﾏﾄﾞﾘｰﾄﾞ** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **ｽｳｪｰﾃﾞﾝ - ｲｪｰﾃﾎﾞﾘ** Tel: 46-31-704-60-40 **ｽｳｪｰﾃﾞﾝ - ｽﾄｯｸﾎﾙﾑ** Tel: 46-8-5090-4654 **ｲｷﾞﾘｽ - ｳｫｰｷﾝｶﾞﾑ** Tel: 44-118-921-5800 Fax: 44-118-921-5820