



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	こ際して・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	• 3
	1.1.	AVR <sup>®</sup> とSAMの開発ツール概要・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	• 4
	1.2.	$AVR^{\mathbb{B}} \ge SAM \mathcal{O} n - h \dot{D}_{I} \mathcal{P} \mathcal{V} - \mu \ge f \dot{n} \mathcal{V} \mathcal{D}_{I}$	• 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.	ェディタ: 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.	〒゙ベッグ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	夏歴 • • • • • • • • • • • • • • • • • • •	59
Mic	rochip	<u>ሳェブ サイト ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・</u>	60
おる	客様へ	の変更通知サービス・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	60
おる	<b>}様支</b>	援 ••••••••••••••••••••••••••••••••••••	60
Mic	rochip	デバイス コード保護機能 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	60
法的	小涌知		60
商机			61
יואם	~ Vによ-	って認証された品質管理システム・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	61
₩ ₩	見的たい	ッ こ 記 皿 こ 1 い こ 田 貞 日 22 / / / ム 販 吉 レサード 7	62
120	LH1.41		04

# 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

練習の多くはエデ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。 ハードウェア要件:

- ATtiny817 Xplained Pro
- ・標準A-マイクロB USBケーフル

ソフトウェア要件:

- $\cdot$  Atmel Studio 7.0
- avr-gccツールチェーン
- ・tineyAVR®デベイス用最新部品ペック
- 使うAtmel Studio 7.0プラグイン
- ・Atmwl 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。



本項はAVR<sup>®</sup>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

In this video:						
Context in Microchip Tools Ecosystem	8-Bit PIC® MCU	16-Bit PIC MCU and dsPIC®	32 Bit PIC MCU	AVR® MCU	SAM N	
<ul> <li>IDE, Compiler, MCU &amp; SW configurator tools, Firmware Libraries</li> </ul>		MPLAB <sup>®</sup> x IDE MPLAB Xpress IDE (Cloud-Based)			Atmei Studio	
<ul> <li>START, Software Content and IDEs</li> <li>How these pieces fit together.</li> <li>START-based development</li> </ul>	MPLAB XC C Compilers			AVR GCC C Compilers	ARM GC Compil	
<ul> <li>START user manual</li> <li>Getting Started projects in START</li> </ul>	M	IPLAB Code Configurat	tor	Atme	I START	
Atmel Studio 7	Microchip Applicati	Libraries for ons (MLA)	MPLAB Harmony	Advance	d Software nework	
<ul> <li>Bare-metal- vs. START-based development</li> <li>Build from scratch (bare-metal):         <ul> <li>Getting Started Atmel Studio 7</li> <li>Getting Started with AVR Tools</li> </ul> </li> </ul>	MPLA	B XC PRO C Compiler	Licenses	LAR Workbench	IAR Workbe Keil M	

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、生成されたコート、プロジェクトをダウンロート、してAtmel Studio 7、IAR embedded Workbench<sup>®</sup>、Keil<sup>®</sup>、µVsion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。
- ・トライバ、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。



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 済みのドバイバとコード単位部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。Cはメモリ量、コード性能を改善するだけでなく、Atmel STARTウェブ使用者 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ウィザードを置き換えます。

AVR製品部門を支援するFoundation Services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。Found ation 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 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デバックするための統合開発基盤(IDP:Integrated Development Platform)です。Atmel Studio 7 IDPはC/C++またはアセンブリ言語のコート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のデバイスを支援するデバッガ、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 Studio 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。

# 1.2. AVR<sup>®</sup>とSAMのハート ウェア ツールとテ ハ ッカ

本項はAVR<sup>®</sup>とSAM MCU用のハートウェア 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・ウェア ツール & デ・バッカ

#### 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ガ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、コート、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、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プログラミングとデバッグ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デバッガ

PowerデバッカゴはUPDI、JTAG、PDI、デバック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デバッガ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デバッガはCDC仮想COMホートだけでなく、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みます。

Powerデバッガは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互換デバッガです。Powerデバッガ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。

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゛サイト: Powerテ゛ハ゛ッカ゛

# 1.3. データ可視器とPowerデバッガ実演

本項はPowerデバッガ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



#### 映像: データ可視器とPowerデバッガ実演



```
ADCO.CTRLC = ADC_PRESC_DIV8_gc | ADC_REFSEL_VDDREF_gc;
    ADCO. CTRLA = ADC_ENABLE_bm | ADC_RESSEL_8BIT_gc;
    ADCO.MUXPOS = ADC_MUXPOS_AIN6_gc;
                                      // picoPower 1: 故に休止で走行可
    ADCO. CTRLA = ADC_RUNSTBY_bm;
    ADCO.CTRLE = ADC_WINCM_OUTSIDE_gc;
                                        // picoPower 3: 故に自身の採取を評価可
    ADCO. INTCTRL = ADC_WCMP_bm;
    ADCO. WINHT = 200;
    ADCO. WINLT = 100;
    ADCO. EVCTRL = ADC_STARTEI_bm;
                                  // picoPower 4: 故に事象は変換を起動可
}
uint8_t adc_get_result(void)
{
    return ADCO.RESL;
// picoPower 5: 素早く送出、その後休止に戻る: compare 9600,115200,1250000のボーレートを比較
// 1バイトだけ送出に注意
#define BAUD_RATE 57600
void usart_init()
{
    USARTO. CTRLB = USART_TXEN_bm;
    USARTO. 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。
    USARTO. STATUS = USART_TXCIF_bm;
    VPORTB. OUT |= PIN6_bm;
    USARTO. TXDATAL = c;
    while(!(USARTO.STATUS & USART_TXCIF_bm));
    VPORTB. OUT &= ~PIN6_bm;
    VPORTB. DIR &= ~PIN2_bm | PIN6_bm; // picoPower 2b: 送信間でTxピン禁止
}
// 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++)</pre>
    {
          (&PORTA. PINOCTRL) [pin] = PORT_ISC_INPUT_DISABLE_gc;
          (&PORTB. PINOCTRL) [pin] = PORT_ISC_INPUT_DISABLE_gc;
          (&PORTC. PINOCTRL) [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、スタンハイと比較
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

#### 1.4.1. 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 &Windows Server 2012 R2
- Windows 10

## 支援基本構造

- 32ビット(x86)
- 64ビット(x64)

#### ハート・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バ小RAM
- ・6Gベイトの利用可能なハードディスク空間

#### ダウンロート、とインストール

- ・最新のAtmel Studioインストーラーをダウンロードしてください。: Atmel Studion 7
- ウェブ インストーラは(<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トします。 - 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・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 ウィザート 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 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。

### 1.4.2. オフライン資料のダウンロート

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って)data sheets(データシート)、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デバッカ)使用者手引書)、ATtiny817 Xplained Prouser manual(ATtiny817 Xplained Pro使用者手引書)だけでなく、ATtiny817 Complete data sh eet(ATtiny817完全データシート)もダウンロート、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トを開始します。





Contents 9	Help Viewer Home Manage Content								
Filter Contents	Add and Remove Content Adding content will automatically refresh all local documentation with available updates								
ATmega328PB Data Visualizer mega328PB Xplained Mini (TEST] ATmega328P Datasheet - Prelim	Installation source:      Online O Disk: C:\Users\M43959\AppData\Local\M	Local store path: C:\ProgramData\Microsoft\He Move							
	power	×		Pending changes:					
	Name ATURIY417 / ATURIY0147 ATURIY0107 ATT ATtiny417 / ATURIY0147 Complete 4 User Guides	Action Aug Cancel	Status Add (pe	Add ATtiny817 Xplained Pro [X] ATtiny417 / ATtiny817 Complete [X] Power Debugger [X]					
	UC3L Evaluation Kit Programmers and Debuggers Power Debugger	Add Cancel	Add (pe						
	4		•	Estimated download size: 15 MI Free disk space: 158213 MI Required disk space: 75 MI					

# 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

In this video: How to add extensions	Atmel Galle	rv 😭	
Tools -> Gallery Profile	ELEVER Inthe Alexand Calibry, particip interact Calibry, provide the cases. Toomised and vectors, attent Status and records. Status of Status and records.	Studio Constantino de la const	Search the Gallery
Extensions:	Supported by Report, or 2004, 5 and 1000 for any light to consti- a 4	Anno Assa Anno Rogana Fighter	
<ul> <li>START, Data Visualizer, Toolchain</li> <li>Popular: Arduino<sup>®</sup> IDE for Studio 7, LUFA Library, ASF (Naggy)</li> <li>Used in series: Doxygen integrator, Git Source Control Provider,</li> <li>Extension options/settings</li> <li>Tools → Options</li> </ul>	RECENTLY ADOED	Barr 1 - Contract           Barr 1 - Contrant           Barr 1 - Contrant	HOLEST FALSE Fage 1. 1, 4, 63 Team of the second

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

In this video:	AVR42779 Ultrasonic Distance Measurement - AtmetStudio File Edit View VAssistX ASF Project Build Debug Tools Winds
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	New       Briget       Chi-Shin-N         Open       Brie       Chi-Shin-N         Add       File       Chi-Shin-N         Close       Solution       Briend Start Project       Solution         Core Solution       Briend Start Project       Solution       Solution         Solution       Briend Start Project       Solution       Solution         Solution       Briend Start Project       Chi-Shit+E         Solution       Briend Start Project       Chi-Shit+E         Solution       Briend Start Project       Solution         Solution       Briend Start Project       Solution         Output Files       Duponencies       Output Files         Output Files       Duponencies       Output Files         Output Files       Dubaries       Duponencies         Output Files       Duponencies       Solution         Dependencies       Duponencies       Solution Start Project         Project       Dubaries       Duponencies         Dependencies       Duponencies       Solution         Dependencies       Duponencies       Duponencies         District       Duponencies       Duponencies       Duponencies         District

映像: Atmel START統合

Y 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ェクスポート)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(一括ダウンロート)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トされます。

図1-1. 構成	設定したプロジェクトのダウンロード
	DOWNLOAD YOUR CONFIGURED PROJECT
	Download a generated pack containing all your configured software components.
	Select which IDE or command line tool you want the pack to include support files for:
	Atmel Studio:
	Ψ μVision from Keil:
	IAR Embedded Workbench:
	BRT Somnium DRT. (Atmel Studio plugin):
	Makefile (standalone):
	Specify file name (optional): My Project
	🛨 DOWNLOAD PACK

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

- 4. Atmel Studioを起動してください。
- 5. File(ファイル)⇒Import(インホ<sup>°</sup>ート)⇒Atmel START Project(Atmel STARTフ<sup>°</sup>ロシ<sup>\*</sup>ェクト)を選んでください。

图1-	2. Atmel START7 ወያ ፤ሳኮው ብንቱ	: - <b>h</b>						
*	AtmelStudio (Administrato	r)						
File	Edit View ASF Project	Debug	Tools	Wi	ndow	Help		
	New		•	2	- C" -	E Q		
	Open		•	ex X	80.	- 5		, <b>m</b> _ š :
	Close							
ж	Close Solution							
	Import		•		AVR32 St	tudio Projec	t	Ctrl+3
	Save Selected Items	Ctrl+S			AVR Stud	dio 4 Project	t	Ctrl+4
	Save Selected Items As				Atmel St	art Project		
<b>1</b>	Save All	Ctrl+Shift	+S		Project T	emplate		Ctrl+T
	Export Template				,	1		
Ð	Page Setup							
	Print	Ctrl+P						
	Recent Files							
	Recent Projects and Solutions		×					
×	Exit	Alt+F4						

- 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 インホート ダイアログ ボックスが開きます。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。

mel Start Importer			
mport Atmel Start	Project		
Atmel Start Project(.atzip):	C:\Users\m43934\Downloads\My817Pro.atzip		Browse
View project summary (CMS	Spackage information)		
Project Name:	My817Pro1		
Location:	C:\Users\m43934\Documents\Atmel Studio\7.0		Browse
Solution:	Create New Solution	Ψ.	
Solution Name:	My817Pro1		
fiew project import summar	<u>x</u>		

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

My817Pro0 - AtmetStudio         File       Edit       View       VAssistX       ASF       Project       Build       Debug       Tools       Window       Help         Image: Image	Debug Browser * 🔹 🗾 full J 🕅 🛫 🗰 ATtiny817 🏌 No Tool 🖕
<pre>driver_isr.c atmel_start.c * ×</pre>	Solution Explorer         Search Solution Explorer (Ctrl+ -)         Solution My817Pro0         Dependencies         Output Files         Dependencies         Output Files         Config         Config         Config         Config         Config         Config         Config         Include         strc         utils         c atme_start.c         h atme_start.h         c driver_isr.c         main.c

🖹 すべきこと: Atmel STARTの出力をAtmel 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からダウンロート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ドウで開きます。

溄 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)釦をクリックするか、またはSokution Explorer内のプ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. Atmel Studio内のウィントウで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ウィントウの下部でGENERATE PROJECT(プロジェクト生成) 釦をクリック してください。

# 1.7. 新規プロジェクト作成

本項は新しい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:新規プロジェクト作成

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

- 1. 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(プロジェクト)に行ってください。

	Edit View VAssistX ASI	Project D	ebug	Tools Window Help		
	New		• <b></b>	Project	Ctrl+Shift+N	- De
	Open		• *	File	Ctrl+N	- 8
	Close			Atmel Start Project		
З	Close Solution			Atmel Start Example Proje	ect	
	Import		• 🗄	Example Project	Ctrl+Shift+E	
iii	Save Selected Items	Ctrl+S	1			_
	Save Selected Items As		22	The		
٩	Save All	Ctrl+Shift+S				
	Export Template		E D			
B	Page Setup					
Ð,	Print	Ctrl+P	ned	Mini evaluation kit is a hard	ware	
	Recent Files		tmel	ATtiny817 microcontroller.	Supported	
	<b>Recent Projects and Solutions</b>		ated	development platform, the	kit 7 and	
	Exit	Alt+F4	he d	evice in a customer design.	/ allu	

# Atmel Studio 7

3. プロジェクト生成ウィサートが現れます。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1-6.をご覧ください。

ew Project					-8	
Recent	Sort by:	Default	• II' 🗉		Search Installed Templates (Ctrl+E)	p.
Installed		GCC C ASF Boar	d Project	C/C++	Туре: С/С++	1000
Assembler AtmelStudio Solu	ution	GCC C Executab	le Project	C/C++	Creates an AVK 8-bit or AVK/AKM 32 project	2-bit C
	đ	GCC C QTouch	Executable Project	C/C++		
		GCC C Static Lib	rary Project	C/C++		
		GCC C++ Execut	table Project	C/C++		1
		GCC C++ Static	Library Project	C/C++	sinclude carry	
	0	Create project fr	om Arduino sketch	C/C++	Int exin(void) f Printf("Hello" GCC	
	M. Exclusion					
yame:	MyrinstProject	ors\Documents\A	Atmel Studio\7.0		Browce	
olution name:	MyFirstProject	our pocuments (*			Create directory for solution	



**助言**:全てのAtmel Studioプロジェクトは解決策(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プロジェ 外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(解決策用にデルクトリを作成)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を作成します。

# プロジェクト形式について

衣 三. ノロ	リエクト形式	
区分	プロジェクト雛形	説明
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 静的ライフ゛ラリ プロシ゛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(コート、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 プロシ゛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(コート、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。

# Atmel Studio 7

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(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。

의 / 제 / (1)		17进代				
Device Selection						×
Device Family:	All	•		817		×
Name	App./Boot Memo	ory (Kbytes)Data Memory	y (bytes)EEPROM (bytes)	Device Info:		*
ATtiny817	8	512	128	Device Name: Speed: Vcc: Family: Datasheet (Summ Device Page Supported Tools Atmel-ICE X EDBG EDBG MSD DIAGICE3 Im mEDBG Power Debugge STK600	ATtiny817 N/A N/A ATtiny ary)	т. Т.
					<u>o</u> k <u>(</u>	ancel

**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ます。



# 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/デベッガに対する必要性を無くす基板上の組み込みデベッガ(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: 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						
ATtiny817 Xplained Pro - 0150	😐 🗙 Start Page						
MCU board	ATt: 017 Valained	2					
ATtiny817 Xplained Pro	Altiny817 Xplained Pro						
Extension							
	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.						
	S Atmel START example New Atmel START pro	projects using this board ject using this board					
	Launch Data Visualizer						
	External Links:						
	Technical Documentation	<u>on</u>					
	ATtiny817 Device Datash	neet					
	Xplained Pro Hardware [	Development Kit (HDK) User Guide					
	Kit Details						
	Serial number	ATML2654041800000150					
	Board name	ATtiny817 Xplained Pro					
	Manufacturer	Atmel					
	Target name	ATtiny817					
	Interfaces	SPI TWI GPIO CDC					
Show page on connect Update board database							

- 1.1. 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。
- 1.2. 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(デハイス プログラミンク)によってProgramming(プログラミンク))タイアログを開きます。
  - 2.1. EDBGツールを選んでDevice=ATtiny817を確実にし、その後にデバ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(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。
- 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
- 2.7. Lock bits(施錠ビット): 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デバイスを書き込み
- 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さえも持ちます。
- 2.10. 例えば、SAML21JデベイスはEEPROMを持ちません(フラッシュ メモリで模倣できます)。デベ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、デハイス名で選別することによってデハ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 ハーに配置されたTool(ツール)釦をクリックしてください。

図1-10. ツール釦				
Window Help				
🔍 🛛 🖬 🖬 🔊	Debug 🔹	Debug Browser 👻	~	
🦷 🖮 🖶 🐺 🖕	🏙 📥 🛛 🖕 🖗	🗰 ATtiny817 🦷	No Tool 💂	

5. Project Properties(プロシェクトプロハティ)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。

/FirstProject* 👳 🗙 A	Ttiny817 Xplained Pro - 0806	main.c				
Build Build Events	Configuration: N/A	-	Platform:	N/A	•	
Toolchain Device Tool Components Advanced	Selected debugger/programm EDBG ATML2654041800000806 Simulator P Custom Programming Tool Erase entire chip • P Preserve EEPROM	ner				
	Debug settings Keep timers running in sta Cache all flash memory ex	op mode kcept				

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。



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。

Available Tools		<b>т</b> џ х	
Tools and Simulators	Status		
EDBG (ATML265404:	1800000693)	Disconnected	
EDBG (ATML265404	18000007031	Connected	
Simulator	Add Target	onnected	
	Upgrade 🍃		
	Show Info Window		

## 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

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Atmel Studio 7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

In this video:		VO
Context:		Name Value * 2000 Analog to Digital Converte # 2000 Analog to Digital Converte * 2000 Voltage reference (VREF)
<ul> <li>Turn on LED, then blink with delay.</li> </ul>		Name Address Value Bits
Programming References:		RESEL 0x01
(How to easily access & what to use each for)		CTRLE RUNSTBY (ADC0) 04
<ul> <li>Device datasheet</li> </ul>		Address: 0x600 00     Address: 0x600 00
<ul> <li>Datasheet (from IO view)</li> </ul>		₩ 🖻 MUXPOS 0x606 0x09 == = 🗋 🗖
<ul> <li>IO view (debugging)</li> </ul>	¢ sthest;/sheets = C © strai-stato-docal-setuites	an mar 1 an ann an Anna an Anna an Anna anna a
Kit user-guide & schematics	ATtiny417 / ATtiny414 / ATtiny416 /	Attony617
<ul> <li>Device header files</li> </ul>	ADC - Analog to Digital Converter	Control A
Editor (Visual Assist)	+ D Oversee + D Functional Description	0 maat 0.000 Resett 0.000 Access: 0 m 7 0 5 4 1 2 1 0
AVR <sup>®</sup> LibC	Fighter Decorption     CTRLA	Rulester         RESSEL         Etranut           Access         R/W         R         R         R         R/W         R/W           Reset         0         0         0         8         0         0         0
Atmel START	C CTRLD C CTRLC C CTRLD	BIT 7 - BUNKTRY: Nos in Standby This bit determines whether the ADC needs to run when the chip is in standby sleep mode.

#### 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、これは編集やデベッグの時 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、デベ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。デベッグ時の 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「デバッグ3: I/O表示部、メモリ表示部と監視」だけでなく「エディタ: コートの記述と整理 (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: ATtiny8177°ロジェクト

その過程で以下のコートが書かれます。このコートが簡単とは言え、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す。



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トじ、この行なしではコンパイラが上のコート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表示部がレシ、スタレヘ、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ハイトアドレスで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.DIR SET)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(POR T.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。



#### I/O表示部(デバッグ)

この機能はStart Debugging and Break(デベッグ開始と中断)を用いてデベ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。 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「デ゙^ ゙ッグ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 ロク ラミンク とテ ハ ック 作 美開始)					
釦	機能	キーホート゛ショートカット				
Þu	デバッグを開始して中断	Alt + F5				
Ď	目的対象に取り付け					
	デバッグを開始	F5				
Ш	全て中断	Ctrl + Alt + Break				
	デバッグなしで開始	Ctrl + F5				

	Ports     P	(PORTE) (PORTC) Controlle (OCKBIT) tile Mem/	r (CPUL	
PORI.DIR4のクリックはPB4を出力に	Name	Address	Value	Bits
設定します。	B DIR	0x420	0x10	
「「お町台のため」「「ひけついった」	DIRSET	0x421	0x10	000600000
・Lowか既定のためLEDはONです。	DIRCLR	0x422	0x10	000000000
	DIRTGL	0x423	0x10	00080000
	D OUT	0x424	00400	00000000
	DUTSET	0x425	0x00	00000000
Harris techini - Beccorcocce	DUTCLR	0x426	0x00	00000000
	DUTTGL	0x427	0x00	00000000
	D IN	0x428	OxEF.	
	🖲 🛃 INTFLAGS	0x429	0500	00000000
	🕫 🗎 PINOCTRL	0x430	0x00	0===0000
	🗄 📄 PINLCTRL	0x431	0.000	0===0000
	🛞 🗎 PIN2CTRL	0x432	0.00	0===00000
mendered and a second second	H D PINGCTRL	0x433	0,00	0===0000
the second secon	B D PINACTRL	0x434	0x00	0===0000
and the second s	I DINSCTRL	0x435	0x00	0===0000
S	I PIN6CTRL	0x436	0x00	0===0000
	🗏 🗎 PIN7CTRL	0x437	0.000	0===0000

#### Atmel Studio7資料ダウンロート

データシートはAtmel Studio 7のヘルプ システムを使うことによってもダウンロート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「オフライン資料のダウンロート」で記述されます。

#### Atmel Studio 7 エディタ (Visual Assist)

Visual Assistを装備したAtmel Studio 7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、一方でコート誘導の概要は図1-20.で示されます。次の「エディク: コートの記述と整 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1-20. A ATtiny83 {\$ PIN5	Atmel Studio 7 エディタの言 17 Xplained Pro - 0693 5_bm	秀導概要 main.c → × MyFirstProject.lss	▼ ▼ Co
3 4 5 6 7 8 9 10 11	freated: 3/12/2017 11:31:46 Att	定義領域	移動定義
13 14 15 16 17 18 19 29 21	<pre>int main(void) {     PORTB.DIRSET = PIN4     PORTB.PINSCTRL = POR </pre>	_bm; RT_PULLUPEN_bm;	
22 23 24 25 26 27 28 29	while (1) { { uint8_t \$ if (SW0) { PORT8 } else	Alt + G 移動定義	

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・ハ・イス ヘッタ・ ファイル

ェ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)釦をクリックするか、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1-21. 示唆一覧とMCUァハイス ヘッタ ファイル	
CFile1.txt ATtiny817 Xplained Pro - 0113 iotn817.h • × main.c* Start Pa	age interrupt.h 🖿 🗙 👻
PORT_ISC_enum     POR	- 😤 Go
1169 /* I/O Ports */ 1170 ⊖typedef struct PORT_struct 1171 { projektor2 + DID: /* Data Dispettion */	
<pre>1122 register8_t DIR; /* Data Direction */ 1123 register8_t DIRSET;/* Data Direction Clear */ 1124 1125 register8_t DIRTGL;/* Data Direction Clear */ 1126 register8_t OUT; /* Output Value */ 1127 register8_t OUTSET;/* Output Value Set */ 1129 register8_t OUTCLR;/* Output Value Set */ 1129 register8_t OUTCLR;/* Output Value Clear */ 1120 register8_t OUTCLR;/* Output Value Clear */ 1121 register8_t reserved_0x80; 1125 register8_t reserved_0x80; 1126 register8_t reserved_0x80; 1126 register8_t reserved_0x80; 1126 register8_t reserved_0x80; 1126 register8_t PINCTRL; /* Pin 0 Control */ 1129 register8_t PINCTRL; /* Pin 2 Control */ 1120 register8_t PINCTRL; /* Pin 4 Control */ 1122 register8_t PINCTRL; /* Pin 5 Control */ 1123 register8_t PINCTRL; /* Pin 5 Control */ 1124 1125 register8_t PINCTRL; /* Pin 5 Control */ 1125 register8_t PINCTRL; /* Pin 5 Control */ 1126 register8_t PINCTRL; /* Pin 5 Control */ 1129 register8_t PINCTRL; /* Pin 5 Control */ 1120 register8_t PINCTRL; /* Pin 5 Control */ 1121 register8_t PINCTRL; /* Pin 5 Control */ 1122 register8_t PINCTRL; /* Pin 5 Control */ 1123 register8_t PINCTRL; /* Pin 5 Control */ 1124 register8_t PINCTRL; /* Pin 5 Control */ 1125 register8_t PINCTRL; /* Pin 7 Control */ 1126 register8_t PINCTRL; /* Pin 7 Control */ 1127 1128 register8_t PINCTRL; /* Pin 7 Control */ 1129 register8_t PINCTRL; /* Pin 7 Control */ 1129 register8_t PINCTRL; /* Pin 7 Control */ 1120 register8_t PINCTRL; /* Pin 7 Control */ 1121 122 123 124 125 125 125 125 125 125 125 125 125 125</pre>	DIR   DIRCLR   DIRSET   DIRTGL   IN   INTFLAGS   OUT   OUTCLR   OUTSET   OUTTGL
<pre>1396</pre>	「PORTB.」を入力 ・ PORT_構造体を参照 (デバイス ヘッダ ファイル) t input buffer enabled */ fer disabled */

#### 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。



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1-3. ATtiny817 Xplained Pro GPIO接続					
シルク文字	ATtiny817 GPIOビン				
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。
- ・SWO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ッ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1-24.で示されるように、ライフ<sup>ラ</sup>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<sup>ラ</sup>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タ<sup>、フ</sup>アイルが単位部(Module)名で示されます。例えば"interru pts(割り込み)"を検索すると、関連インクルート<sup>\*</sup>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。

⊠1−24. AVR Libc715	フリ参照基準					
avr-libc Modules x	¥ 200		and the Real Property lies	10		and March 1977
← → C @ www.nongnu.or	g/avr-libc/user-manual/r	nodules.html			x 🖬 🔩 🖬	🖪 🗟 🚺 📓
🔠 Apps 📋 Treehouse Learn Wei	Home - Atmel Techni	😵 Training Jira 🌘	npi-trd/ASF GitHub 🔒	WebHome < Custom	Apps Program Revie	* Other bookmark
AVR Libc Home Page				-	interrupts	Development Pages
Main Page	User Manual	Library	Reference	FAQ	Example Projects	
Modules						
Here is a list of all modules:						
						[detail level 1 2]
<alloca.h>: Allocate space in</alloca.h>	n the stack					
<assert.h>: Diagnostics</assert.h>						
<ctype.h>: Character Operat</ctype.h>	tions					
<errno.h>: System Errors</errno.h>	anu and and					
<intrypes.n>: Integer Type co</intrypes.n>	onversions					
<main.n>: Mainematics</main.n>						
<stdint ba:="" integer<="" standard="" td=""><td>Types</td><td></td><td></td><td></td><td></td><td></td></stdint>	Types					
<stdint.iv: facilit<="" io="" standard="" td=""><td>ties</td><td></td><td></td><td></td><td></td><td></td></stdint.iv:>	ties					
<stdlib.h>: General utilities</stdlib.h>						
<string.h>: Strings</string.h>						
<time.h>: Time</time.h>						
<avr boot.h="">: Bootloader Su</avr>	pport Utilities					
<avr cpufunc.h="">: Special AV</avr>	R CPU functions					
<avr eeprom.h="">: EEPROM ha</avr>	andling					
<avr fuse.h="">: Fuse Support</avr>						
<avr interrupt.h="">: Interrupts</avr>						
<avr io.h="">: AVR device-spec</avr>	ific IO definitions					
<avr lock.h="">: Lockbit Suppo</avr>	rt					
<avr pgmspace.h="">: Program</avr>	Space Utilities					
<avr power.h="">: Power Reduc</avr>	ction Management					

-25. AVR Libcでの割り込みの依 D AVR Libc Home Page ×	起い方	
- → C ③ www.nongnu.org/avr-libc/		
Global manipulation of the interrupt f	120	
The global interrupt flag is maintained in the I bit of th	e status register (SREG).	tered by code running within an interrupt context
see <util atomic.h="">. Frequently, interrupts are being disabled for periods of</util>	of time in order to perform certain operations w	(thout being disturbed; see Problems with
#define sei() Enables interrupts by se	with respect to compiler optimizations.	
#define cli()		
Macros for writing interrupt handler f	unctions	
#define ISR(vector, attributes)		
#define SIGNAL(vector)		
#define EMPTY_INTERRUPT(vector)	へぃね゛ファイルご白 カロ	#include <avr interrupt.h<="" th=""></avr>
#define ISR_ALIAS(vector, target_vector)	い外外が加速加	_
#define RADISR vact	関連IRQベクタ	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誂えるためにトライバやミドル 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、生成されたコート、プロジェクトをダウンロート、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プログラミング参照基準の一覧を使って0からコート'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、構成コート'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 333333です。

As START X	And in case of the local division in which the local division in t	of a diam and a	And in case of the local division of the loc	the second line	successive in a summer	-		al an
← → C ③ start.atmel.com/#pro	ect				० 🕁 📴 🧣		G 🛛 🕯	3
🛛 Apps 🕒 Treehouse Learn Wei 🎦 H	iome - Atmel Techni 🤶 Traini	ing Jira 🛞 npi-trd/ASF	GitHub <u> </u> WebHome	e < Custorn 🕒 A	Apps Program Revie	39	Other bool	kma
1tmel START					🗲 Re	turn To Front P	age   A	Abo
REATE NEW PROJECT								
lect device or board before creating	ng a new project. You can f	filter devices and boa	rds by what softwar	e you need and	also with hardware req	uirements such a	as memory	y
zes.								
-								
FILTERS	F	RESULTS						
HARDWARE	CO	017	$\otimes$	Show all	Chow only boards	C show only	devices	
- HARDWARE	00	817	6	O show an	O show only boards	O show only o	actices.	
EARCH FOR SOFTWARE		Name	Architecture	Package	Pins	Flash	SRAM	6
EARCH FOR SOFTWARE		Name ATtiny817-MNRES	Architecture	Package VQFN24	Pins 24	Flash 8 KB	<b>SRAM</b> 512 B	•
EARCH FOR SOFTWARE		Name ATtiny817-MNRES ATtiny817-MNR	Architecture AVR AVR	Package VQFN24 VQFN24	Pins 24 24	Flash 8 KB 8 KB	SRAM 512 B 512 B	6
EARCH FOR SOFTWARE		Name ATtiny817-MNRES ATtiny817-MNR ATtiny817-MFR	Architecture AVR AVR AVR AVR	Package VQFN24 VQFN24 VQFN24 VQFN24	Pins 24 24 24 24	Flash 8 KB 8 KB 8 KB 8 KB	SRAM 512 B 512 B 512 B	0
EARCH FOR SOFTWARE Find software MIDDLEWARE + Bootloader		Name ATtiny817-MNRES ATtiny817-MNR ATtiny817-MFR Tiny817 QTou	Architecture AVR AVR AVR AVR uch Moisture Demo	Package VQFN24 VQFN24 VQFN24 VQFN24	Pins 24 24 24 24	Flash 8 KB 8 KB 8 KB	SRAM 512 B 512 B 512 B	
EARCH FOR SOFTWARE Find software MIDDLEWARE + Bootloader + Crypto	© © [	Name ATtiny817-MNRES ATtiny817-MNR ATtiny817-MFR Tiny817 QTou ATtiny817 Xpl	Architecture AVR AVR AVR uch Moisture Demo Iained Pro	Package VQFN24 VQFN24 VQFN24 VQFN24	Pins 24 24 24 24	Flash 8 KB 8 KB 8 KB 8 KB	<b>SRAM</b> 512 B 512 B 512 B	
EARCH FOR SOFTWARE Find software MIDDLEWARE + Bootloader + Crypto	© (	Name ATtiny817-MNRES ATtiny817-MNR ATtiny817-MFR Tiny817 QTot ATtiny817 Xpl ATtiny817 Xpl ATtiny817 Xpl	Architecture AVR AVR AVR AVR uch Moisture Demo Iained Pro Iained Mini	Package VQFN24 VQFN24 VQFN24	Pins 24 24 24 24	Flash 8 KB 8 KB 8 KB	<b>SRAM</b> 512 B 512 B 512 B	
EARCH FOR SOFTWARE Find software MIDDLEWARE + Bootloader + Crypto DRIVERS	© 0 1 0 1 0 0	Name ATtiny817-MNRES ATtiny817-MNR ATtiny817-MFR Tiny817 QTou ATtiny817 Xpl ATtiny817 Xpl	Architecture AVR AVR AVR AVR Uch Moisture Demo Iained Pro Iained Mini	Package VQFN24 VQFN24 VQFN24	Pins 24 24 24 24	Flash 8 KB 8 KB 8 KB 8 KB	SRAM 512 B 512 B 512 B	
EARCH FOR SOFTWARE  Find software  MIDDLEWARE  + Bootloader  + Crypto DRIVERS AC	<ul> <li>○</li> <li>○</li> <li>○</li> <li>○</li> <li>○</li> <li>○</li> <li>○</li> <li>○</li> </ul>	Name ATtiny817-MNRES ATtiny817-MNR ATtiny817-MFR Tiny817 QTou ATtiny817 Xpl ATtiny817 Xpl	Architecture AVR AVR AVR AVR Uch Moisture Demo Iained Pro Iained Mini	Package VQFN24 VQFN24 VQFN24	Pins 24 24 24 24	Flash 8 K8 8 K8 8 K8	SRAM 512 B 512 B 512 B	
EARCH FOR SOFTWARE Eind software MIDDLEWARE + Bootloader + Crypto DRIVERS AC ADC		Name ATtiny817-MNRES ATtiny817-MNR ATtiny817-MFR Tiny817 QTou ATtiny817 Xpl ATtiny817 Xpl	Architecture AVR AVR AVR AVR Uch Moisture Demo Iained Pro Iained Mini	Package VQFN24 VQFN24 VQFN24	Pins 24 24 24 24	Flash 8 K8 8 K8 8 K8	SRAM 512 B 512 B 512 B	
		Name ATtiny817-MNRES ATtiny817-MNR ATtiny817-MFR Tiny817 QTou ATtiny817 Xpl ATtiny817 Xpl	Architecture AVR AVR AVR Juch Moisture Demo Jained Pro Jained Mini	Package VQFN24 VQFN24 VQFN24	Pins 24 24 24 24	Flash 8 KB 8 KB 8 KB 8 KB	SRAM 512 B 512 B 512 B	

#### 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 Atmel START ATTINY817 + Return To Front Page | Help And Support {} VIEW CODE P SAVE CONFIGURATION EXPORT PROJECT PINMUX CONFIGURATOR ? Pin label Board label Signa Show labels... 🗸 🕀 Zoom in \ominus Zoom out 🛛 Auto fit 🗸 Instance name Pad User Header Pin Label PA6 EXI1,QL ADC(+).. Component name PINMUX PA7 EXT1,QT\_ ADC(-),Q. Signal label PB7 EXT1 GPI01 Board header 🗹 Board label 🔥 10 PB6 EXT3 IRQ/GPIO 3P10 061\_55 PB5 EXT3 SPI SS B. 11 12 PB4 EXT1 GPIO2/L 13 PB3 EXT1,EX. USART\_R. $\odot$ 14 PB2 EXT1,EX. USART\_T. SPI MISO SPI MISO DGI MISO 15 PB1 EXT1 PWM(-) SPI 55 A SPI SCK.SPI SCK.DBI SCK 16 PBO EXT1 PWM(+) PWN(+) 17 PCO EXT1.EX... SPI SCK ... PWM(-) SPI\_SS\_BIGPIO T TXD XOUTS2 18 PC1 EXT1,EX. SPI\_MIS. 19 PC2 EXT1,EX. SPI\_MOS. 20 PC3 EXT1 SPI\_SS\_A 21 PC4 DGI SPI DGL\_SS 22 PC5 EXT3 GPIO 23 PAD PA1 EXT1.EX... I2C SDA .... 24 4.11

#### 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. エティタ: コートの記述と整理 (Visual Assist)

Atmel Studio 7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



#### 映像: Atmel Studio 7 エディタ (Visual Assist)

1.「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。



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。



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.PIN5	CTRL = pp	
	IN       PORT_PULLUPEN_bm         IN       PORT_PULLUPEN_bp         IN       PORTA_PINSCTRL         IN       PORTB_PINSCTRL         IN       PORTC_PINSCTRL	<pre>#define PORT_PULLUPEN_bm 0x08 /* Pullup enable bit mask. */ Accept with: <tab> or <enter></enter></tab></pre>
	A?	

- 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)
PORTE	B.DIR = PIN4_bm;
PORTE	B.PIN5CTRL = por
	👘 PORT_PULLUPEN_bm 🔺 单 #define PORT_PULLUPEN_bm 0x08
	PORT PULLUPEN bp /* Pullup enable bit mask. */
	PORT struct Accept with: <tab> or <enter></enter></tab>
	T PORT t
	PORTA
	PORTA DIR
	IPP PORTA_IN
	Enumまたはtypedef 🚅 🕴 🕇 📫 typedef構造体
	Enum/typedefのパラメータ 🛁 #Define 🖿 Extern(全域変数)

4. if(){...}else{...} Visual assistコート 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。

![](_page_31_Figure_7.jpeg)

![](_page_32_Figure_1.jpeg)

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。

![](_page_32_Figure_3.jpeg)

- 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(デハックなしで開始) ▶ (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1-30. 方式抽出 PORTB.OUTCLR = PIN4 bm; /\* Turn LED o Goto Implementation Alt+G } else Refactor (VA) Shift+Alt+R Rename... { Surround With (VA) . Change Signature.. PORTB.OUTSET = PIN4\_bm; /\* Turn LED o 1 Insert Snippet... Ctrl+K, Ctrl+X 3 Encapsulate Field 1 Surround With... Ctrl+K, Ctrl+S Create From Usage... Shift+Alt+C Breakpoint Create Declaration Create Implementation Run To Curson Ctrl+F10 Add Missing Case Statements Run Flagged Threads To Cursor Add Member... X Cut Ctrl+X Add Similar Member... С Сору Ctrl+C Add Include 6 Paste Ctrl+V Add/Remove Braces Outlining ٠ Extract Method ... Add Data Plot Introduce Variable. Remove Data Plot Implement Interface View Help 0 Document Method

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。

ОК	Cancel
	ОК

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 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。

if (!(PORTB.IN & F	PIN5 bm)	) /* Check switch state */	Alt+G	3		
LED_on();		Refactor (VA)	AICO		B	Chife Alt D
} else {		Surround With (VA)		2	Change Signature	Shift+Alt+K
, t	<b>*</b> 1	Insert Sninnet	Ctrl+K Ctrl+X		Encansulate Field	
LED_off();	<b>†</b> 1	Surround With	Ctrl+K, Ctrl+S		Create From Usage	Shift+Alt+C
}		Breakpoint		2	Create Declaration	Shire Air e
	*	Run To Cursor	Ctrl+F10		Create Implementation	
	h:	Run Flagged Threads To Cursor			Add Missing Case Statements	
	ж	Cut	Ctrl+X		Add Member	
	Ð	Сору	Ctrl+C		Add Similar Member	
	റ	Paste	Ctrl+V		Add Include	
		Outlining	•	Add/Remove Braces		
- 1	n Co	Add Data Plot			Extract Method	
	w	Remove Data Plot           View Help		_	Introduce Variable	
	0				Implement Interface	
				-	Create File	
					Move Selection to New File	
					Move Implementation to Source File	
					Bassing Eller	

図1-33.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。

Introduce Variable		? <mark>×</mark>
Variable signature:		
uint8_t SW0_state		
	ОК	Cancel

![](_page_34_Picture_5.jpeg)

**助言**: ブール値で扱うのに追加のヘッダをインクルート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。

![](_page_34_Figure_8.jpeg)

- 9. Refactor(整理)⇒Extract Method(方式(メソット)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り	1
<pre>28</pre>	<pre>m; /* Enable pull-up Tor SWØ pin */ tate(); ich state */  Preview of method signature:     void LED_set_state(uint8_t SW0_state)     OK Cancel    </pre>
L	

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。

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。

id LED_set_s	stat tate	e(uint8_t SW0_state) ) /* Check switch state *	*/		
{		Goto Implementation	Alt+G		
LED_0		Refactor (VA)	•	Rename	Shift+Alt+R
		Surround With (VA)	۲	Change Signature	
} else {	<b>t</b> 1	Insert Snippet	Ctrl+K, Ctrl+X	Encapsulate Field	
LED_0	<b>†</b> 1	Surround With	Ctrl+K, Ctrl+S	Create From Usage	Shift+Alt+C
1		Breakpoint	•	Create Declaration	
J	k	Run To Cursor	Ctrl+F10	Create Implementation	

図1-36.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。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。

Trendrice .			-R
Rename SW0_state (in file) to:			
state		Rename	Cancel
Display inherited and overridden references	Display uses in comments and	d strings	
Search all projects			

Rename(改名)をクリックし、コード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)
   ł
      PORTB. DIRSET = PIN4_bm;/* LEDピンを出力として構成設定 */PORTB. PIN5CTRL = PORT_PULLUPEN_bm;/* SWOピンに対してプルアップ許可 */
      while(1)
       {
           uint8_t SWO_state = SW_get_state(); /* スイッチ状態読み込み */
                                                /* LED状態設定 */
           LED_set_state(SW0_state);
       }
   }
  uint8_t SW_get_state(void)
   {
      return ! (PORTB. IN & PIN5_bm);
                                                /* スィッチ状態読み込み */
   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 デバック

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ィ、ック(中断点(ブレーク ポイント)設定とコートを通した段階実行)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

			Decement	0.00000028
In this video:			Stack Poin	iter 0x3FFD
Studio 7: AVR MCU Simulator			X Register Y Register	0x0000 0x3FFF
Project Setup:			Z Register	0x0000
Modify project from Studio 7 Editor video			Cycle Cou	inter (2)
<ul> <li>Basic debugging: set breakpoint, step,</li> </ul>			Frequency Stop Wate	1.000 MHz
<ul> <li>Processor view: Demonstrate use of cycle counter &amp; stop watch</li> </ul>			B Registe	rs
<ul> <li>Dissassembly view: difference how code compiled</li> </ul>			in {	t main(void)
<ul> <li>Simulate IRQ (IO view)</li> </ul>	1)	Use read-modify-write in co	de —	PORTB.DIR &= ~PIN4_bm; PORTB.DIR  = PIN4_bm;
Context:	2)	HW read-modify-write regis	ters —	<pre>//PORTB.DIRSET = PIN4_bm; //PORTB.DIRCLR = PIN4_bm;</pre>
<ul> <li>Set up 3 options to clear, then set register bit</li> </ul>	3)	Bit-accessible virtual port I/	0	<pre>//VPORTB.DIR &amp;= ~PIN4_bm //VPORTB.DIR  = PIN4_bm;</pre>
<ul> <li>LED on when switch pressed (using pin change IRQ)</li> </ul>			2	<pre>wmile (1) { }</pre>

#### 映像: AVRシミュレータ デ・バック

上の映像で使われたコードは「エディタ: コート、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。

M ğ ≡   → 1	🕨 🕨 🔅 🔹 🕈 👔 Hex 🔏 📓 - 🖕 🗐 💷 🌐 🖓 📲 🚽 🏙 🖄 🚽 📟 ATtiny817 🚺 No Tool
GccApplication1* 🐄	X main.c ATtiny817 Xplained Pro - 0703 Pending Changes
Build Build Events	Configuration: N/A   Platform: N/A
Toolchain Device Tool*	Selected debugger/programmer
Components Advanced	Simulator EDBG ATML2654041800000703 P Custom Programming Tool
	✓ Preserve EEPROM Debug settings
	✓ Keep timers running in stop mode         ✓ Cache all flash memory except

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ク「作業を開始 するためにStart Debugging and Break(デバック「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)ウィントウを開いてください(また、これはDebug(デバック))→Windows(ウィントウ)→Proces sor 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)ウィントウを開くこともできます。

Standard Mode 🔻 🕇 💎	pro	×
Most Recently Used (2)		
Debug → Windows → Processor Status		
Standard Mode 🔻 1	dis	×
Standard Mode <b>▼1</b> Most Recently Used (2)	dis	×

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。これはパグと遅延の両 方に対してCycle Counter(周期計数器)を正確にします。St opwatch(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 <	実行されつつある 命令のアドレス
Stack Pointer	0x00003821 <	ー現在のスタック ポインタイ
X Register	0x0002	
Y Register	0x3FF1	
Z Register	0x3824	
Status Register	ITHSV	NZC
Cycle Counter	8186	← 模倣開始からの 経過周期数
Frequency	1,000 MHz	正地/可/9398
Stop Watch	8 186,00 us	← 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。

		Read-modify-wri Eile Edit View	te (Debugging) - AtmelStudio VAssisty ASF Project Bu IIII - IIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	Hol Debug Iools Window Help ⊃ + C + [] = Q + [] Holog Investor + Hee NS ■ + , [] ⊟ Holog Investor + Hee NS ■ + , [] ⊟ Holog Investor +	-   🎜 enable_sleep.m	iode •	ਙ≁©ਛਡ⊡
		Data Visualizer	Disassembly # × Read-modif	y-write main.c Read-modify-write.lss		Processor Status	
1	S ≢include <avr io.h=""> 9 8 ⊜int main(void)</avr>	Address main Wewing Options 00000024 JMP 0: C:\Users\V40 {	x0000000 Jump 1959\Documents\Atmel Stud	io\7.0\Read-modify-write\Read-modify-write\Debug//./main.c		Name     Program Counts     Stack Pointer     X Register     Y Register     T Register	Value or 0x0000020 0x3FFD 0x0000 0x3FFF 0x0000
1	1 {	PORTB	.DIR &= ~PIN4_bm	;		Z Register Status Register	20000000000000000000000000000000000000
• 1 1 1	BORTB.DIR &= ~PIN4 bm; PORTB.DIR  = PIN4_bm; S	© 00000026 00000027 00000028	LDI R30,0x20 LDI R31,0x04 LDD R24,Z+0	Load immediate Load immediate Load indirect with displacement		Frequency Stop Watch	1000 MHz 1000 µs
1	<pre>6 //PORTB.DIRSET = PIN4_bm; 7 //PORTB.DIRCLR = PIN4_bm; 8</pre>	00000029 00000020	ANDI R24,0xEF	Logical AND with immediate	Cycl	e Counter	0
1	<pre>9 //VPORTB.DIR &amp;= ~PIN4_bm; 0 //VPORTB.DIR  = PIN4_bm; 1</pre>	PORTB 00000028	.DIR  = PIN4_bm; LDD R24,Z+0	Load indirect with displacement	Stop	Watch	1.000 MHz 0.00 µs
• 2	<pre>2 asm("nop"); 3 4 while (1)</pre>	0000002C 0000002D	ORI R24,0x10 STD Z+0,R24	Logical OR with immediate Store indirect with displacement		90/0 R07 R08	0.00
2	5 { 6 } 7 }	0000002E 0000002F	NOP No op RJMP PC-0x0000	eration Relative jump		R10 R11 R12 R13	0x00 0x00 0x00 0x00

(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ー変更ー書き(前図)とビット操作命令(次図)間で生成されたアセンブリ コートでの違いに気付いてください。

![](_page_39_Figure_2.jpeg)

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
ソフトウェア読みー変更ー書き	10	
ハートウェア読みー変更ー書きレジスタ	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。 デハッグ時に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。

		in the second	
Name	Valu	Je	
<ul> <li>I/O Ports (PORTB)</li> </ul>			
Virtual Ports (VPORTE	3)		
Name	Address	Value	Bits
DIR DIR	0x420	0x00	0000000
DIRSET	0x421	0x00	0000000
DIRCLR	0x422	0x00	0000000
DIRTGL	0x423	0x00	00000000
DUT DUT	0x424	0x00	00000000
OUTSET	0x425	0x00	00000000
OUTCLR	0x426	0x00	00000000
OUTTGL	0x427	0x00	00000000
IN IN	0x428	0x00	00000000
INTFLAGS	0x429	0x20	
🛃 INT		0x20	
E PINOCTRL	0x430	0x00	
E PIN1CTRL	0x431	0x00	Bit 5 1000
	0v432	0_00	

下で示されるように割り込み(ISR)に的中します。 以降のコートで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。

	76	<pre>□ISR(PORTB_PORT_vect)</pre>		😑 📿 INTFLAGS	0x429	0x20	0000000
	77	{		INT 💽		0x20	
0	78	<pre>uint8 t intflags = PORTB.INTFLAGS;</pre>		I PINOCTRL	0x430	0x00	
	79	PORTB.INTFLAGS = intflags:	Internet in the second	I PIN1CTRL	0x431	0x00	
	80		a	PIN2CTRL	0x432	0x00	
	81	bool SW state = SW get state():	And the second	E PIN3CTRL	0x433	0x00	
	01	LED sot state(Shi state);		🗉 🗎 PIN4CTRL	0x434	0x00	
	82	LED_Set_state(SW_state);	an april	🖃 🗎 PIN5CTRL	0x435	0x09	
	83			INVEN		0x00	
	84	_}	- N(84) (5.45)	ISC ISC		0x01	
	85			DULLUPEN		0x01	
			and other sector	PIN6CTRL	0x436	00x0	
			and ing of the lot	PIN7CTRL	0x437	0x00	
			100 V 101 KP 101 - 100 MP 101	Arrister in a second			ROCKAL CONTRACTOR STREET

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 バイス外囲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。

Atmel Studio 7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するご覧ください。

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

![](_page_41_Figure_1.jpeg)

# 1.13. 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Atmel Studio 7のディ、ックが能力を紹介します。主な話題は中断点(フレーク 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

![](_page_41_Figure_4.jpeg)

#### 映像: Atmel Studio 7 ディックー1

「エディタ: コート」の記述と整理 (Visual Assist)」項で作成されたのと同じコート」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1-37.で示されるように、スィッチ状態を得る行に中断点を設定してください。

図1-37. 中国	断点を配置
65	<pre> □ISR(PORTB_PORT_vect) </pre>
66	{
67	<pre>uint8_t intflags = PORTB.INTFLAGS;</pre>
68	<pre>PORTB.INTFLAGS = intflags;</pre>
Ø 🖗	
5 70	<pre>bool SW_state = SW_get_state();</pre>
Location: n	<pre>nain.c, line 70 character 1 tate(SW_state);</pre>
72	
73	}

![](_page_42_Picture_1.jpeg)

- ・エディタウィントウ左端の灰色バーをクリック。
- ・最上部のメニュー ハーで、Debug(デバック)→Toggle Breakpoint(中断点ON/OFF)へ行く。
- ・キーホートでの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キット上のスィッチ(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。

![](_page_42_Figure_6.jpeg)

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。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ァーカスを持ちます。

- 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。 スィ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、関数が返すスィッチの状態は何ですか?。仮定はスィッチ押下 が割り込みを起動すること、スィ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、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、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ト、ショートカットは右図で略述されます。

![](_page_42_Figure_10.jpeg)

表1-4. Atmel	Studio釦機能	(コート、段階実行)

釦	機能	キーホ゛ート゛ 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
k	カーソルまで走行	Ctrl + F10
Î	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ィ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ィ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LE D\_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**: スィ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ィッチの状態が記録されます。この行を外側段階実行 **?** する時にスィ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。

![](_page_42_Figure_16.jpeg)

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。敏速起動ハーは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。

Standard Mode 🔻	1 bre	ak 🗙	-	Ð
Most Recently Used (1)				
Debug → Windows → Breakpoints (Alt	t+F9)			
Menus (7)	4	S Debug → Windows → Breakpoin	ts (Alt	+ F91
Debug → Windows → Breakpoints (Alt	t+F9)			
Debug → Windows → Data Breakpoint	s			•
Debug → Toggle Breakpoint (F9)				
Debug $\rightarrow$ New Breakpoint $\rightarrow$ New Data	Break	point (Ctrl+Shift+D, B)		
Debug $\rightarrow$ New Breakpoint $\rightarrow$ Function	Breakp	oint (Ctrl+B)		
🎒 Debug → Delete All Breakpoints (Ctrl+	Shift+I	F9)		

#### 図1-42. Debugメニューでの"break"的中

	Windows	•	51	Breakpoints	Alt+F9
en i	Start Debugging and Break	Alt+F5	5	Data Breakpoints	
Ď,	Attach to Target			Processor Status	
	Stop Debugging	Ctrl+Shift+F5			
Þ	Start Without Debugging	Ctrl+Alt+F5	-		
	Disable debugWIRE and Close				
	Continue	F5			
G	Execute Stimulifile				
G.	Set Stimulifile		L		
ð	Restart				
П	Break All	Ctrl+F5			
60	QuickWatch	Shift+F9			
*	Step Into	F11			
?	Step Over	F10			
:	Step Out	Shift+F11			
k	Run To Cursor	Ctrl+F10			
î	Reset	Shift+F5			
	Toggle Breakpoint	F9			
	New Breakpoint	K		New Data Breakpoint	Ctrl+Shift+D, B
5	Delete All <mark>Breakpoin</mark> ts	Ctrl+Shift+F9		Function Breakpoint	Ctrl+B
	Disable All <mark>Breakp</mark> oints		-		
	Clear All DataTips				
	Export DataTips				
	Import DataTips				
	Save Dump As				
Ø	Options				
s	Getting Started Properties		I		

![](_page_44_Picture_1.jpeg)

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、敏速起動ハーで"call"を入力してDebug(デハック)→Windows(ウィントウ)→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)ウィントウを開いてください。

注: このウィントウを開くにはデバック作業を活性(有効)にする必要があります。

	Standard Mode	₹1	call	×
Most Recently	Used (2)			
Æ Debug →	Windows → Call Stack (A	l++7)		
Densed.	Con Stack (A	1171		
Text Edito	r → All Languages → Gen	eral (	apply Cut or Copy	to blank lines T
Text Edito	r → All Languages → Gen	eral (	Apply Cut or Copy	/ to blank lines, T
Text Edito	r → All Languages → Gen	eral (	Apply Cut or Copy	/ to blank lines, T
Ö Text Edito Menus (1) (■ Debug →	windows → Call Stack (A	eral (/ it+7)	Apply Cut or Copy	/ to blank lines, T

3. それはコート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、これはコンパ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

Call Stack			
Name			
C Getting	Started.elf! LEI	D_on Line: 39	
Getting	Started.elf!v	ector_4 Line: 74	
Call Stack	Breakpoints	Command Window	Immediate Window
Stopped			

i

情報: 呼び出し指示はコンパイラ最適化のために異なります。このコートは理解するのが比較的簡単で、例えコンパイラで最適化され、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。もっと複雑なプロジェクトではパグを探し出すのに時にはコンパ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。

Pending Chang	es (master)	main.c 🗢 🗙
SPORTB_POF	T_vect	▼ ↓ #define PORTB_PORT_vect_VECTOR(4)
64		
<b>65</b> E	ISR (PORTB	PORT_vect)
66	{	
67	uint8_	_t intflags = PORTB.INTFLAGS;
60	PORTR	INTELAGS = intflags:

- 4. Stop Debugging(デバッグ停止) = 釦をクリック、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1-48.でのように、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(コンハ°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。

ting Started* 🕘	K main.c	
Build Build Events Toolchain*	Configuration: Active (Debug)	Platform: Active (AVR)
Device	A MVP/GNULCommon	
Tool Components Advanced	AVR GNU Compiler     General     Output Files     AVR/GNU C Compiler     General     Preprocessor     Symbols     Directories     Optimization     Debugging     Warnings     Miscellaneous     AVR/GNU Linker     General     Libraries     Optimization	AVR/GNU C Compiler       Optimization         Optimization Level:       Optimize (-01)         Other optimization flags:       Optimize (-01)         Image: Prepare functions for gart       Optimize most (-03)         Image: Prepare data for garbage       Optimize for size (-05)         Image: Prepare data for garbage       Optimize debugging experience (-Og)         Image: Prepare data for garbage       Optimize debugging experience (-Og)         Image: Prepare data for garbage       Optimize debugging experience (-Og)         Image: Prepare data for garbage       Optimize debugging experience (-Og)         Image: Prepare data for garbage       Optimize debugging experience (-Og)         Image: Prepare data for garbage       Optimize debugging experience (-Og)         Image: Prepare data for garbage       Optimize debugging experience (-Og)         Image: Prepare data for garbage       Optimize debugging experience (-Og)         Image: Prepare data for garbage       Optimize debugging experience (-Og)         Image: Prepare data for garbage       Optimize debugging experience (-Og)         Image: Prepare data for garbage       Optimize debugging experience (-Og)         Image: Prepare data for garbage       Optimize debugging experience (-Og)         Image: Prepare data for garbage       Optimize debugging experience (-Og)         Image: Prepare data for garbage

日 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ト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1-49.で示されるように、コートが実際 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。

助言: Atmel 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、中断点がディブッグ中に無視されたように思えたり、 実行の流れがコート、段階実行中に従うのが難しい場合に役立ち得 ます。

図1-49. 最適化なしでの呼び出しスタック

Call Stack

Getting Started.elf! LED\_on Line: 39 Getting Started.elf! LED\_set\_state Line: 57 Getting Started.elf! \_\_vector\_4 Line: 74

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に使われるコート

```
/*
LEDはスィ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&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. 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Atmel Studio 7でのもっと高度なデベッグの話題を網羅します。主な話題はコート゛ 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

![](_page_47_Figure_4.jpeg)

![](_page_47_Figure_5.jpeg)

![](_page_47_Figure_6.jpeg)

1. 使われるコート(下参照)は「エディタ: コートの記述と整理 (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(デバック)開始) ▶ 釦をクリックすることによって新しいデバック)作業を開始してください。
- 4. キット上のSWO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。

![](_page_48_Figure_9.jpeg)

**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1-51.でのように、変数値、データ型、メモリアト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。

図1-51. Watch(監視)ウィント・ウ・	へ変数追加		
main.c* + ×			
SW_get_state.SW0_edge	_cc 🔹 🌻 🕻	static uint8_t SW0_edge_count = 0	•
* multiple of */	3		
<pre>return SW0_cur_</pre>	state    !(	(SW0_edge_count % 3);	
[}			
86 % 🔹 📢			
Watch 1		- M. H. L.	्र
Name	Value	Туре	
SW0_edge_count	26	uint8_t{data}@0x3e01	
Autos Locals Watch1 C	all Stack Br	reakpoints Output Error List	
Stopped			

- 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。

ļ	図1-52. Watch(監視)ウィントウで新しく更新された変数値		
	Watch 1		
	Name	Value	Туре
	SW0_edge_count	3	uint8_t{data}@0x3e01

![](_page_49_Picture_8.jpeg)

情報: Watch(監視)ウィントウのValue(値)列は監視ウィント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. 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)ウィント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。

![](_page_50_Picture_1.jpeg)

結果: Atmel Studioは今やコート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、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、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端計数毎にデベ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)ウィント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

Location	main.c, Line: 66, Character: 1, Must m	atch source			
Conc	ditions		((SM0 adaption of \$1.5)		10-
	Conditional Expression 4 1 is	true	(Iswo_edge_count x5) == 0) at swo_curs	tate	Can
	Add condition				
Actio	205				

- 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ト、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ト、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、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: 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、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)ウィント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1-54. 活動付き中断点の例

SW0_prv_state = SW0_cur_state;	<pre>/* Keep track of previous state */</pre>	
		Breakpoint Settings >
Location: main.c, Line: 60, Character: 1, Must m	atch source	
Conditions		
<ul> <li>Actions</li> </ul>		
Log a message to Output Window:	Prv state:{SW0_prv_state}, Cur_state:{SW0_cur_state}, Edge count:{SW0_e	dge_count} (j) Cance
Continue execution		
Close		
Close		

- 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SWOが押された場合、内容が更新されます。

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	

▲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、実行を一時的に停止します。結果として、実行タイングが影響を及ぼされます。控え目な手法はSWO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SWOが押された時にだけ一時的に停止させますが、コート、の1行によって遅らされるデバッグウィント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: Atmel 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。

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

条件付きと活動付きの中断点に使われるコート

```
#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)
ł
    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 */
    ł
        SW0_edge_count++;
   SW0_prv_state = SW0_cur_state;
                                                  /* 直前状態の把握 */
    /*
    * スィッチが押されるか、または端計数器が3の倍数の時にスィ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. 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Atmel Studio 7でのもっと高度なデバックが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)ウィントウの使い方もです。 開始に際しての話題

In this video:	I/O View	R D Clock controll	Name	Value
in this video.		O clock selec	(MCLKCTRLA)	20x00 •
Studio 7: Debugging 3		Name	Address Value	Bits
Context:		MCLKCTRLA	0x60 0x00	00000
Project from Debugging 2 add function to save			0.01	
data to eenrom. Change clock freq to 10 MHz	_PROTECTED_WRIT	E(CLKCTRL.MCLKCTRLB,	CLKCTRL_PDIV_2X_gc	CLKCTRL_PEN_br
Features Covered:				
I/O View:	Memory V	eeprom_wri	<pre>llo[] = "Hello World! te_block(Hello,(void*</pre>	"; )0,sizeof(Hello));
<ul> <li>Configuration Change Protect (CCP) registers</li> </ul>	0	27 uint8_t Hi 28 eeprom_wrl	te_block(Hi,(void*)0,	sizeof(H1));
Configuration Change Protect (CCP) registers     Memory view:	© 110 % Men	27 28 eeprom_wrl 00y1	te_block(Hi,(void*)0,	sizeof(H1));
<ul> <li>Configuration Change Protect (CCP) registers</li> <li>Memory view:</li> <li>EEPROM Write (AVR<sup>®</sup> LibC)</li> </ul>	110 4 Men Mer	27 28 eeprom_wrl: eprom_wrl: eoyi data EEPROM a 0x1400 48 65 6c 6c 6f	-   20 57 6f 72 6c 64 21 00	sizeof(Hi));
<ul> <li>Configuration Change Protect (CCP) registers</li> <li>Memory view:         <ul> <li>EEPROM Write (AVR<sup>®</sup> LibC)</li> </ul> </li> <li>Watch:</li> </ul>	130 % Hen da da da da da da da	27         ufint@_t Hill           28         eeprom_wrli           30071         a 0.1400         48 65 66 67           301400         48 65 66 67         69           301412         ff ff ff ff ff ff         69.1412           401412         ff ff ff ff ff ff ff ff         60.1432           50142         ff ff ff ff ff ff ff ff         60.1432		<pre>slzeof(H1));  ff Hello world1.y  ff yyyyyyyyyyyyy ff yyyyyyyyyyyy ff yyyyyyyy</pre>
<ul> <li>Configuration Change Protect (CCP) registers</li> <li>Memory view: <ul> <li>EEPROM Write (AVR<sup>®</sup> LibC)</li> </ul> </li> <li>Watch: <ul> <li>Cast pointer to array of specified size, so can view in Watch Window</li> </ul> </li> </ul>	Watch Vie	27 ufrit@_thu eeprom_wrl 28 0071 0071 0071 0071 007400 007400 007400 00740000000000	-    - w 3003 H1 + 0 -    - 28 57 67 72 66 64 21 00 ff ff ff ff f 72 66 64 21 01 ff	sizeof(H1)); off Helle Harld:g ff yyyyyyyyyyyyy ff yyyyyyyyyyyy ff yyyyyyyy
<ul> <li>Configuration Change Protect (CCP) registers</li> <li>Memory view: <ul> <li>EEPROM Write (AVR<sup>®</sup> LibC)</li> </ul> </li> <li>Watch: <ul> <li>Cast pointer to array of specified size, so can view in Watch Window</li> </ul> </li> </ul>	Watch Vie Watch Vie Watch Vie Watch Vie	27 27 27 27 27 27 27 27 27 27		sizeof(H1)); off Helle world!.g off Helle world!.g ff yyyyyyyyyyyyy ff yyyyyyyyyyyyy ff yyyyyyyyyy
<ul> <li>Configuration Change Protect (CCP) registers</li> <li>Memory view: <ul> <li>EEPROM Write (AVR<sup>®</sup> LibC)</li> </ul> </li> <li>Watch: <ul> <li>Cast pointer to array of specified size, so can view in Watch Window</li> </ul> </li> </ul>	Watch Vie	27 27 27 27 27 27 27 27 27 27		sizeof(Hi)); off Hells world: p off Hells world: p off Systynysystyny ff Systynysystyny ff Systynysystyny ff Systynysystyny ff Systynysystyny ff Systyny ff Systyny

![](_page_53_Figure_4.jpeg)

#### 1.15.1. 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・バック「ツールはデ・バック」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。

![](_page_53_Picture_7.jpeg)

- 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 II 釦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 ハーからDebug(デ'ハ'ック')⇒Windows(ウィントウ)⇒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レジスタを見つけて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GPI OのPB4t°ンでの出力レベルを切り替え、これはATtiny817 Xplain ed Pro上のLEDを制御します。

情報: I/O表示部は何れ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。

<u>x</u> 1–50	6. I/O表示部を	で使うレソノ	(別人の)	ヒット値探作
I/O				
	Filter:			- 1
	Nam			Value
+	IN HUSES (FUSE	)		value
	10 General Pur	, oose IO (G	PIO)	
Đ	I/O Ports (P	ORTA)		
÷	VO I/O Ports (P	ORTB)		
Ŧ	1/O Ports (P	ORTC)		
	Name	Address	Value	Bits
	DIR	0x420	0x10	00000000
	DIRSET	0x421	0x10	
	DIRCLR	0x422	0x10	
	DIRTGL	0x423	0x10	
	DUT	0x424	0x00	
	OUTSET	0x425	0x00	
	OUTCLR	0x426	0x00	
	OUTTGL	0x427	0x00	
	IN IN	0x428	0xEC	
±		0x429	0x00	
±	PINOCTRL	0x430	0x00	
±		0x431	0x00	
		0x432	00x00	
		0x455	0x00	
E I		0x434	0x00	
(F)	PIN6CTRI	0x436	0x00	
Đ	PIN7CTRL	0x437	0x00	

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)

2 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。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. 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. II 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	
Clock controller (CLKCTRL)		
Clock select (MCLKCTRLA)	20MHz internal oscillator	0x00 🔻
Prescaler divition (MCLKCTRLB)	6X	0x08 ▼
Crystal startup time (XOSC32KCTR	1K cycles	• 00x0

![](_page_55_Picture_1.jpeg)

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レジスタを選択してください。

Name	Address	Value	Bits	
🗉 🗎 MCLKCTRLA	0x60	0x00		
MCLKCTRLB	0x61	0x11		
PDIV		0x08		
PEN		0x01		
MCLKLOCK	0x62	0x00		
MCLKSTATUS	0x63	0x10		
OSC20MCTRLA	0x70	0x00		
RUNSTDBY		0x00		
OSC20MCALIBA	0x71	0x9C		
CALSEL20M		0x02		
CAL20M		0x1C		

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ト・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 のマクロ名にカーソルを置いてキーボート、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1-59.で示されるように、コート\*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影	没定									
Clock controller (CLKCTRL)										
Clock select (MCLKCTRLA)	20MHz internal oscillator	0x00 ▼								
Prescaler divition (MCLKCTRLB)	2X	0x00 ▼								
Crystal startup time (XOSC32KCTR	1K cycles	0x00 -								

**助言**: Processor Status(プロセッサ状態)ウィントウはAVRコアに対するレシブスタ表示ツールです。このツールは上部メニュー ハーからDebug (デ・ハ・ッグ)⇒Windows(ウィントウ)⇒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。このウィントウは内 部AVRコアレシブ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 レシブスタのIL・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ク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ト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。

	<pre>uint8_t hello[] = "Hello World";</pre>
•	<pre>eeprom_write_block(hello, (void *)0, sizeof(hello));</pre>
	<pre>uints_t hi[] = "AVR says hi"; eencom write block(hi (woid *)@ sizeof(hi));</pre>
	ccprom_mite_block(ni, (void )0, size((ni)))
	while(1)
ł	<pre>while(1) {     uint8_t SW0_state = SW_get_state(); /* Read switch state */     LED_set_state(SW0_state); /* Set LED state */</pre>

- 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 ハーからDebug(デ'ハ'ッグ')⇒Windows(ウィント'ウ)⇒Memory(メモリ)⇒Memory 1(メモリ1)へ行くことによT り、メモリ ウィント'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.	. EEP	ROM	書	ðì,	<u>አ</u> ታ	⊁後	<b>έ</b> σ,	)	EIJ	表	πŧ	部	更業	斤												
Memory 1	1																									- ± ×
Memory:	data f	EEPRON	1					٠	A	ddre	\$5;	0x1	400,	data	1									•	C	Columns: Auto -
data 0 data 0 data 0 data 0 data 0 data 0 data 0	x1400 x141A x1434 x144E x1468 x1468 x1462 x1482 x149C	48 65 ff ff ff ff ff ff ff ff ?? ?? ??	60 ff ff ff ???	6c ff ff ff ff ??	6f ff ff ff ff ff ?? ??	20 ff ff ff ff ?? ??	77 ff ff ff ff ff ?? ??	6f ff ff ff ff ?? ??	72 ff ff ff ff ?? ??	6c ff ff ff ff ?? ??	64 ff ff ff ff ?? ??	21 ff ff ff ff ?? ??	00 ff ff ff ff ?? ??	ff ff ff ff ff ?? ??	ff ff ff ff ff ?? ?? ??	ff ff ff ff ff ?? ??	ff ff ff ff ff ?? ??	ff ff ff ff ff ?? ??	ff ff ff ?? ?? ??	ff         Hello world!.yyyyyyyyyyyy           ff         gygygygyyyyyyyyyyyyyyyyyyyyyy           ff         gygygygyyyyyyyyyyyyyyyyyyyyyyyyyyyy           ff         gygygygygygyyyyyyyyyyyyyyyyyyyyyyyyyy						
Memory	1																									* # ×
Memory	; data	EEPRON	й.						A	ddre	195;	0x1	400,	data	i.									-	C	Columns: Auto -
data 0 data 0 data 0 data 0 data 0 data 0	0x1400 0x141A 0x1434 0x1434 0x144E 0x1468 0x1468	41 56 ff fr ff fr ff fr ff fr ff fr ff fr	5 51 F fi F fi F fi F fi	20 ff ff ff ff	53 ff ff ff ff 22	61 ff ff ff ff ??	79 ff ff ff ff ??	73 ff ff ff ff ??	20 ff ff ff ff ??	68 ff ff ff ff ff ff ??	69 ff ff ff ff ??	21 ff ff ff ff ??	00 ff ff ff ff ??	ff ff ff ff ff ff i?	ff ff ff ff ff ff ??	ff ff ff ff ff ??	ff ff ff ff ff ff ??	ff ff ff ff ff ??	ff ff ff ff ff ??	ff ff ff ff ff ff ??	ff ff ff ff ff ??	ff ff ff ff ff ff ??	ff ff ff ff ff ??	ff ff ff ff ff ff ??	ff ff ff ?? ??	<pre>ff AVR Says hi! yyyyyyyyyyyy * ff yyyyyyyyyyyyyyyyyyyyy</pre>

**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・ウ

これは「デバック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。

![](_page_57_Picture_3.jpeg)

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: Atmel 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ビ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 (SWO_cur_state != SWO_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)
{
                                              /* LEDをON */
   PORTB. OUTCLR = PIN4_bm;
}
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								
А	2018年1月	初版資料公開								

# Microchipウェフ゛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トウェア支援資料、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、セントとイベントの一覧、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 Notificati on"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デバイスコート、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。
- コート、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トの安全を保証することはできません。コート、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。

コート、保護は常に進化しています。Microchipは当社製品のコート、保護機能を継続的に改善することを約束します。Microchipのコート、保護機能を破る試みはデジタルシ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

Microchipの名前とロゴ、Mcicrochipロゴ、AnyRate、AVR、AVRロゴ、AVR Freaks、BeaaconThings、BitCloud、CryptoMemory、CryptoR F、dsPIC、FlashFlex、flexPWR、Heldo、JukeBlox、KeeLoq、KeeLoqロゴ、Kleer、LANCheck、LINK MD、maXStylus、maXTouch、Med iaLB、megaAVR、MOST、MOSTロゴ、MPLAB、OptoLyzer、PIC、picoPower、PICSTART、PIC32ロゴ、Prochip Designer、QTouch、Rig htTouch、SAM-BA、SpyNIC、SST、SSTロゴ、SuperFlash、tinyAVR、UNI/O、XMEGAは米国と他の国に於けるMicrochip Technology Incorporatedの登録商標です。

ClockWorks、The Embedded Control Solutions Company、EtherSynch、Hyper Speed Control、HyperLight Load、IntelliMOS、mTou ch、Precision Edge、Quiet-Wire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、BodyCom、chipKIT、chipKITロゴ、C odeGuard、CryptoAuthentication、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、 DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet¤ ゴ、Mindi、MiWi、motorBench、MPASM、MPF、MPLAB Certifiedロゴ、MPLAB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PureSilicon、QMatrix、RightToucpロゴ、REAL ICE、Ripple Blocker、SAM-IC E、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、View Sense、WiperLock、Wireless DNA、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。

Silicon Storage Technology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。

© 2018年、Microchip Technology Incorporated、米国印刷、不許複製

AMBA、Arm、Arm7、Arm7TDMI、Arm9、Arm11、Artisan、big.LITTLE、Cordio、CoreLink、CoreSight、Cortex、DesignStart、DynamI Q、Jazelle、Keil、Mali、Mbed、Mbed Enabled、NEON、POP、RealView、SecurCore、Socrates、Thumb、TrustZone、ULINK、ULINK 2、ULINK-ME、ULINK-PLUS、ULINKpro、µVision、Versatileは米国や他国に於けるARM limited(またはその子会社)の登録商標または商標です。

# 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、アリゾナ州のチャントラーとテンヘ。、オレゴン州グ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符号飛び回りデバイス、直列EEPROM、マイクロ周辺機能、不揮発性メモリ、アナログ製品用です。加えて、開発システムの設計と製造のためのMic rochip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。

![](_page_61_Picture_0.jpeg)

# 世界的な販売とサービス

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 ウェブ<sup>\*</sup>アトレス: www.microchip.com

米国

**アトラン9** 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

**\$`7X** 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

**1)7<sup>\*</sup>7†π<sup>°</sup>リス** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380

#### ロサンセ゛ルス

Mission Viejo, CA Tel: 949–462–9523 Fax: 949–462–9608 Tel: 951–273–7800 **D–IJ–** NC Tel: 919–844–7510

**⊐⊐−∃−**7 NY Tel: 631−435−6000

**サン**ホセ CA Tel: 408-735-9110 Tel: 408-436-4270 **カナダ<sup>・</sup> - トロント** 

Tel: 905-695-1980 Fax: 905-695-2078 亜細亜/太平洋 **オーストラリア - シト・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

### 亜細亜/太平洋

イント - ハンガロール Tel: 91-80-3090-4444 イント - ニューデリー Tel: 91-11-4160-8631 イント・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 ー ヘ・ナン Tel: 60-4-227-8870 フィリピン ー 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 ー バンコク Tel: 66-2-694-1351 ベトナム ー ホーチミン Tel: 84-28-5448-2100

欧州 オーストリア – ウェルス Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 テンマーク - コヘンハーケン Tel: 45-4450-2828 Fax: 45-4485-2829 フィンラント – エスホー 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 トイツ – ガルヒング Tel: 49-8931-9700 ドイツ – ハーン Tel: 49-2129-3766400 トイツ – ハイルフロン Tel: 49-7131-67-3636 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 - ローセンハイム Tel: 49-8031-354-560 イスラエル - ラーナナ Tel: 972-9-744-7705 イタリア ー ミラノ Tel: 39-0331-742611 Fax: 39-0331-466781 イタリア ー パドバ Tel: 39-049-7625286 オランダーデルーネン Tel: 31-416-690399 Fax: 31-416-690340 ノルウェー - トロンハイム Tel: 47-7289-7561 ポーラント゛ー 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 - マドリート Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 スウェーデン – イェーテホリ Tel: 46-31-704-60-40 スウェーデン – ストックホルム Tel: 46-8-5090-4654 イキ・リス - ウォーキンガム

Tel: 44-118-921-5800Fax: 44-118-921-5820