書き込み器とﾃﾞﾊﾞｯｶﾞ

Power Debugger

使用者の手引き

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、Atmel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。

Rev. 42696D-10/2016, 42696DJ1-08/2020

目次

1. Atmel Power Debugger

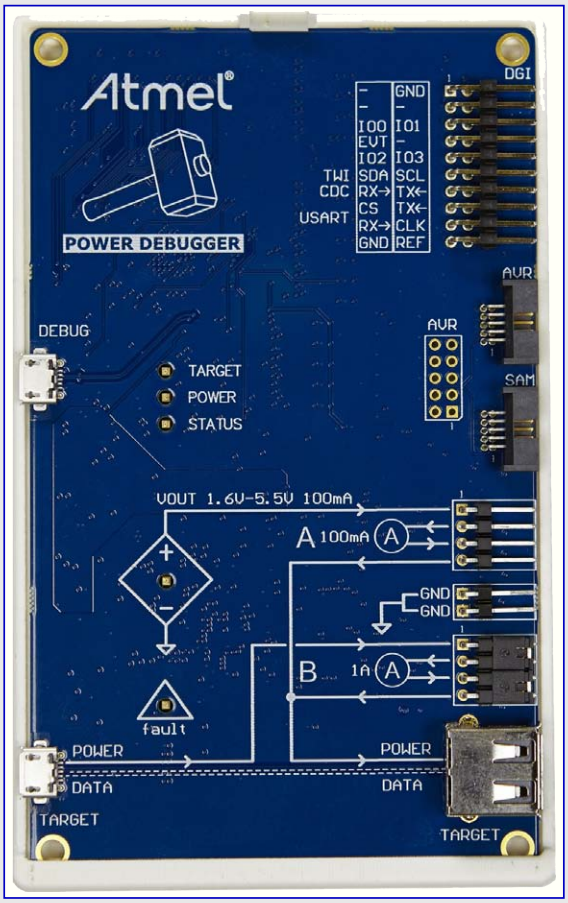
Atmel® Power DebuggerはJTAG、SWD、PDI、UPDI、ﾃﾞﾊﾞｯｸﾞWIRE、aWire、TPI、またはSPI目的対象ｲﾝﾀｰﾌｪｰｽを使うARM® Cortex® - Mに基づくAtmel SAMとAtmel AVR®のﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞ用の強力な開発ﾂｰﾙです。

加えてAtmel Power Debugger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。

Atmel Power DebuggerはまたCDC仮想COMﾎﾟｰﾄ ｲﾝﾀｰﾌｪｰｽだけでなく、SPI、USART、TWI、または汎用入出力(GPIO)の供給元からﾎｽﾄ ｺﾝﾋﾟｭｰﾀに応用ﾃﾞｰﾀを流すためのAtmelﾃﾞｰﾀ交換器ｲﾝﾀｰﾌｪｰｽ(DGI:Data Gateway Interface)も含みます。

Atmel Sudioはhttp://www.atmel.com/tools/atmelstudio.aspxから、Atmelﾃﾞｰﾀ可視器(Data Visualizer)はhttps://gallery.atmel.com/Products/Details/2f6059f5-9200-4028-87e1-ba3964e0acc2からﾀﾞｳﾝﾛｰﾄﾞすることができます。

Power DebuggerはAtmel Studio 7.0またはそれ以降、または標準CMSIS-DAP単位部に接続する能力がある他の前処理ｿﾌﾄｳｪｱと動くCMSIS-DAP互換ﾃﾞﾊﾞｯｶﾞです。Power Debuggerは実時間分析のために電力測定と応用ﾃﾞﾊﾞｯｸﾞのﾃﾞｰﾀをAtmelﾃﾞｰﾀ可視器に流します。



1.1. ｷｯﾄ内容

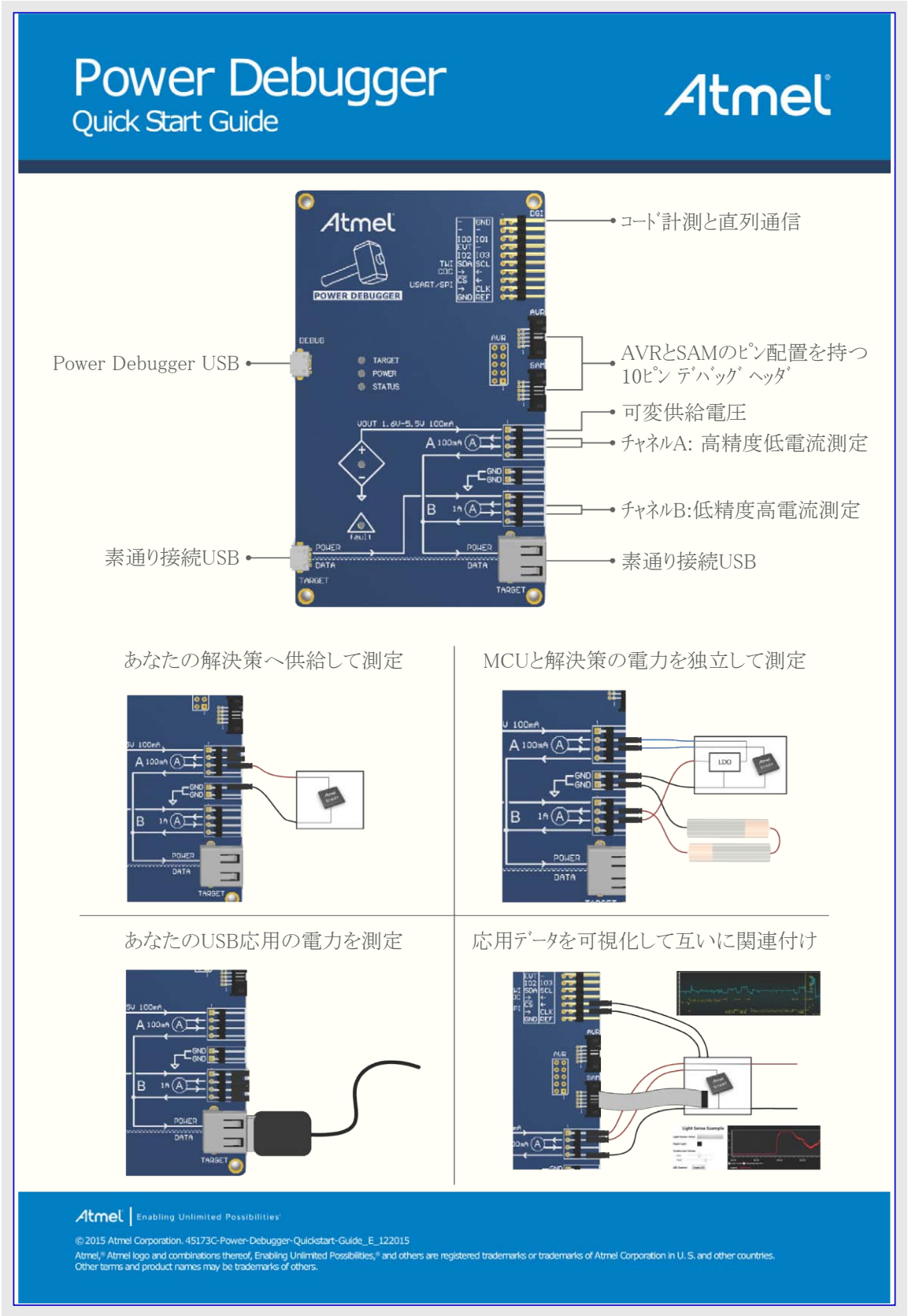
Power Debuggerｷｯﾄは以下を含みます。

- ﾌﾟﾗｽﾃｨｯｸの背面を持つ1つのPower Debugger本体
- 2つのUSBｹｰﾌﾞﾙ (1.5m、高速(HS)、ﾏｲｸﾛB)
- 50mil AVR、100mil AVR/SAM、100mil 20ﾋﾟﾝSAMｱﾀﾞﾌﾟﾀを含む1つのｱﾀﾞﾌﾟﾀ基板
- 10ﾋﾟﾝ50milと6ﾋﾟﾝ100milのｺﾈｸﾀを持つ1つのIDCﾌﾗｯﾄ ｹｰﾌﾞﾙ
- 10×100milｿｹｯﾄを持つ1つの50mil10ﾋﾟﾝ ﾊﾞﾗ線ｹｰﾌﾞﾙ
- 20×100milの分離可能な接続ｹｰﾌﾞﾙ

図1-1. Power Debuggerｷｯﾄ内容

2. Power Debuggerでの開始に際して

Power Debuggerで開始するのは、特にAtmel-ICE書き込み器/ﾃﾞﾊﾞｯｶﾞやAtmel Xplained Proｷｯﾄの何れかに精通していれば簡単です。



助言: ここで示される即時開始の手引きは利便性のためにｷｯﾄに含まれます。

使用事例章はいくつかの筋書きで電力消費分析を使うのにPower Debuggerがどう置かれ得るかの記述を含みます。Power Debuggerに対する素早く簡単な紹介についてはあなたの現在の計画や利用可能な基板に最も近くて合う使用事例を選んでください。

ﾊｰﾄﾞｳｪｱ説明章は独自基板で行うことを望む使用者に対してPower Debuggerの能力の技術的な詳細を提供します。

関連ﾘﾝｸ　55頁の「ﾊｰﾄﾞｳｪｱ説明」

3. Power Debuggerの接続

Power Debuggerの多くの接続の可能性は殆どの接続に対する機能的な参照として扱うｼﾙｸ ｽｸﾘｰﾝと共に論理的な規則で配置されます。接続概要は以下のようにここで与えられます。

USB ‘DEBUG’ｺﾈｸﾀ：ﾃﾞﾊﾞｯｶﾞの電源と全てのﾃﾞｰﾀ転送用のﾎｽﾄ ｺﾝﾋﾟｭｰﾀに接続されなければなりません。

USB ‘TARGET’ｺﾈｸﾀ：任意選択で目的対象にUSB電力を供給するのに使うことができます。ﾃﾞｰﾀ線は対応する’TARGET’A型ｼﾞｬｯｸへ素通りされます。利便性のためにUSB給電される応用のﾃﾞﾊﾞｯｸﾞ時に、目的対象USBｺﾈｸﾀへの出力は、ｼﾙｸ ｽｸﾘｰﾝの描画によって示されるように、ｼﾞｬﾝﾊﾟを使って’A’または’B’のどちらかのﾁｬﾈﾙに容易に接続することができます。

DGI 20ﾋﾟﾝ ﾍｯﾀﾞ列：任意選択でｺﾝﾋﾟｭｰﾀへﾃﾞｰﾀを流すために目的対象のﾃﾞｰﾀ供給元に接続することができます。

AVRとSAMのﾃﾞﾊﾞｯｸﾞ ﾍｯﾀﾞ：AVR 10ﾋﾟﾝ配列(または関連ｱﾀﾞﾌﾟﾀ経由)または10ﾋﾟﾝARM Cortexﾃﾞﾊﾞｯｸﾞ配列のどちらかを使って目的対象ﾃﾞﾊﾞｲｽに接続するためのﾍｯﾀﾞです。

VOUT：任意選択で、1.6～5.5Vの範囲でPower Debuggerから目的対象へ電力を提供します。

GND：Power Debuggerの安全で正確な動作を保証するため、共有接地が不可欠です。

‘A’と‘B’のﾁｬﾈﾙの電流測定ﾎﾟｰﾄ：ｼﾙｸ ｽｸﾘｰﾝの電流計記号で描かれたこれら2つの測定ﾁｬﾈﾙは高い側での構成設定で接続されます。これは供給電圧が電流計の入力に接続され、負荷(目的対象)が出力に接続されることを意味します。

情報: それらの隣接ﾋﾟﾝ関して’A’と’B’のﾁｬﾈﾙの入力と出力の信号の分類は純粋に利便性のためです。構成設定可能なVOUT電圧源をﾁｬﾈﾙ’A’に配線するのにｼﾞｬﾝﾊﾟを使うことができます。この電圧は配線を使ってﾁｬﾈﾙ’B’に配線することができます。同じことはTARGET USBｺﾈｸﾀによって提供される電圧にも適用されます。

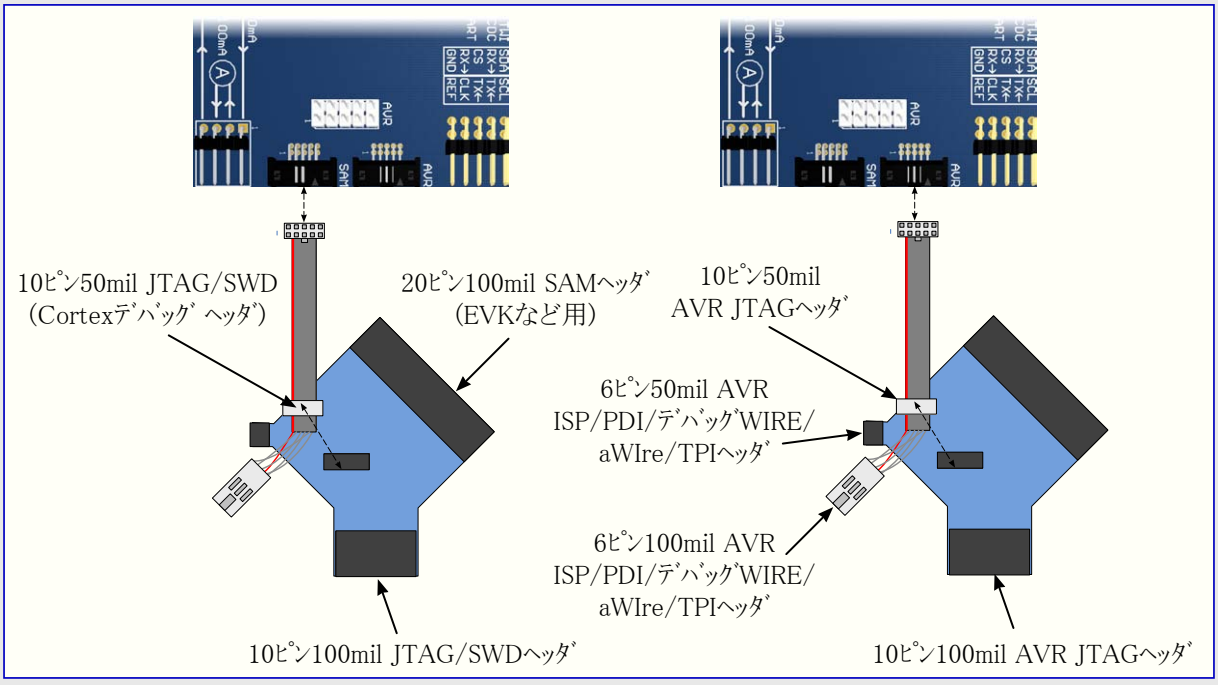
助言: 一般的な電力配線構成についてはｷｯﾄに含まれる印刷された即時開始の手引きと開始に際して章を参照してください。

3.1. AVRとSAMの目的対象ﾃﾞﾊﾞｲｽへの接続

Power Debuggerは2つの50mil 10ﾋﾟﾝJTAGｺﾈｸﾀが装備されます。両ｺﾈｸﾀは電気的に直接接続されますが、2つの異なるﾋﾟﾝ配列、AVR JTAGﾍｯﾀﾞとARM Cortexﾃﾞﾊﾞｯｸﾞ ﾍｯﾀﾞに従います。ｺﾈｸﾀは目的対象MCUの形式ではなく目的対象基板のﾋﾟﾝ配列に基づいて選ばれるべきで、例えば、AVR STK®600階層に実装されたSAMﾃﾞﾊﾞｲｽはAVRﾍｯﾀﾞを使うべきです。

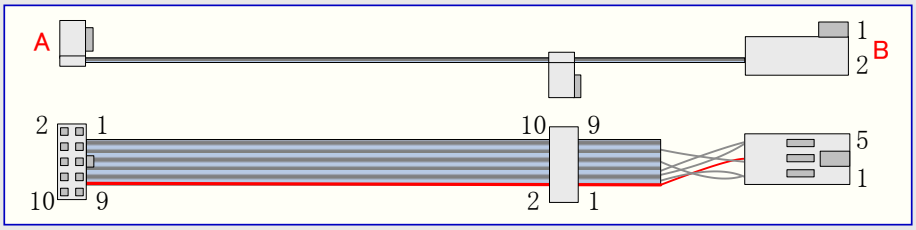
Power Debuggerｷｯﾄは必要な全てのｹｰﾌﾞﾙとｱﾀﾞﾌﾟﾀを含みます。接続任意選択の概要が示されます。

図3-1. Power Debugger接続任意選択



赤線は10ﾋﾟﾝ50milｺﾈｸﾀの1番ﾋﾟﾝを記します。6ﾋﾟﾝ100milｺﾈｸﾀの1番ﾋﾟﾝはｺﾈｸﾀがｹｰﾌﾞﾙから見られる時に鍵突起の右に配置されます。ｱﾀﾞﾌﾟﾀの各ｺﾈｸﾀの1番ﾋﾟﾝは白丸で記されます。下図はﾃﾞﾊﾞｯｸﾞ ｹｰﾌﾞﾙのﾋﾟﾝ配置を示します。Aと記されたｺﾈｸﾀはﾃﾞﾊﾞｯｶﾞに繋がれ、同時にB側は目的対象基板に繋がれます。

図3-2. ﾃﾞﾊﾞｯｸﾞ ｹｰﾌﾞﾙ ﾋﾟﾝ配置



4. 詳細な使用事例

Power Debuggerに対する使用事例の集合がここで概説されます。各使用事例は或る筋書きで与えられた問題を解決するように試みようとし、その後にPower Debuggerを用いてこれがどう達成され得るかの詳細な説明を提供します。

低電力(電池)応用

基本的な低電力応用の各種休止動作が分析されます。

USB応用

素通りUSB接続を使い、USB給電された基板の全体電流消費が測定されて分析されます。

4.1. 低電力応用

この使用事例はPower Debuggerを紹介するのにmegaAVR®ﾃﾞﾊﾞｲｽを使います。Atmel StudioとAtmelﾃﾞｰﾀ可視器(Data Visualizer)を共に道具に使い、代表的な低電力応用の電力消費を測定、分析、理解し、そして最適化するのにそれをどう使うのかを学びます。ﾃﾞｰﾀ交換器ｲﾝﾀｰﾌｪｰｽを使う計装ｺｰﾄﾞの例も考察します。

例はATmega328PB Xplained Mini基板を使いますが、どんな簡単な独自基板にも合うように凄く容易に適合することができます。

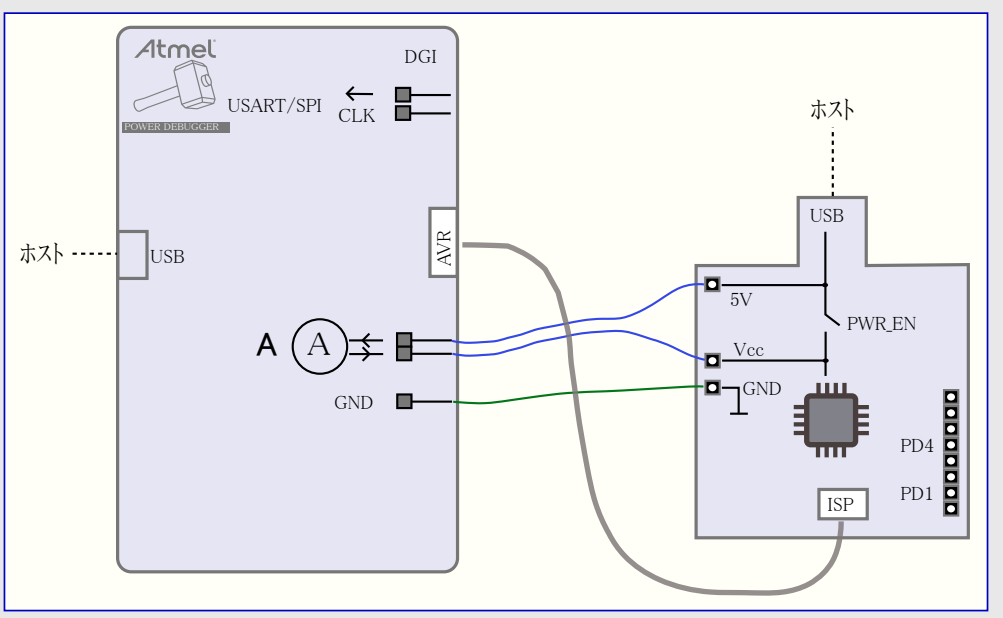
4.1.1. 必要条件

この例を通して動くようにできるには以下が必要とされます。

- (Atmelﾃﾞｰﾀ可視器(Data Visualizer)が含まれる)ｲﾝｽﾄｰﾙされたAtmel Studio 7(またはそれ以降)を持つﾎｽﾄ ｺﾝﾋﾟｭｰﾀ
- (ｹｰﾌﾞﾙを含む)Atmel Power Debuggerｷｯﾄ
- Atmel ATmega328PB Xplained Miniｷｯﾄまたは同様の目的対象基板
- ﾋﾟﾝ ﾍｯﾀﾞ: 2×3 100mil ISPﾍｯﾀﾞ 1個、1×8 100milﾍｯﾀﾞ 2個
- 基本的な半田付け装備の利用

4.1.2. 初期ﾊｰﾄﾞｳｪｱ準備

初期構成の構成図がここで示されます。



開始するにはﾊｰﾄﾞｳｪｱに幾許かの変更を行うことが必要です。

目的対象MCUによって消費される電流を測定するために、それの電源を横取りしてPower Debuggerの測定回路を通してそれを供給することが必要です。これは電源線ﾍｯﾀﾞがXplained Mini基板に実装されることを必要とします。

すべきこと: Xplained Mini基板で電源線ﾍｯﾀﾞを実装してください。

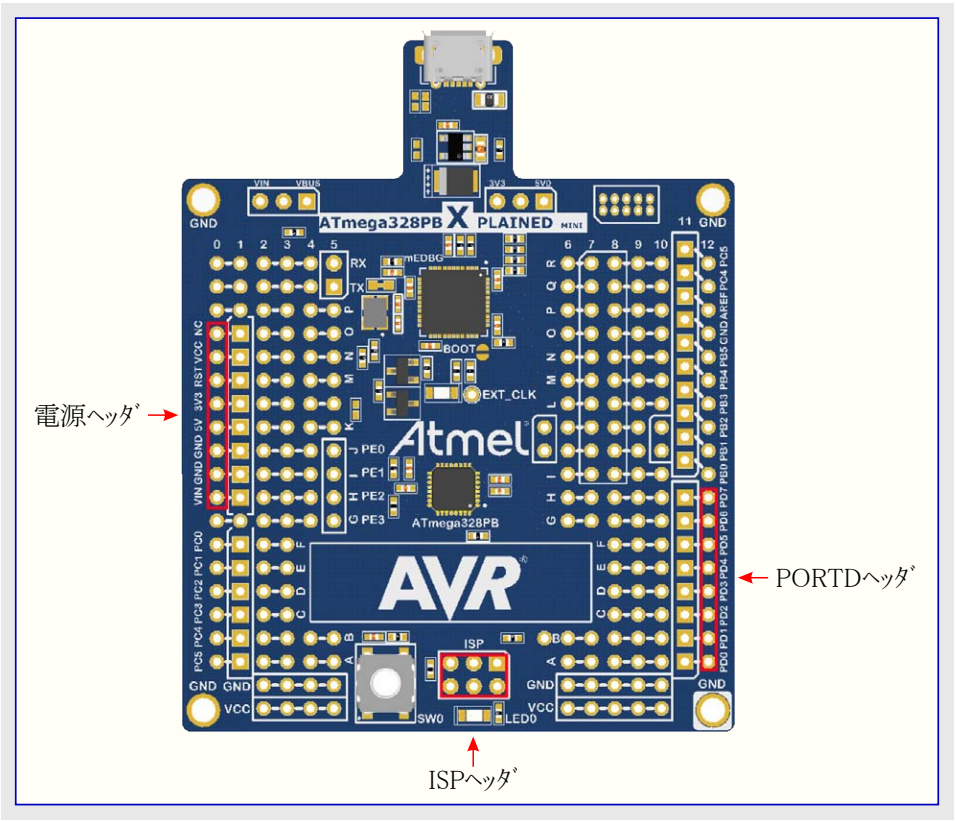
Xplained Mini基板からPower DebuggerのDGIにﾃﾞｰﾀを送るのに計装ｺｰﾄﾞを使います。これを行うには目的対象MCUでPORTD機能への入出力が必要です。

すべきこと: Xplained Mini基板でPORTD7～0ﾍｯﾀﾞを実装してください。

ATmega328PB Xplained Mini基板が既にﾃﾞﾊﾞｯｶﾞ(mEDBG)を含むとは言え、この例では使われません。代わりにむしろﾌﾟﾛｸﾞﾗﾐﾝｸﾞ目的のためにPower Debuggerを利用します。

すべきこと: Xplained Mini基板で6ﾋﾟﾝ100mil ISPﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾍｯﾀﾞを実装してください。

実装されるべきﾍｯﾀﾞがここで示されます。



4.1.3. 接続

一旦ﾊｰﾄﾞｳｪｱ変更が行われてしまうと、目的対象にPower Debuggerを接続する準備が整います。

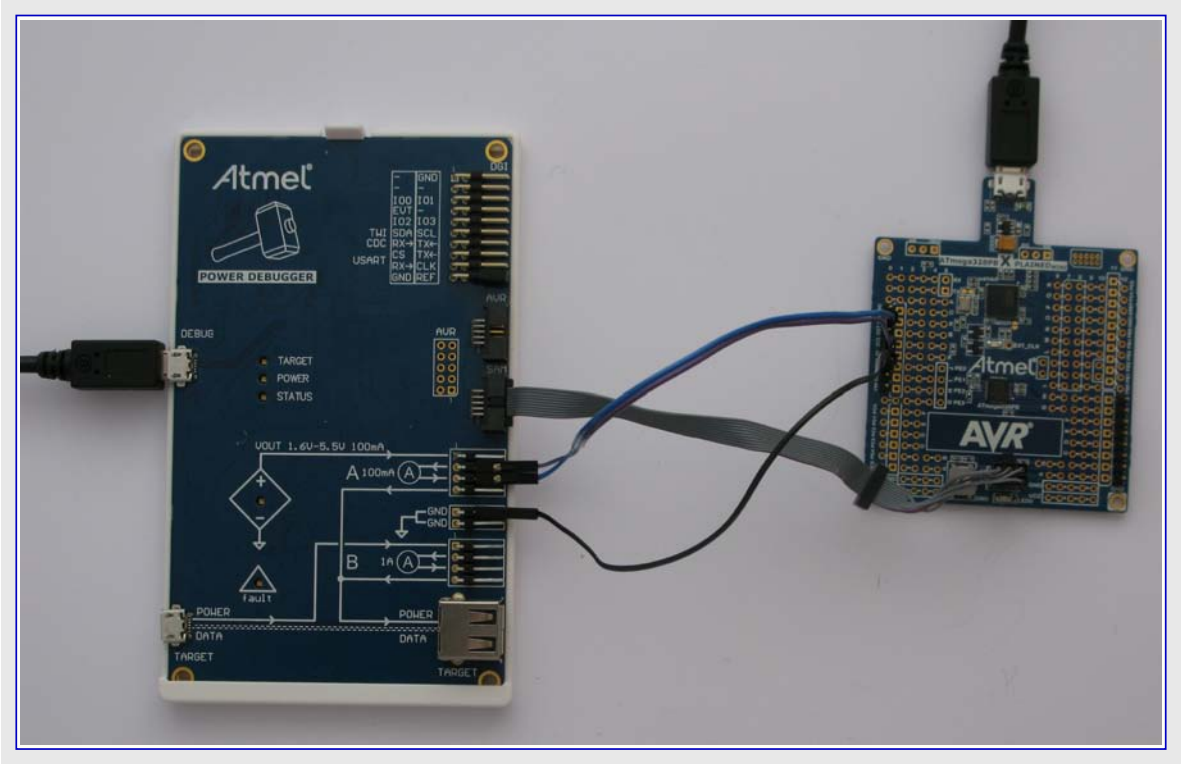
ATmega328PBﾃﾞﾊﾞｲｽの電流を測定するため、それの(電源)供給をPower Debuggerを通して配線しなければなりません。FET切り替え器のOFF設定でATmega328PBﾃﾞﾊﾞｲｽは目的対象電源を持ちません。Aﾁｬﾈﾙを通して5V Xplained Mini電圧をATmega328PBﾃﾞﾊﾞｲｽの電源供給線に配線するには以下を行ってください。

すべきこと:
- Xplained Mini基板の電源ﾍｯﾀﾞの5VﾋﾟﾝをPower DebuggerのAﾁｬﾈﾙの入力ﾋﾟﾝに接続してください。
- Aﾁｬﾈﾙの出力ﾋﾟﾝをXplained Miniの電源ﾍｯﾀﾞのVCCﾋﾟﾝに接続してください。

USBとﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ｹｰﾌﾞﾙを接続してください。

すべきこと:
- Power DebuggerのAVR出力からのﾃﾞﾊﾞｯｸﾞ ｹｰﾌﾞﾙをXplained Mini基板のISPﾍｯﾀﾞに接続してください。
- Power DebuggerとXplained Miniの両方をﾎｽﾄ ｺﾝﾋﾟｭｰﾀのUSBﾎﾟｰﾄに差し込んでください。

構成はこのようなものに見えるでしょう。



目的対象ﾃﾞﾊﾞｲｽのﾃﾞﾊﾞｯｸﾞWIREｲﾝﾀｰﾌｪｰｽが許可されている場合、禁止されることが必要です。

助言: 電源OFF/ON後にISPを使ってﾃﾞﾊﾞｲｽ識票を読むことができるなら、ﾃﾞﾊﾞｯｸﾞWIREが成功裏に禁止されてDWENは解除(1)されています。

4.1.4. ﾃﾞﾊﾞｯｸﾞWIREｲﾝﾀｰﾌｪｰｽの禁止

ﾃﾞﾊﾞｯｸﾞWIREの禁止は以下のどちらかによって行うことができます。

- ﾃﾞﾊﾞｯｸﾞ作業中にDebug(ﾃﾞﾊﾞｯｸﾞ)ﾒﾆｭｰから"Disable DebugWIRE and close(ﾃﾞﾊﾞｯｸﾞWIREを禁止して閉じる)"を選択、または
- Atmel Studioｺﾏﾝﾄﾞ行ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾕｰﾃｨﾘﾃｨのatprogram.exeを次のように使い、

```
atprogram.exe -t powerdebugger -i debugwire -d atmega328pb dwdisable
```

その後、Atmel Studioのﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾀﾞｲｱﾛｸﾞで、または以下のようにatprogram.exeを使ってのどちらかで、DWENを解除するのにISPを使ってください。

- 上位ﾋｭｰｽﾞ(変位(ｵﾌｾｯﾄ)1)を読んでください。

```
atprogram.exe -t powerdebugger -i isp -d atmega328pb read -fs -s 1 -o 1 --format hex
```

- または、$40(DWENはﾋﾞｯﾄ6)の出力値で
- 上位ﾋｭｰｽﾞ(変位(ｵﾌｾｯﾄ)1)に値を書き戻してください。

```
atprogram.exe -t powerdebugger -i isp -d atmega328pb write -fs -o 1 --values <新しいﾋｭｰｽﾞ値>
```

注意: 目的対象ﾃﾞﾊﾞｲｽでのﾋｭｰｽﾞ書き込み時に細心の注意を払ってください。不正なﾋｭｰｽﾞ変更はXplained Miniｷｯﾄを定常的な使用不能に終わらせるかもしれません。

4.1.5. Xplained Miniの基板上電源の禁止

Xplained MIni基板上のmEDBGﾃﾞﾊﾞｯｶﾞはそれの電力を切り替えすることができるように目的対象ﾃﾞﾊﾞｲｽのVCCの制御を持ちます。この切り替え器を開放することにより、使用者は外部電流測定探針を通して目的対象ﾃﾞﾊﾞｲｽに電力を再配線することができます。電源投入での既定によって切り替え器は閉じられます(電力がON)。

切り替え器を開くにはAtmel Studioｺﾏﾝﾄﾞ行ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾕｰﾃｨﾘﾃｨのatprogram.exeを次のように使ってください。

```
atprogram.exe -t medbg parameters -psoff
```

重要: mEDBGから目的対象ﾃﾞﾊﾞｲｽへのｸﾛｯｸ元は活性に留まり、故に目的対象ﾃﾞﾊﾞｲｽは入出力漏れを通して部分的に給電されるかもしれません。

切り替え器を再び閉じるには次を実行してください。

```
atprogram.exe -t medbg parameters -pson
```

助言: より古いmEDBGﾌｧｰﾑｳｪｱは電源ON要求を支援していません。その場合には目的対象電力を回復するためにUSB電力をOFF/ONしてください。

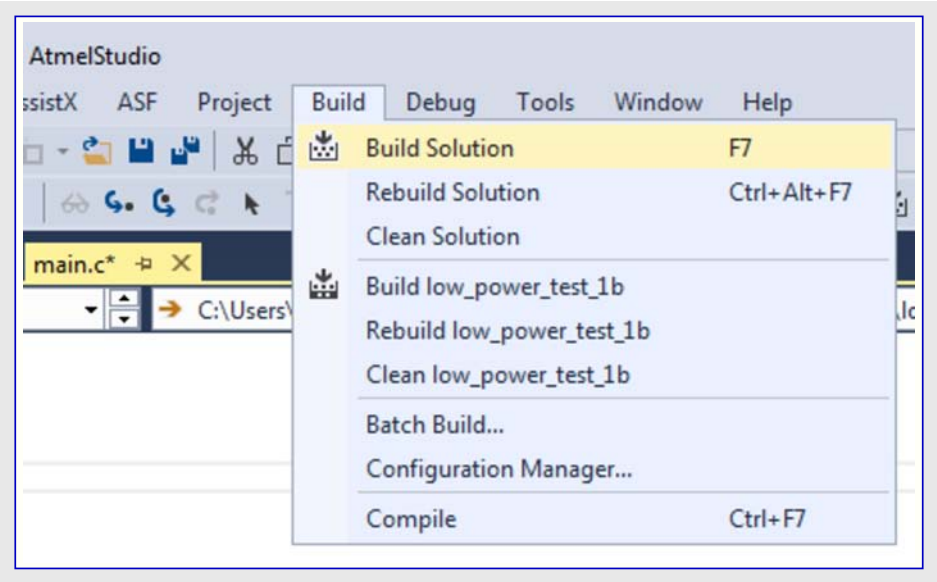
4.1.6. 簡単な電流測定での開始

Xplained MIni基板上のLEDを点滅する応用を書くことによって始めましょう。大きな計数器の内側のNOP命令を持つ繰り返し遅延を使って約1%のﾃﾞｭｰﾃｨ ｻｲｸﾙでLEDを脈動してください。low_power_101用ｺｰﾄﾞはここで示されます。

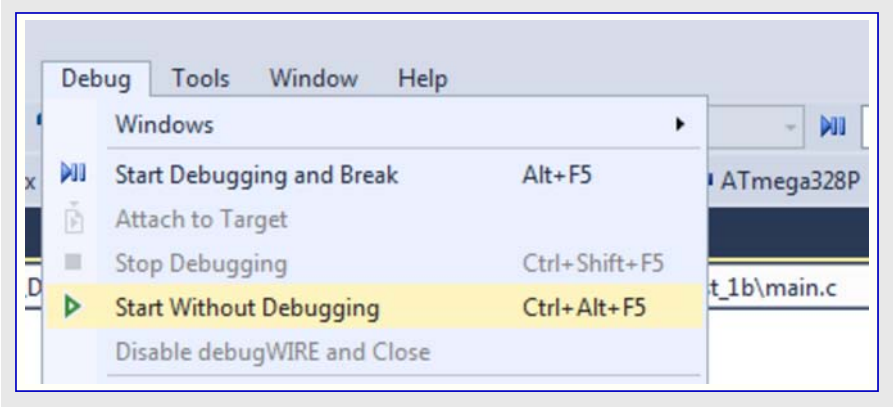
```
#include <avr/io.h>

void delay (uint16_t length)
{
    // 単純な繰り返し遅延
    for (uint16_t i=0; i<length; i++) {
        for (uint8_t j=0; j<255; j++) {
            asm volatile("nop");
        }
    }
}

int main(void)
{
    DDRB = (1 << 5);            // PORTB5を出力に
    // 永久に実行
    while (1) {
        PORTB = (1 << 5);       // PORTB5をON
        delay(50);              // 短い遅延
        PORTB = 0x00;           // PORB5をOFF
        delay(5000);            // 長い遅延
    }
}
```

重要: この例に対してはToolchain(ﾂｰﾙﾁｪｰﾝ)⇒AVR/GNU C Compiler(AVR/GNU Cｺﾝﾊﾟｲﾗ)⇒Optimization(最適化)下のﾌﾟﾛｼﾞｪｸﾄ任意選択で最適化水準をNone (-O0) (なし)に設定してください。

すべきこと: ﾌﾟﾛｼﾞｪｸﾄ/解決策を構築(F7)してください。



すべきこと: Start Without Debugging(ﾃﾞﾊﾞｯｸﾞなしで開始) (Ctrl+Alt+F5)を選ぶことによって応用を目的対象に書き込んでください。



mEDBG上のLED0が今や点滅を開始するでしょう。

情報: この例ではﾌﾟﾛｸﾞﾗﾐﾝｸﾞ動作だけを使うつもりです。ﾃﾞﾊﾞｯｶﾞを通してｺｰﾄﾞを走行する殆どのｼｽﾃﾑでは正確な電流測定を提供しません。これは目的対象のﾃﾞﾊﾞｲｽのﾃﾞﾊﾞｯｸﾞ単位部(OCD)がﾃﾞﾊﾞｯｸﾞ中に禁止することができないｸﾛｯｸ元を必要とするからです。

重要: Xplained Miniの基板上電力を禁止することを忘れないでください。

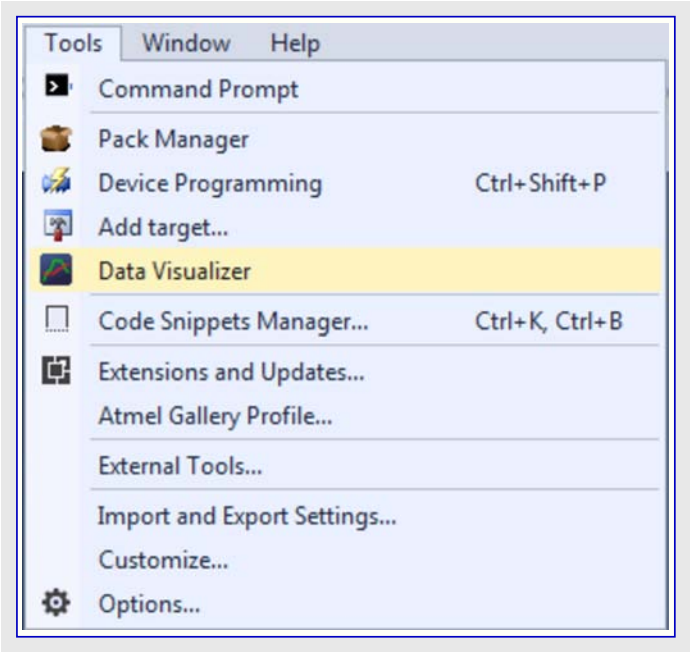
4.1.7. Atmelﾃﾞｰﾀ可視器の起動

Atmelﾃﾞｰﾀ可視器(Data Visualizer)はAtmel Studioの一部として含まれ、Atmel Studio拡張として、または独立動作形態でのどちらでも動かすことができます。

Atmel Studio内の拡張としてAtmelﾃﾞｰﾀ可視器を動かすにはTools(ﾂｰﾙ)ﾒﾆｭｰでそれを選んでください。

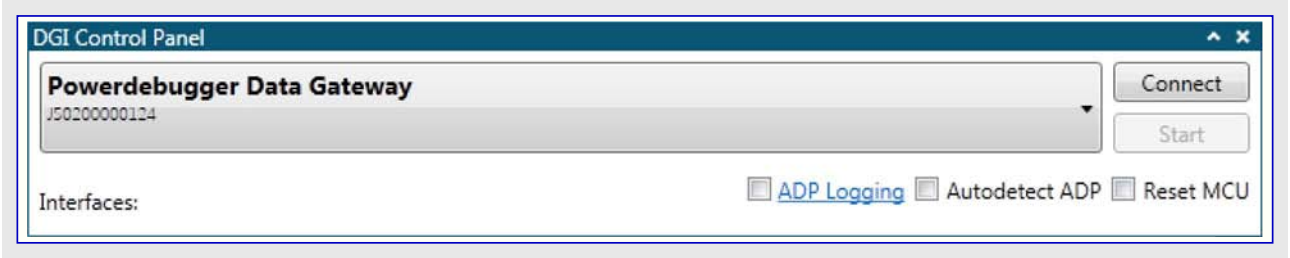
ﾃﾞｰﾀ可視器機能を支援するｷｯﾄはAtmel Studio内のそれらの開始頁でその拡張へのｼｮｰﾄｶｯﾄを含みます。

独立版のAtmelﾃﾞｰﾀ可視器がｲﾝｽﾄｰﾙされた場合、Windows®のｽﾀｰﾄ ﾒﾆｭｰでｼｮｰﾄｶｯﾄを探してください。独立版はgallery.atmel.comからのﾀﾞｳﾝﾛｰﾄﾞで入手可能です。

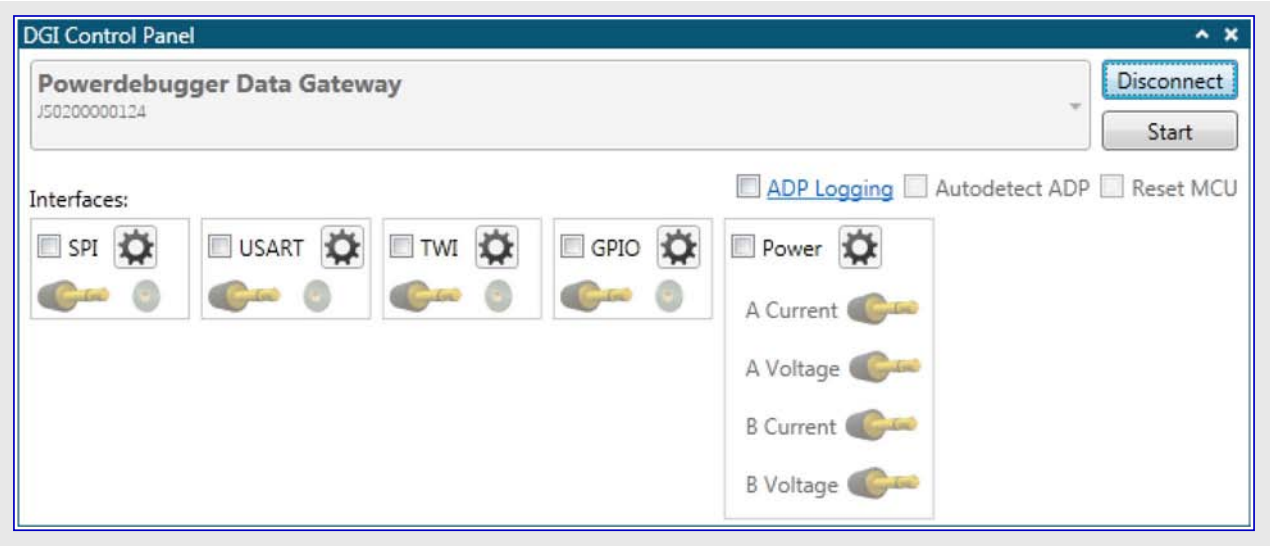


4.1.8. 基本的な電流測定

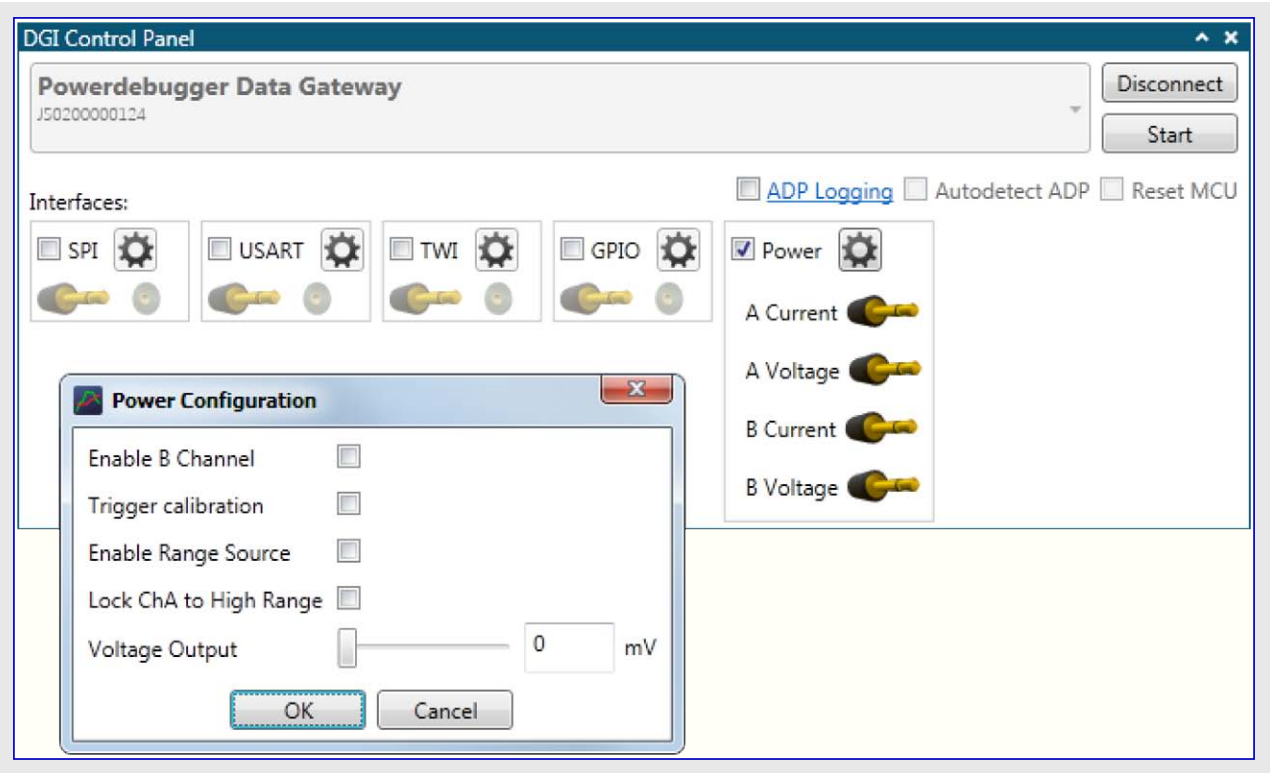
すべきこと: DGI Control Panel(DGI制御盤)でﾂｰﾙを選んでください。



すべきこと: 選んだツールのDGIに接続(Connect)してください。



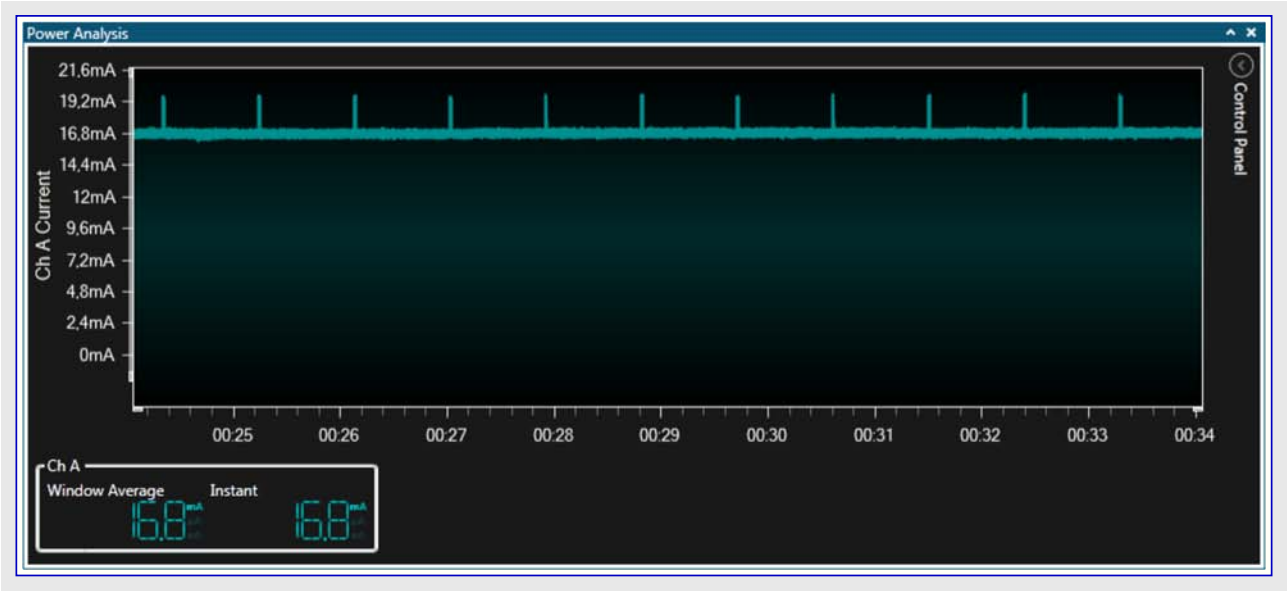
すべきこと: Power(電力)インターフェースを許可して関連チャネルを監視するようにそれの設定を変更してください。



すべきこと: データ可視器作業を開始(Start)してください。

4.1.9. LED点滅

数秒間のﾃﾞｰﾀ可視器の走行は以下のこれと同様の図を与えるでしょう。



この図から何を知ることができますか?。

- ｷｯﾄはLED OFFで約17mA引き出します。
- 電流引き出しはLEDがONに躍動される時に約20mAに増加します。
- 1%のﾃﾞｭｰﾃｨ ｻｲｸﾙは概ね正しいと思われます。

図がこのように見えない場合、戻って以下に対する構成設定を調べてください。

- Xplained Miniへの電源 (USBｹｰﾌﾞﾙ)
- VCCﾍｯﾀﾞから'A'ﾁｬﾈﾙへの接続線
- 共通GND接続
- Xplained Miniで基板上電力が禁止されていますか?。
- LEDが点滅しますか?。書き込みは成功しましたか?。
- ﾃﾞﾊﾞｯｸﾞWIRE(DWEN)は禁止されていますか?。
- Power Debuggerの'Fault(障害)'LEDがONの場合、もう一度、配線と半田付けを調べてください。

4.1.10. ｸﾛｯｸ周波数の低減

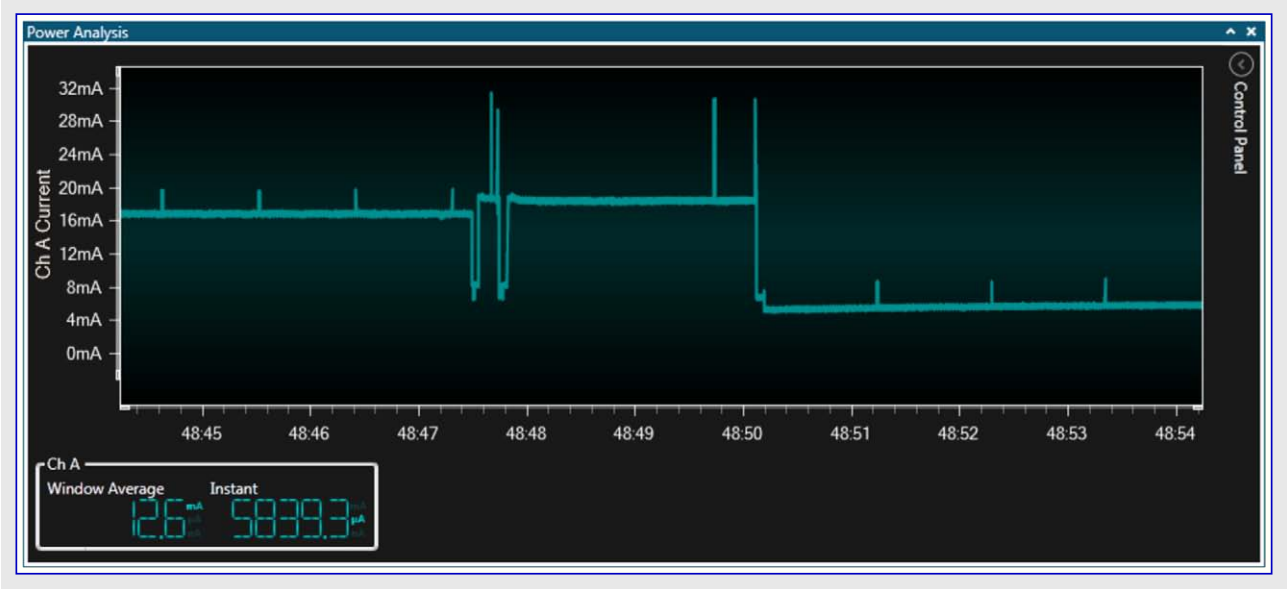
さて、応用の電力消費をどう減らすことができるかを考察して、それが改善されることを確認しましょう。

最初の考えは繰り返し遅延を除去して計時器割り込みのLED点滅部を動かすことかもしれません。加えて、このような簡単な例は高速で動かす必要がなく、故により遅く動かすのにｸﾛｯｸ前置分周器を使うことができます。下のｺｰﾄﾞはlow_power_102ﾌﾟﾛｼﾞｪｸﾄに含まれます。

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

// 計時器0割り込み処理ルーチン
ISR (TIMER0_OVF_vect)
{
    if (PORTB == 0x00) {                    // LEDがOFFなら、
        PORTB = (1 << 5);                   // それをONに切り替え
        TCNT0 = 0x102;                      // ON周回用制限時間を短くする。
    }
    else
    {                                       // LEDがONなら、
        PORTB = 0x00;                       // それをOFF切り替え
    }
}

int main(void)
{
    CLKPR = (1 << CLKPCE);                  // ｸﾛｯｸ前置分周器変更許可
    CLKPR = (1 << CLKPS2) | (1 << CLKPS1) | (0 << CLKPS0);    // 64分周(DIV64)に調整
```

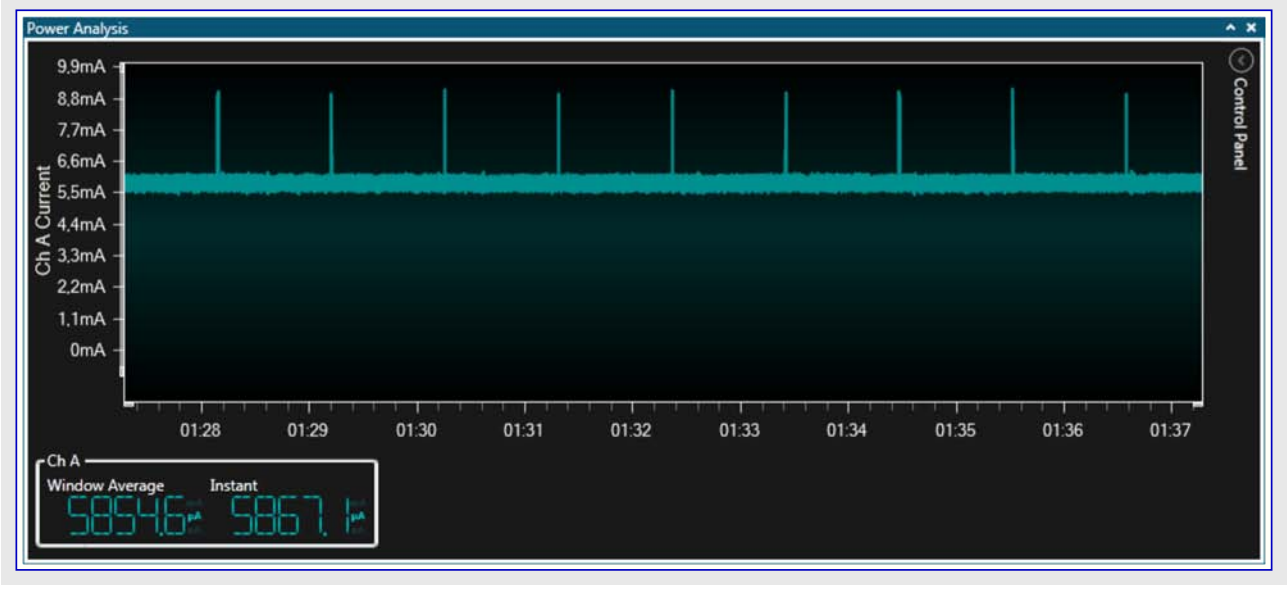
```
    DDRB = (1 << 5);                        // ﾎﾟｰﾄB5を出力に設定
    TCCR0B = (1 << CS02) | (1 << CS00);     // 計時器0を1024分周(DIV1024)で開始
    TIMSK0 = (1 << TOIE0);                  // 溢れ割り込み許可
    sei();                                  // 全体割り込み許可
    while (1);                              // 何もしない。
}
```

すべきこと:
- ﾌﾟﾛｼﾞｪｸﾄ/解決策を構築(F7)してください。
- Start Without Debugging(ﾃﾞﾊﾞｯｸﾞなしで開始) (Ctrl+Alt+F5)を使い、目的対象ﾃﾞﾊﾞｲｽに応用を書いてください。
- 結果を見るためにﾃﾞｰﾀ可視器(Data Visualizer)に切り替えてください。



上の図は目的対象を再書き込み中に捕獲されました。この図から何を見ることができますか?。
- 2つの負の電力ﾊﾟﾙｽが見られます(ﾃﾞﾊﾞｲｽがﾘｾｯﾄに引かれます)。
- 4つの正のﾊﾟﾙｽが見られます(ID読み込み、消去、ﾌﾗｯｼｭ ﾒﾓﾘ書き込み、ﾌﾗｯｼｭ ﾒﾓﾘ検証)。
- 新しい応用が実行を始めます。

数秒間この応用を走らせたままにすることが以下のこれと同様な図を与えるでしょう。



この図から何を見ることができますか?。
- ｷｯﾄは今やLED OFFで約6mAを引き出します。
- 電流引き出しはLEDがONに躍動される時に約9mAに増加します。
- 1%のﾃﾞｭｰﾃｨ ｻｲｸﾙは未だ概ね正しいと思われます。

結果: 電力消費はﾃﾞﾊﾞｲｽをより遅くｸﾛｯｸ駆動することによってかなり改善されました。

重要: この例がｸﾛｯｸを8MHz/64に前置分周するので、ﾃﾞﾊﾞｲｽを再書き込みしようとする時にISPﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ｸﾛｯｸは主ｸﾛｯｸの1/4以下の32kHz未満に設定されなければなりません。

4.1.11. 休止動作の使い方

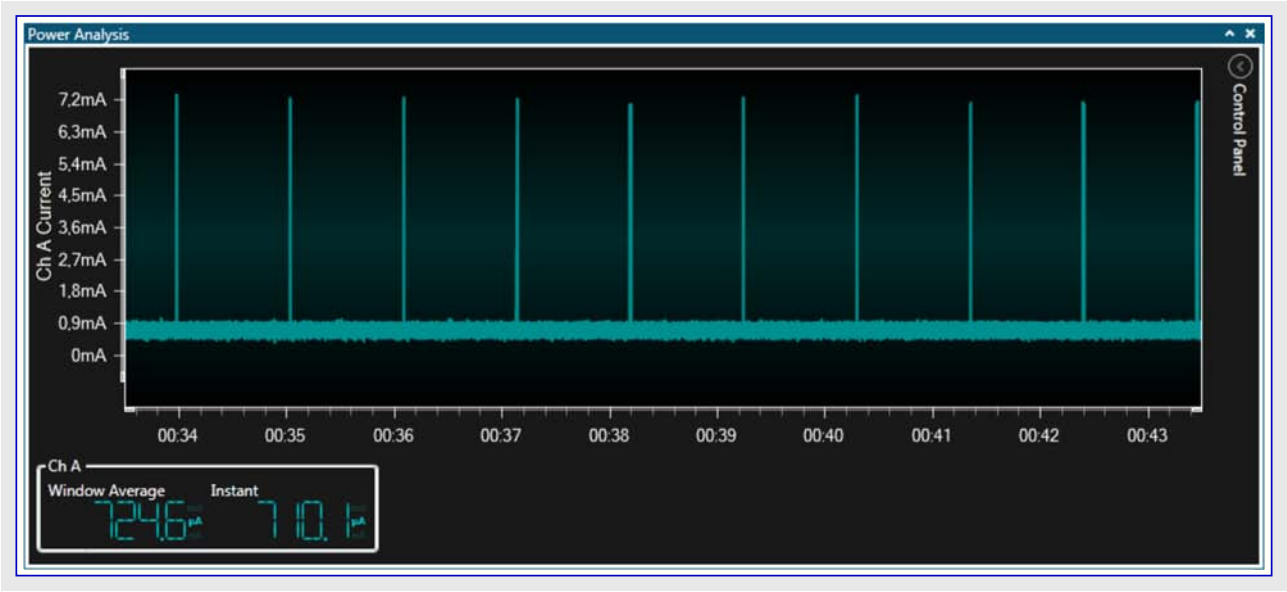
電力消費は減らされてしまいましたが、CPUは未だ積極的に何もしていません。割り込み駆動様式からの恩恵を与えるため、割り込みを待つ間、CPUを休止に置くことができます。加えて、更に電力消費を減らすために、使われない全ての周辺機能を電力停止にします。下のｺｰﾄﾞはlow_power_103ﾌﾟﾛｼﾞｪｸﾄに含まれます。

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

ISR (TIMER2_OVF_vect)
{
    if (PORTB == 0x00) {                    // LEDがOFFなら、
        PORTB = (1 << 5);                   // それをONに切り替え
        TCNT2 = 0x102;                      // ON周回用制限時間を短くする。
    }
    else {                                  // LEDがONなら、
        PORTB = 0x00;                       // それをOFF切り替え
    }
}

int main(void)
{
    DIDR0 = (1 << ADC5D) | (1 << ADC4D) | (1 << ADC3D) | (1 << ADC2D) | (1 << ADC1D) ¥
        | (1 << ADC0D);                     // ADCﾋﾟﾝのﾃﾞｼﾞﾀﾙ入力緩衝部を禁止
    DIDR1 |= (1 << AIN1D) | (1 << AIN0D);   // ｱﾅﾛｸﾞ比較器ﾋﾟﾝのﾃﾞｼﾞﾀﾙ入力緩衝部を禁止
    ACSR &= ~(1 << ACIE);                   // ｱﾅﾛｸﾞ比較器割り込みを禁止
    ACSR |= (1 << ACD);                     // ｱﾅﾛｸﾞ比較器を禁止
    // 節電のため未使用周辺機能を禁止
    ADCSRA &= ~(1 << ADEN);                 // ADC動作禁止(ADCは遮断される前に禁止されなければなりません。)
    power_adc_disable();                    // ADCを遮断(禁止)
    power_spi_disable();                    // SPIを禁止
    power_twi_disable();                    // TWIを禁止
    power_usart0_disable();                 // USART0単位部を禁止
    power_timer1_disable();                 // 計時器1単位部を禁止
    power_timer0_disable();                 // 計時器0単位部を禁止
    CLKPR = (1 << CLKPCE);                  // ｸﾛｯｸ前置分周器変更許可
    CLKPR = (1 << CLKPS2) | (1 << CLKPS1) | (0 << CLKPS0); // 64分周(DIV64)に調整
    DDRB = (1 << 5);                        // ﾎﾟｰﾄB5を出力に設定
    TCCR2B = (1 << CS22) | (1 << CS21) | (1 << CS20);       // 計時器2を1024分周(DIV1024)で開始
    TIMSK2 = (1 << TOIE2);                  // 溢れ割り込み許可
    sei();                                  // 全体割り込み許可
    while (1) {
        set_sleep_mode(SLEEP_MODE_PWR_SAVE);
        sleep_mode();
    }
}
```

すべきこと:
- ﾌﾟﾛｼﾞｪｸﾄ/解決策を構築(F7)してください。
- Start Without Debugging(ﾃﾞﾊﾞｯｸﾞなしで開始) (Ctrl+Alt+F5)を使い、目的対象ﾃﾞﾊﾞｲｽに応用を書いてください。
- 結果を見るためにﾃﾞｰﾀ可視器(Data Visualizer)に切り替えてください。

重要: 前の例がｸﾛｯｸを8MHz/64に前置分周したので、ISPﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ｸﾛｯｸは主ｸﾛｯｸの1/4以下の32kHz未満に設定されなければなりません。

重要: Xplained Miniの基板上電力を禁止することを忘れないでください。



この図から何を見ることができますか?。

- ｷｯﾄは今やLED OFFで1mA未満を引き出します。
- 電流引き出しはLEDがONに躍動される時に約7mAに増加します。
- 1%のﾃﾞｭｰﾃｨ ｻｲｸﾙは未だ概ね正しいと思われます。

結果: 電力消費は休止動作の使用と未使用周辺機能の禁止によって再び改善されました。

4.1.12. ﾊﾟﾜｰﾀﾞｳﾝ動作の使い方

電力消費は再び減らされてしまいましたが、CPUは求められるよりも未だ"より軽い"休止動作です。ﾊﾟﾜｰﾀﾞｳﾝ動作へ行くために、起こし元としてｳｫｯﾁﾄﾞｯｸﾞ ﾀｲﾏに切り替えなければなりませんｳｫｯﾁﾄﾞｯｸﾞが割り込み起動を起こす時にLEDを点灯し、それを再びOFFに切り替えるまでｱｲﾄﾞﾙ動作へ行きます。下のｺｰﾄﾞはlow_power_104ﾌﾟﾛｼﾞｪｸﾄに含まれます。

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <avr/power.h>
#include <avr/wdt.h>

volatile uint8_t deep_sleep = 0;

ISR (WDT_vect)
{
    PORTB = (1 << 5);                       // LEDがOFFなのでそれをONに切り替え
    deep_sleep = 0;                         // より軽い休止状態を合図
}

ISR (TIMER2_OVF_vect)
{
    PORTB = 0x00;                           // LEDがONなのでそれをOFFに切り替え
    deep_sleep = 1;                         // 深い休止状態を合図
}

int main(void){
    DIDR0 = (1 << ADC5D) | (1 << ADC4D) | (1 << ADC3D) | (1 << ADC2D) | (1 << ADC1D) ¥
          | (1 << ADC0D);                   // ADCﾋﾟﾝのﾃﾞｼﾞﾀﾙ入力緩衝部を禁止
    DIDR1 |= (1 << AIN1D) | (1 << AIN0D);   // ｱﾅﾛｸﾞ比較器ﾋﾟﾝのﾃﾞｼﾞﾀﾙ入力緩衝部を禁止
    ACSR &= ~(1 << ACIE);                   // ｱﾅﾛｸﾞ比較器割り込みを禁止
    ACSR |= (1 << ACD);                     // ｱﾅﾛｸﾞ比較器を禁止
```

```
    // 節電のため未使用周辺機能を禁止
    ADCSRA &= ~(1 << ADEN);                 // ADC動作禁止(ADCは遮断される前に禁止されなければなりません。)
    power_adc_disable();                    // ADCを遮断(禁止)
    power_spi_disable();                    // SPIを禁止
    power_twi_disable();                    // TWIを禁止
    power_usart0_disable();                 // USART0単位部を禁止
    power_timer1_disable();                 // 計時器1単位部を禁止
    power_timer0_disable();                 // 計時器0単位部を禁止
    // 計時器2単位部はONに留まることが必要です。
    //power_timer2_disable();               // 計時器2単位部を禁止
    CLKPR = (1 << CLKPCE);                  // ｸﾛｯｸ前置分周器変更許可
    CLKPR = (1 << CLKPS2) | (1 << CLKPS1) | (0 << CLKPS0); // 64分周(DIV64)に調整
    DDRB = (1 << 5);                        // ﾎﾟｰﾄB5を出力に設定
    wdt_reset();                            // ｳｫｯﾁﾄﾞｯｸﾞをﾘｾｯﾄ
    WDTCSR |= (1<<WDCE) | (1<<WDE);         // 時間制限変更手順開始
    WDTCSR = (1<<WDIE) | (1<<WDP2) | (1<<WDP1);     // 新前分周器(経過時間)値=64K周期(約0.5秒)に設定

    TCCR2B = (0 << CS22) | (1 << CS21) | (1 << CS20);    // 計時器2を32分周(DIV32)で開始
    TIMSK2 = (1 << TOIE2);                  // 溢れ割り込みを許可

    sei();                                  // 全体割り込みを許可

    while (1)
    {
        if (deep_sleep)                     // 深い休止なら、
            set_sleep_mode(SLEEP_MODE_PWR_DOWN); // ﾊﾟﾜｰﾀﾞｳﾝ動作へ
        else {                              // LED ONのｱｲﾄﾞﾙ動作でより短い休止
            TCNT2 = 0;
            set_sleep_mode(SLEEP_MODE_IDLE);
        }
        sleep_mode();
    }
}
```

すべきこと:
- ﾌﾟﾛｼﾞｪｸﾄ/解決策を構築(F7)してください。
- Start Without Debugging(ﾃﾞﾊﾞｯｸﾞなしで開始) (Ctrl+Alt+F5)を使い、目的対象ﾃﾞﾊﾞｲｽに応用を書いてください。
- 結果を見るためにﾃﾞｰﾀ可視器(Data Visualizer)に切り替えてください。

重要: 前の例がｸﾛｯｸを8MHz/64に前置分周したので、ISPﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ｸﾛｯｸは主ｸﾛｯｸの1/4以下の32kHz未満に設定されなければなりません。

重要: Xplained Miniの基板上電力を禁止することを忘れないでください。

この図から何を見ることができますか?。

・LED OFFでいっそうより低い電力消費

結果: 起こし元としてのｳｫｯﾁﾄﾞｯｸﾞ ﾀｲﾏの使用はこの例で最低電力休止動作を使うことを許します。

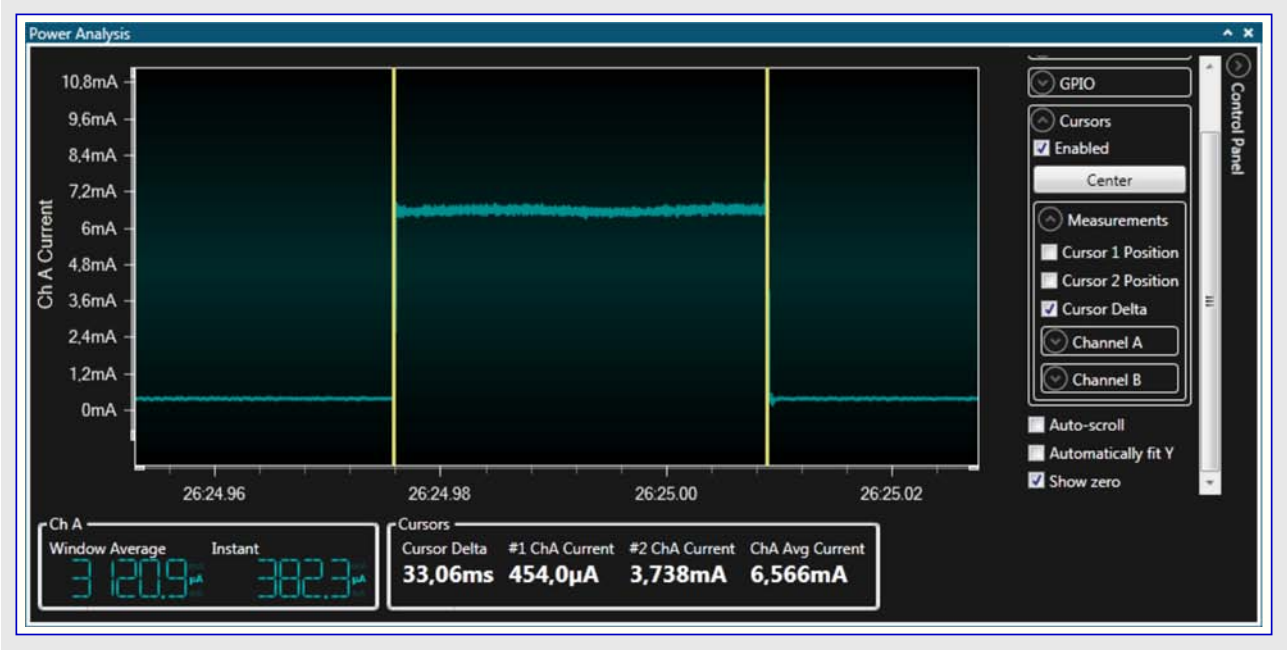
さて、図の詳細をより近くでざっと見てみましょう。ｶｰｿﾙの許可は図からもっと正確な測定の獲得を許します。

すべきこと: ・Power Analysys(電力分析)の右上角でControl Panel(制御盤)を開いてください。
・Cursors(ｶｰｿﾙ)項を展開してください。
・ｶｰｿﾙをONに切り替えるためにEnabled(許可)枠をﾁｪｯｸしてください。

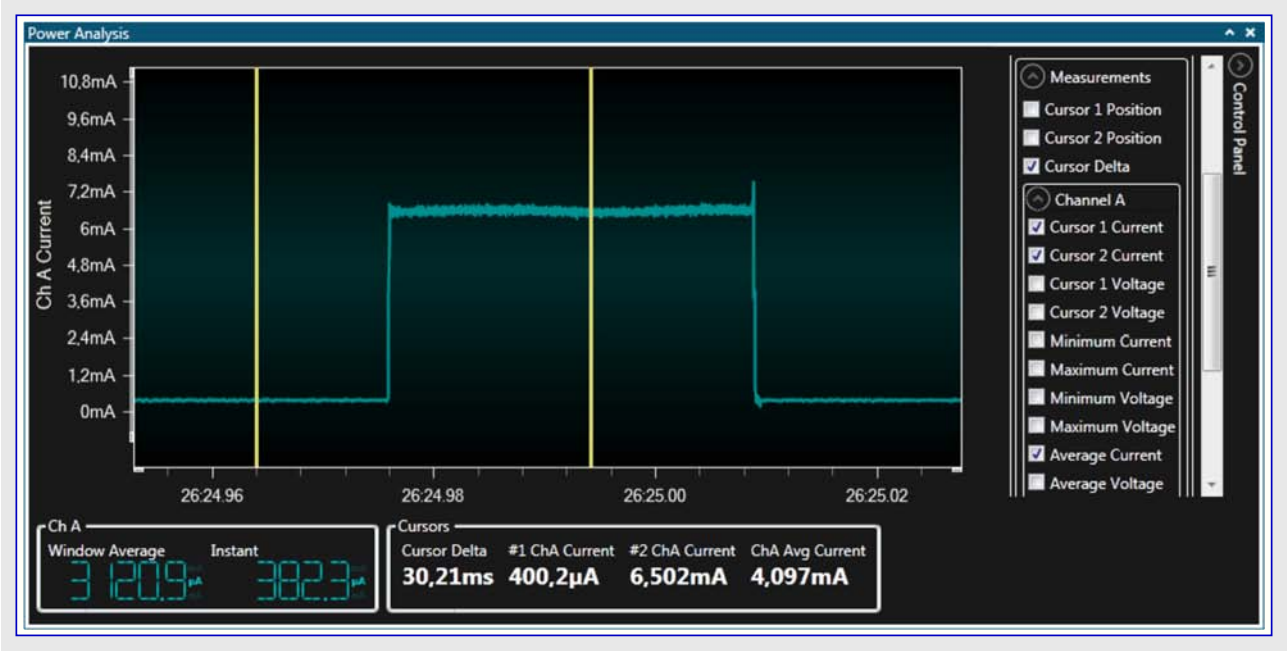
忘れずに: 電流測定が未だ走っている場合、ｶｰｿﾙを許可する前にAuto-scroll(自動ｽｸﾛｰﾙ)の禁止を確実にしてください。そうでなければ図表表示部はｶｰｿﾙを離れて急速にｽｸﾛｰﾙします。

上の図で最初に、ｳｫｯﾁﾄﾞｯｸﾞ ﾀｲﾏが意図されるように概ね毎秒割り込むことを確かめる、差を調べるために2つのｶｰｿﾙを使います。

更に拡大で(Curser Delta(ｶｰｿﾙ間差)として示される)ﾊﾟﾙｽのON幅が33msを僅かに超えるのを見ることができます。これは意図した(1%)よりも少し大きな3.3%のﾃﾞｭｰﾃｨ ｻｲｸﾙと同一視します。主ｸﾛｯｸは5Vで16MHz、CLKPRのDIV64と計時器2のDIV32を使うことで、128µsの刻時を与えます。256周期毎の溢れで、LED ON時間は従って32.7msで、測定(値)近くです。故に1%のﾃﾞｭｰﾃｨ ｻｲｸﾙを達成するために計時器2の再設定値を$00から$B2に増やすことができます。



図からもっと正確な電流測定を得るため、ここ(下図の右側枠)で示されるようにｶｰｿﾙ電流測定を許可してください。ｶｰｿﾙ値はLED OFFで約400µA、LED ONで6.5mAとして示されます。



4.1.13. ｺｰﾄﾞ計装の使い方

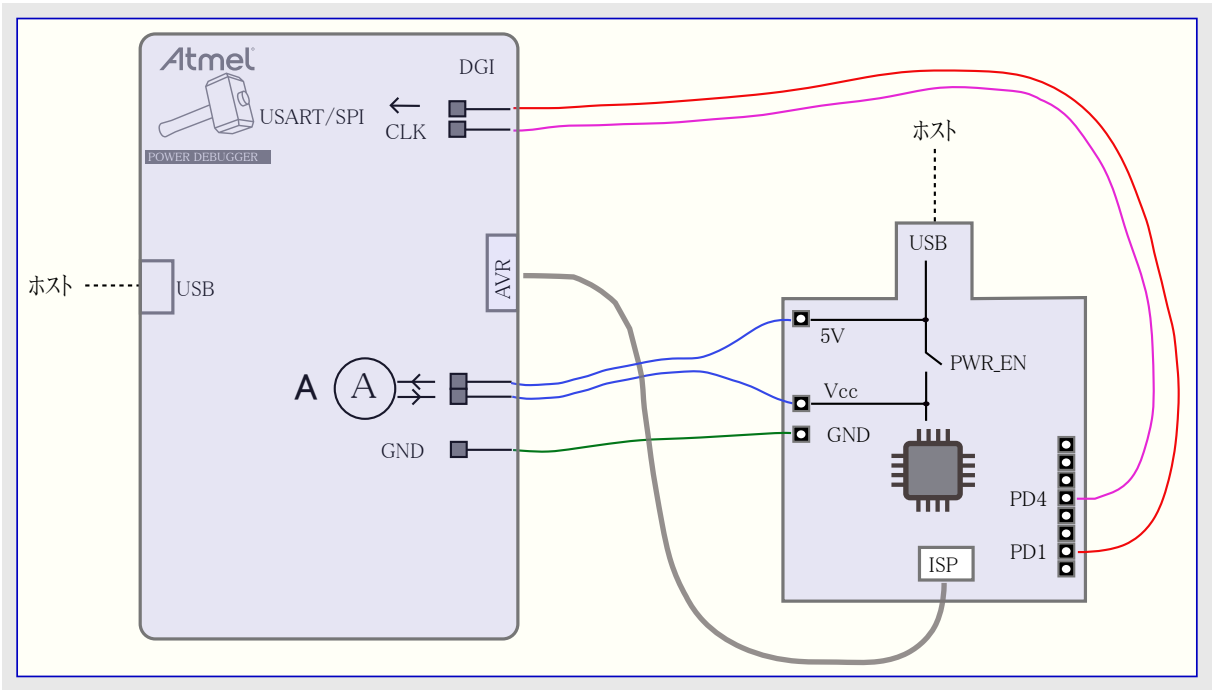
最後に、本項ではAtmelﾃﾞｰﾀ可視器(Data Visualizer)ができるいくつかの他の要領を考察します。

電流測定を得ることはさておき、Power Debuggerは目的対象からﾎｽﾄ ｺﾝﾋﾟｭｰﾀへﾃﾞｰﾀを流すための完全なﾃﾞｰﾀ交換器ｲﾝﾀｰﾌｪｰｽ(DGI:Data Gateway Interface)ﾎﾟｰﾄを持ちます。簡単なｺﾝｿｰﾙ出力のためにこれをUSARTに繋ぎます。

USARTを繋ぐのに行われるべきいくつかのﾊｰﾄﾞｳｪｱ配線が必要です。ﾎﾞｰﾚｰﾄ設定での利便性のため同期動作を使います。

- Xplained MiniのPD1(TXD)はDGIﾍｯﾀﾞのUSART ←に接続されます。
- Xplained MiniのPD4(XCK)はDGIﾍｯﾀﾞのUSART CLKに接続されます。

更新された配線構成図がここで示されます。



例のｺｰﾄﾞはLEDを交互切り替えするのと刻時部を更新するのにXppained Mini上の釦に接続されたﾋﾟﾝ変化割り込みを使います。各釦押下でLEDが交互切り替えして刻時部計数がLED状態ﾒｯｾｰｼﾞと共にDGI USARTを使って送られます。

下のｺｰﾄﾞはlow_power_105ﾌﾟﾛｼﾞｪｸﾄに含まれます。

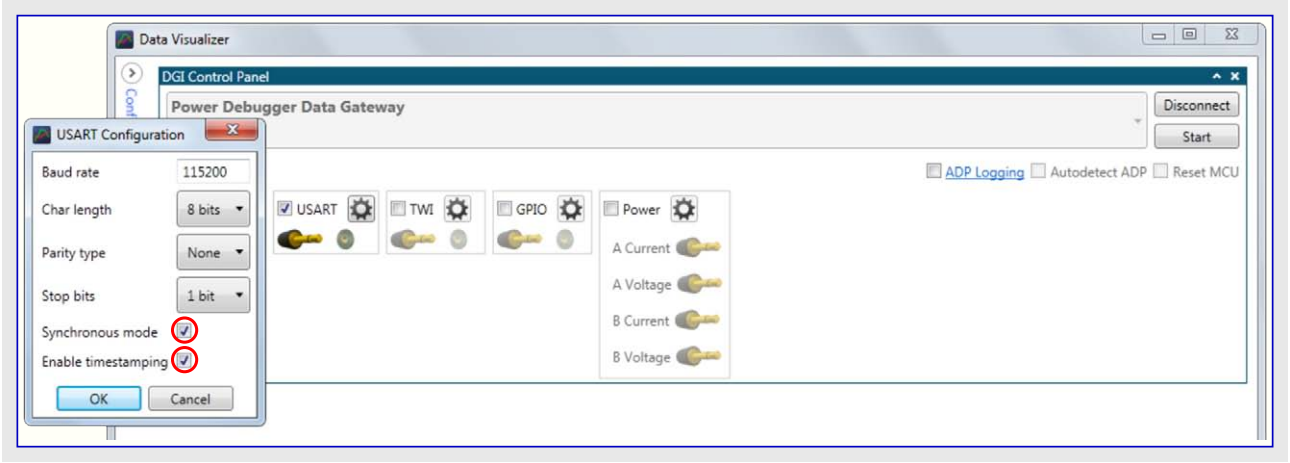
```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>

volatile uint8_t led_on;
volatile uint8_t send_message;
volatile uint8_t ticker = 0;

const char* message_on = "LED ON ";
const char* message_off = "LED OFF ";

ISR (PCINT0_vect)
{
    // 各釦押下は2つのﾋﾟﾝ変化割り込み(押下、開放)を生成、これらの半分は無視
    if (PINB & (1 << 7))
        return;
    // LED更新
    if (led_on)
        PORTB &= ~(1 << 5);
    else
        PORTB |= (1 << 5);
    led_on = ~led_on;                       // led_onﾌﾗｸﾞ反転
    send_message = 1;                       // ﾒｯｾｰｼﾞ送出を合図
    ticker++;                               // 刻時部進行
    if (ticker >= 10)
        ticker = 0;                         // 刻時部ﾘｾｯﾄ
}

void usart_send (const char data)
{
    UDR0 = data;                            // USARTに文字送出
    while (!(UCSR0A & (1 << TXC0)));        // 文字送出済み待ち
    UCSR0A = (1 << TXC0);                   // ﾌﾗｸﾞ解除
}

int main(void)
{
    DDRB = (1 << 5);                        // ﾎﾟｰﾄPB5を出力に設定

    PORTB = 0;                              // LED OFF
    led_on = 0;                             //

    PCICR = (1 << PCIE0);                   // ﾋﾟﾝ変化割り込み許可
    PCMSK0 = (1 << PCINT7);                 //

    // USART0
    UBRR0 = 0x0FFF;                         // 同期動作で転送速度はあまり重要ではありません。
    DDRD |= (1 << 4);                       // 主装置ｸﾛｯｸ出力用XCK許可(出力設定)
    UCSR0B |= (1 << TXEN0);                 // USART送信許可
    UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00) | (1 << UMSEL00);     // 同期動作設定

    sei();                                  // 全体割り込み許可

    while(1)  {
        set_sleep_mode(SLEEP_MODE_IDLE); // 休止動作移行
        sleep_mode();                       //
        // 起き上がり
        if(send_message) {                  // ﾒｯｾｰｼﾞ送出
```

```
                const char* pmessage;
                if (led_on)
                      pmessage = message_on;
                else
                      pmessage = message_off;
                while (*pmessage)
                      usart_send(*pmessage++);
                usart_send(ticker + '0');    // 刻時部の値を送出
                usart_send('¥n');            //
                send_message = 0;            // ﾒｯｾｰｼﾞ送出要求ﾌﾗｸﾞ解除
          }
     }
}
```

すべきこと:
- ﾌﾟﾛｼﾞｪｸﾄ/解決策を構築(F7)してください。
- Start Without Debugging(ﾃﾞﾊﾞｯｸﾞなしで開始) (Ctrl+Alt+F5)を使い、目的対象ﾃﾞﾊﾞｲｽに応用を書いてください。
- 結果を見るためにﾃﾞｰﾀ可視器(Data Visualizer)に切り替えてください。

4.1.13.1. USARTｲﾝﾀｰﾌｪｰｽ許可

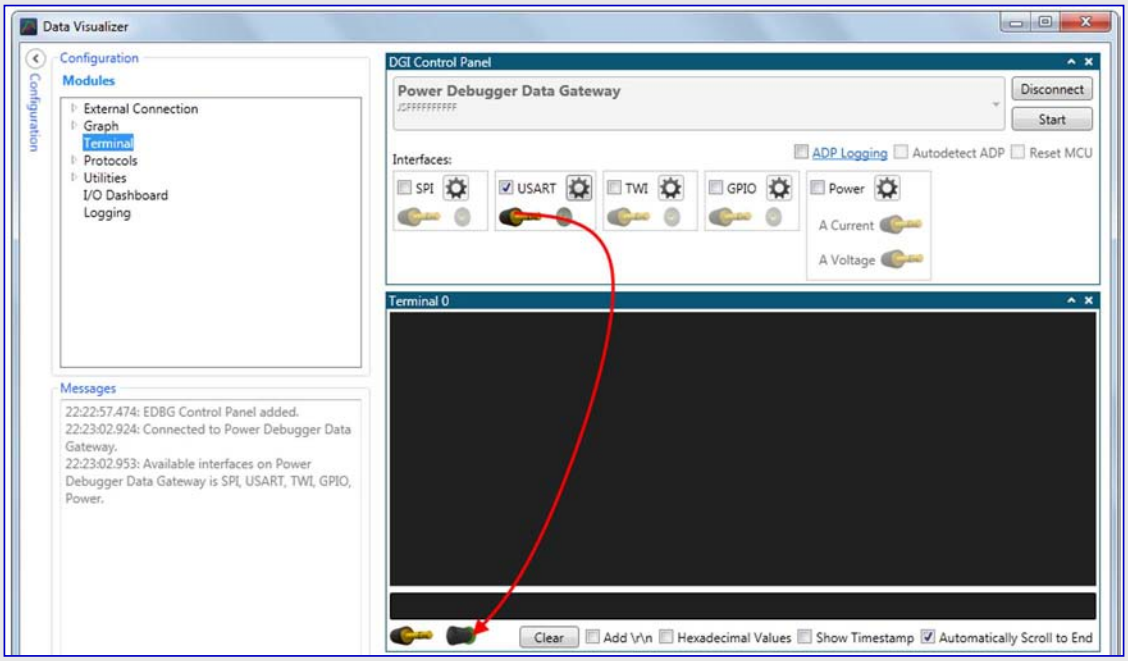
すべきこと:
- USARTﾁｪｯｸ枠をﾁｪｯｸしてください。
- USARTﾁｪｯｸ枠傍らの歯車釦を押すことによってUSART Configuration(USART構成設定)ﾀﾞｲｱﾛｸﾞを開いてください。



すべきこと:
- USART Configuration(USART構成設定)ﾀﾞｲｱﾛｸﾞで時刻印付きの同期動作を許可してください。

4.1.13.2. インターフェースを端末に接続

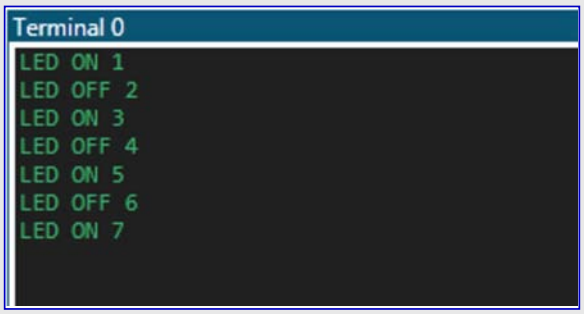
すべきこと:
- Configuration(構成設定)盤を開いてください。
- Data Visualizer(データ可視器)にTerminal(端末)表示部を追加してください。
- 接続を行うためにDGI Control Panel(DGI制御盤)内のインターフェースからのsource(送り手)を端末用のsink(受け手)へ引き摺って(ドラッグして)ください。



4.1.13.3. 走行

すべきこと:
- 作業を開始してください。
- LEDを交互切り替えするためにXplained Mini上の釦を押してください。

各切り替えで端末に於いてﾒｯｾｰｼﾞを受け取るでしょう。



結果: この使用事例では以下を知りました。
- Power DebuggerとAtmelﾃﾞｰﾀ可視器(Daya Visualizer)を使う簡単な電力作図
- ｶｰｽﾙを使うもっと正確な電流と時間の測定取得
- 簡単なUSARTに基づくｺｰﾄﾞ計装

4.2. USB給電応用

この使用事例はPower Debugger上のUSB”通過”ｺﾈｸﾀの使い方を示します。設計者がUSBﾎﾟｰﾄを渡る総電力消費と基板上の目的対象ﾃﾞﾊﾞｲｽによって消費されるそれの両方で興味を持つUSB給電製品の例としてSAM L21 Xplained Proｷｯﾄを使います。SAM L22ｷｯﾄを使うこともできます。

4.2.1. 必要条件

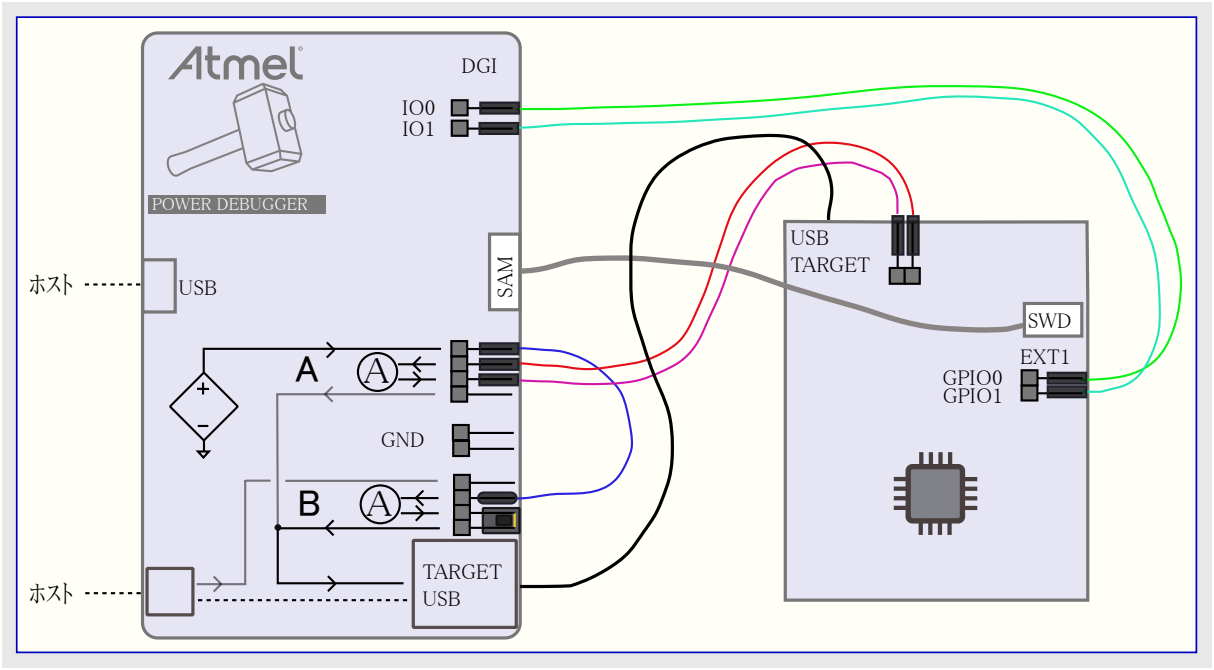
この例を通して動くようにできるには以下が必要とされます。

- (Atmelﾃﾞｰﾀ可視器(Data Visualizer)が含まれる)ｲﾝｽﾄｰﾙされたAtmel Studio 7(またはそれ以降)を持つﾎｽﾄ ｺﾝﾋﾟｭｰﾀ
- (ｹｰﾌﾞﾙを含む)Atmel Power Debuggerｷｯﾄ
- Atmel ATSAM L21 Xplained Proｷｯﾄ、またはUSB装置ｺﾈｸﾀ、外部ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾍｯﾀﾞ、電流測定ﾍｯﾀﾞを持つ同様の目的対象基板

助言: SAM L21 Xplained Proも基板上に統合された電流測定機能を持ちますが、それらはこの使用事例で使われません。

4.2.2. 初期ﾊｰﾄﾞｳｪｱ準備

初期構成の構成図がここで示されます。



4.2.3. 接続

構成図に従ってｷｯﾄを接続してください。接続はここで説明されます。

目的対象に給電するには: 可変電圧供給源から目的対象を給電します。これはUSB仕様の全電圧範囲に渡るUSB装置の試験を許します。

すべきこと:
- 接続ｹｰﾌﾞﾙを使って可変電圧出力をBﾁｬﾈﾙの入力に接続してください。これは目的対象によって引き出される総電流の測定を許します。
- Bﾁｬﾈﾙの出力をUSB A型ｼﾞｬｯｸのPOWER入力に接続してください。これはｼﾞｬﾝﾊﾟを使うことによって簡単に行われます。
- A型ｼﾞｬｯｸから TARTGET USBﾏｲｸﾛ ｼﾞｬｯｸに接続してください。

目的対象MCUによって引き出された電流を測定するには: Xplained Pro基板はTARGET USBｺﾈｸﾀの傍らに2ﾋﾟﾝの電力ﾍｯﾀﾞを持ちます。通常はVCC_TARGETをVCC_MCUに接続するｼﾞｬﾝﾊﾟがここに配置されます。このｼﾞｬﾝﾊﾟを取り外して電流測定ﾁｬﾈﾙを通してこの回路を配線することにより、基板全体ではなく、MCUによってのみ引き出される電流を測定することができます。

すべきこと:
- 目的対象基板のVCC_TARGETをAﾁｬﾈﾙの入力に接続してください。
- Aﾁｬﾈﾙの出力を目的対象基板のVCC_MCUﾋﾟﾝに接続してください。

汎用入出力(GPIO)計装を使うには: Power DebuggerはAtmelﾃﾞｰﾀ可視器(Data Visualizer)で監視することができる4つのGPIOﾁｬﾈﾙを持ちます。その後に応用ｺｰﾄﾞにﾋﾟﾝの交互切り替えを追加してそれらの事象をﾃﾞｰﾀ可視器で見ることができます。

すべきこと:
- Xplained ProのEXT1ﾍｯﾀﾞのPB07(GPIO0)をPower debuggerのDGIﾍｯﾀﾞのIO0に接続してください。
- Xplained ProのEXT1ﾍｯﾀﾞのPB06(GPIO1)をPower debuggerのDGIﾍｯﾀﾞのIO1に接続してください。

目的対象を書いてﾃﾞﾊﾞｯｸﾞするには: Xplained Proがﾃﾞﾊﾞｯｸﾞ能力を持つとは言え、この目的にはPower Debuggerを使います。

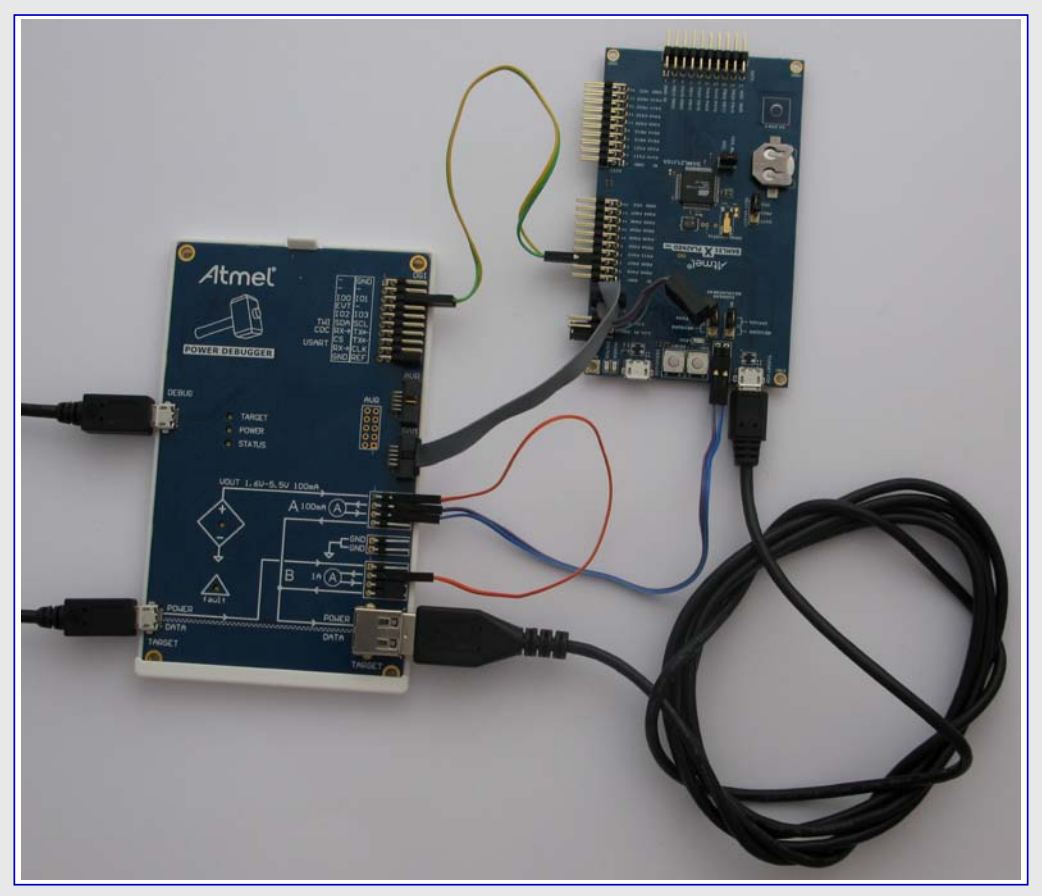
すべきこと:
- Power DebuggerのSAMﾍｯﾀﾞからの10ﾋﾟﾝ ﾃﾞﾊﾞｯｸﾞ ｹｰﾌﾞﾙをXplained ProのCortex DEBUGﾍｯﾀﾞに接続してください。

この全てをﾎｽﾄ ｺﾝﾋﾟｭｰﾀに接続するには:

すべきこと:
- Power debuggerのDEBUGｺﾈｸﾀからのUSBｹｰﾌﾞﾙをﾎｽﾄ ｺﾝﾋﾟｭｰﾀに接続してください。
- Power debuggerのTARGETｺﾈｸﾀからのUSBｹｰﾌﾞﾙをﾎｽﾄ ｺﾝﾋﾟｭｰﾀに接続してください。

助言: ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾍｯﾀﾞが接続されている時はGND接続が必要ありません。

構成はこのようなものに見えるでしょう。



注: 写真は僅かに間違っています。ﾃﾞﾊﾞｯｸﾞ ｹｰﾌﾞﾙはSAMﾃﾞﾊﾞｯｸﾞ ｺﾈｸﾀに接続されていますが、AVRﾃﾞﾊﾞｯｸﾞ ｺﾈｸﾀに接続されているべきです。(**訳注**:本文(4.2.2.と4.2.3.)での記述も対象はSAMｺﾈｸﾀなのでこの注が誤っているものと思われます。)

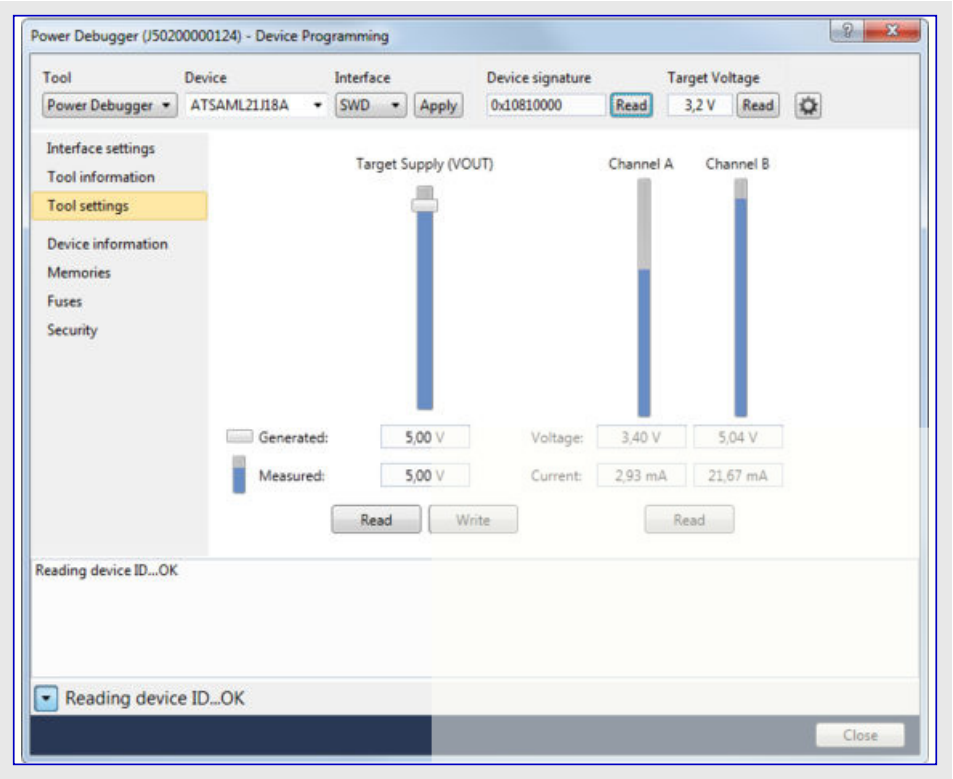
4.2.4. VOUT目的対象供給

Power Debuggerは最大100mA、1.6～5.5Vの範囲で目的対象に電圧を供給することができます。電圧はｼﾙｸ ｽｸﾘｰﾝで示されるようにﾋﾟﾝへ印加され、使用者によって適切な負荷に配線されなければなりません。VOUTはいくつかの方法で設定することができます。

重要: VOUT設定はPower Debugger上の不揮発性ﾒﾓﾘに格納されます。ﾂｰﾙの電源OFF/ONはVOUT切断に帰着します。これは使用者のﾊｰﾄﾞｳｪｱを保護するために行われます。

4.2.4.1. ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾀﾞｲﾛｸﾞを使うVOUT設定

VOUTはAtmel Studioのﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾀﾞｲﾛｸﾞを使って設定と読み戻しをすることができます。これはPower Debuggerを使う時に利用可能なTool settings(ﾂｰﾙ設定)ﾀﾌﾞを使って行われます。望む電圧を設定するのに摺動子を使い、変更を適用するのにWrite(書き込み)をｸﾘｯｸしてください。測定した値が自動的に読み戻されます。



4.2.4.2. atprogram.exeを使うVOUT設定

VOUTは次のように命令ﾊﾟﾗﾒｰﾀ経由でatprogramｺﾏﾝﾄﾞ行ﾕｰﾃｨﾘﾃｨを使って設定することができます。

```
atprogram.exe -t powerdebugger parameters -ts 5.0
```

VOUT値は以下を使い、採取して読み戻すことができます。

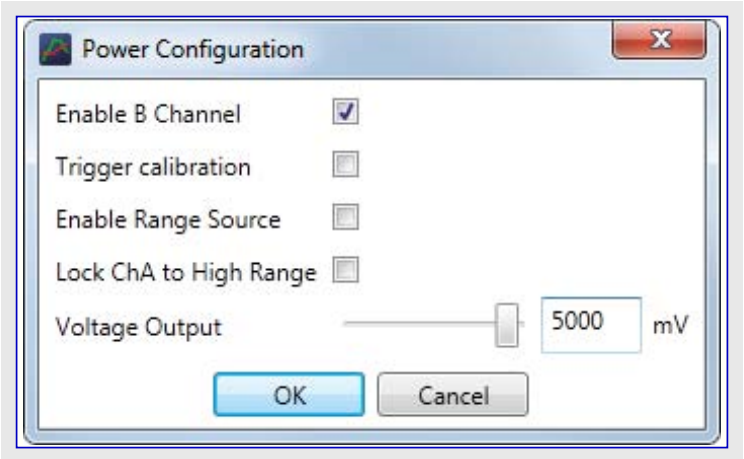
```
atprogram.exe -t powerdebugger parameters -tsout
```

VOUT設定点は以下を使って読み戻すことができます。

```
atprogram.exe -t powerdebugger parameters -tssp
```

4.2.4.3. ﾃﾞｰﾀ可視器を使うVOUT設定

VOUTは電力ｲﾝﾀｰﾌｪｰｽ用構成設定を開くことによってﾃﾞｰﾀ可視器(Data Visualizer)を使って設定することができます。

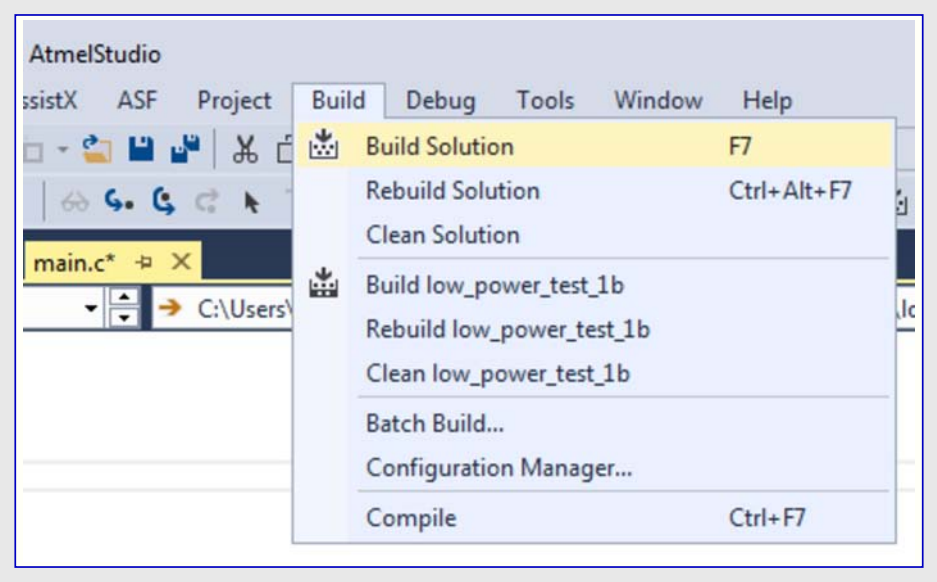


4.2.5. 両測定ﾁｬﾈﾙの使い方

Power Debuggerの2つの電流測定ﾁｬﾈﾙは2つの違う領域の電流消費を分析する時に有用です。例えば、基板ﾚﾍﾞﾙの電流引き出しとMCUだけの電流引き出しの両方を作図することができます。

USBﾊｰﾄﾞｳｪｱ装置を徹底して実演するため、ASFからの大容量記憶装置例を使います。

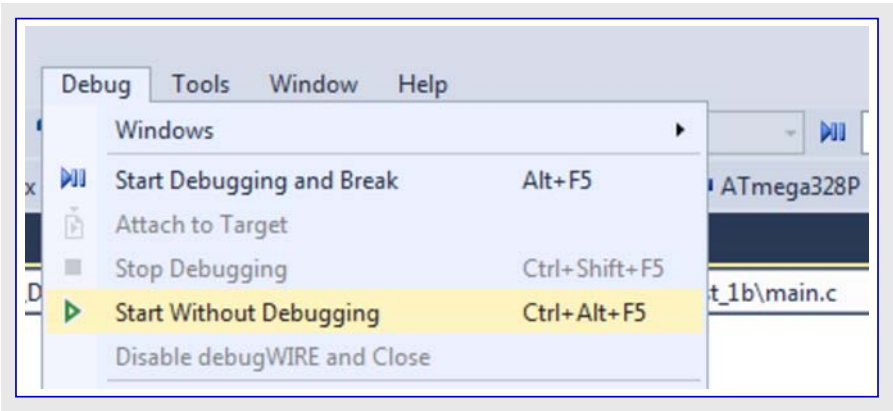
すべきこと:
- Atmel StudioでNew Example Project(新規例ﾌﾟﾛｼﾞｪｸﾄ)を作成してください。
- New Example(新しい例)ﾀﾞｲｱﾛｸﾞで、SAML21ﾃﾞﾊﾞｲｽ系統(または他の適切なﾃﾞﾊﾞｲｽ)を選んで鍵となる語の"MSC"によって選別してください。
- USB Device MSC Example(USB大容量記憶装置例)を選んでください。
- ﾌﾟﾛｼﾞｪｸﾄ/解決策を構築(F7)してください。



すべきこと:
- ﾌﾟﾛｼﾞｪｸﾄ ﾌﾟﾛﾊﾟﾃｨを開いてください(Solution Explorer(解決策ｴｸｽﾌﾟﾛｰﾗ)でﾌﾟﾛｼﾞｪｸﾄを右ｸﾘｯｸしてProperties(ﾌﾟﾛﾊﾟﾃｨ)を選んでください)。
- Tool(ﾂｰﾙ)ﾀﾌﾞで適切なﾂｰﾙとｲﾝﾀｰﾌｪｰｽを選んでください。

すべきこと: Start Without Debugging(ﾃﾞﾊﾞｯｸﾞなしで開始) (Ctrl+Alt+F5)を使って目的対象ﾃﾞﾊﾞｲｽに応用を書いてください。

情報: この例のこの部分ではﾌﾟﾛｸﾞﾗﾐﾝｸﾞ動作だけを使うつもりです。ﾃﾞﾊﾞｯｶﾞを通してｺｰﾄﾞを走行する殆どのｼｽﾃﾑでは正確な電流測定を提供しません。これは目的対象のﾃﾞﾊﾞｲｽのﾃﾞﾊﾞｯｸﾞ単位部(OCD)がﾃﾞﾊﾞｯｸﾞ中に禁止することができないｸﾛｯｸ元を必要とするからです。



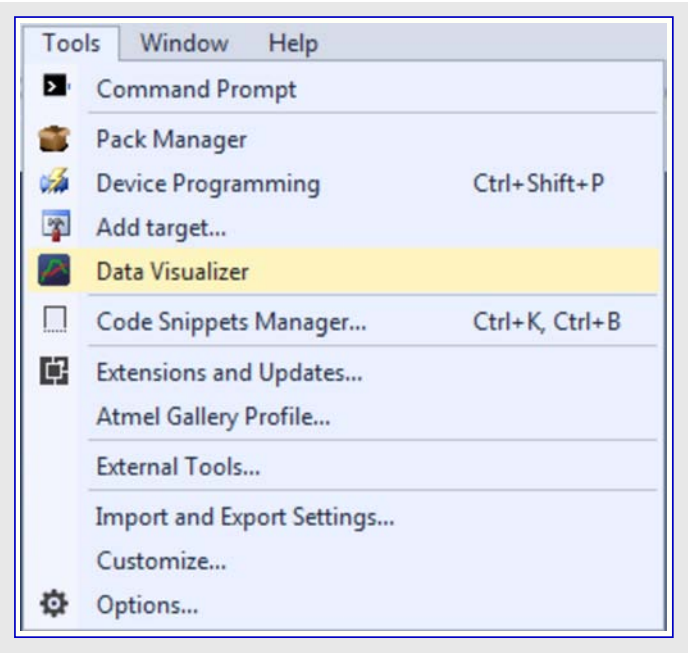
4.2.6. Atmelﾃﾞｰﾀ可視器の起動

Atmelﾃﾞｰﾀ可視器(Data Visualizer)はAtmel Studioの一部として含まれ、Atmel Studio拡張として、または独立動作形態でのどちらでも動かすことができます。

Atmel Studio内の拡張としてAtmelﾃﾞｰﾀ可視器を動かすにはTools(ﾂｰﾙ)ﾒﾆｭｰでそれを選んでください。

ﾃﾞｰﾀ可視器機能を支援するｷｯﾄはAtmel Studio内のそれらの開始頁でその拡張へのｼｮｰﾄｶｯﾄを含みます。

独立版のAtmelﾃﾞｰﾀ可視器がｲﾝｽﾄｰﾙされた場合、Windows®のｽﾀｰﾄ ﾒﾆｭｰでｼｮｰﾄｶｯﾄを探してください。独立版はgallery.atmel.comからのﾀﾞｳﾝﾛｰﾄﾞで入手可能です。

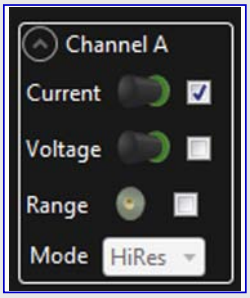


4.2.7. 2ﾁｬﾈﾙ測定

2つの測定ﾁｬﾈﾙと持つﾊｰﾄﾞｳｪｱ使用時、ﾃﾞｰﾀ可視器(Data Visualizer)は(電力構成設定(Power Configuration)で禁止されない限り)同じ図で両方を表示します。

単位部右側のControl panel(制御盤)で電流と電圧の描画だけでなく利用可能な時に範囲の情報も表示または非表示にすることができます。

既定によって両ﾁｬﾈﾙはPower Analysis(電力分析)図で示されますが、各描画は適合された最善としてそれらを分離するために上または下に移動することができます。



4.2.8. 図表の大きさ調整とｽｸﾛｰﾙ

助言: 未だ走っている間にもっと綿密に調べるにはAuto-scroll(自動ｽｸﾛｰﾙ)とAutomatically fit Y(Y自動適合)をOFFにしてください。

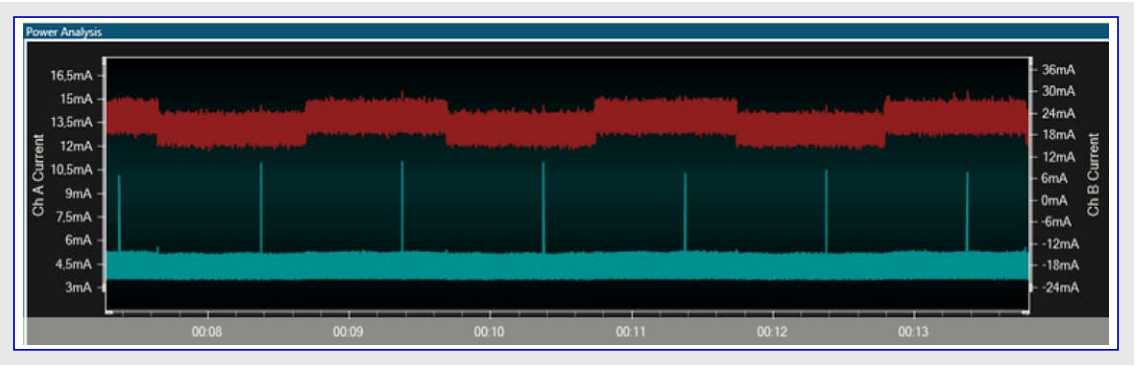
望みに従って描画を移動するにはscale(拡大/縮小)とoffset(変位)の制御を使ってください。ﾏｳｽのｽｸﾛｰﾙ ﾎｲｰﾙはこの関係に於いて以下のように有用です。

- 時間(X)軸で拡大/縮小するには描画上でShift+ｽｸﾛｰﾙを行ってください。
- Y軸で拡大/縮小するには描画上でCtrl+ｽｸﾛｰﾙを行ってください。
- 時間(X)軸での左右移動とY軸の移動(変位)を行うには拡大/縮小するには図を引き摺ってください。
- Y軸(変位)で各々のﾁｬﾈﾙを移動するには左と右の目盛りを引き摺ってください。
- そのﾁｬﾈﾙを拡大/縮小するには各々の軸上でCtrl+ｽｸﾛｰﾙを行ってください。
- 拡大/縮小する領域を選ぶには右ｸﾘｯｸして引き摺ってください。

4.2.9. 大容量記憶例

すべきこと:
- Start Without Debugging(ﾃﾞﾊﾞｯｸﾞなしで開始) (Ctrl+Alt+F5)を使って目的対象ﾃﾞﾊﾞｲｽに大容量記憶ｸﾗｽ例を書いてください。
- 図の捕獲後にﾃﾞｰﾀ可視器(Data Visualizer)を停止してください。

いくつかのスクロールと大きさ調整で描画は次のこのようなものに見えるでしょう。



赤の図はBチャネルで、基板全体によって引き出された電流を表します。約1秒間隔で2つの動作間を交互切り替えしているように見えます。これはLEDのONとOFFの交互切り替えです。

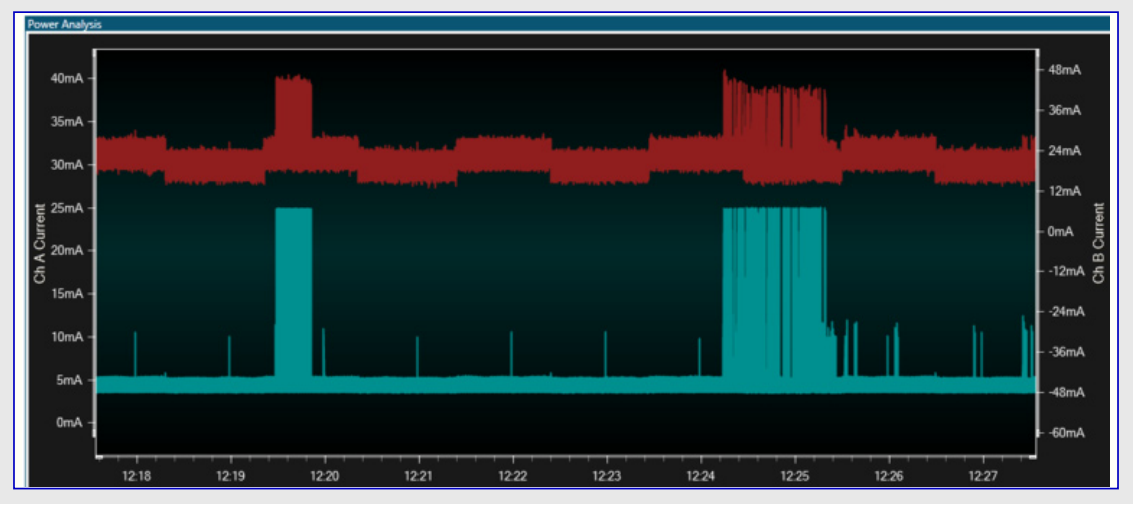
青の図はAチャネルで、目的対象MCUだけによって引き出された電流を表します。これもまた1秒間隔で最大10mAのﾊﾟﾙｽと共に約4.5 mAを引き出すように見えます。

助言: 着目: Windowsのﾃﾞﾊﾞｲｽ ﾏﾈｰｼﾞｬでUSB'ﾃﾞｨｽｸ'を無効にすることは、発生しているこの;ﾊﾟﾙｽを停止し、これはおそらくWindowsからの事象によって起動されることを示します。

すべきこと: Windowsのｴｸｽﾌﾟﾛｰﾗを開いて大容量記憶装置のﾌｧｰﾏｯﾄを実行してください。

注意: 正しいﾃﾞｨｽｸをﾌｫｰﾏｯﾄするよう注意してください!。

ここで示されるように、描画で今や基板とMCUの両チャネルでもっとより高い電流消費の期間をみるでしょう。



4.2.10. GPIO計装

ﾋﾟﾝ交互切り替えは度々、応用をﾃﾞﾊﾞｯｸﾞするための簡単で有用な機構です。大容量記憶装置の読み書き周期特性を明らかにするために今やいくつかの簡単な計装を追加します。

すべきこと: ui.cでui_start_read()関数の実装を見つけてください。これはui.hを開くことにより、関数を選択し、前後関係ﾒﾆｭｰでGoto Inplementation(実装へ行く)任意選択を使って容易に行うことができます。

ここで示されるような、読み込みｱｸｾｽに対してGPIO0、書き込みｱｸｾｽに対してGPIO1を交互切り替えする計装を追加してください。

```
void ui_start_read(void)
{
    port_pin_set_output_level (EXT1_PIN_GPIO_0, true);    // GPIO0をHigh
}

void ui_stop_read(void)
{
    port_pin_set_output_level (EXT1_PIN_GPIO_0, false);  // GPIO0をLow
```

```
}

void ui_start_write(void)
{
    port_pin_set_output_level (EXT1_PIN_GPIO_1, true);    // GPIO1をHigh
}

void ui_stop_write(void)
{
    port_pin_set_output_level (EXT1_PIN_GPIO_1, false);  // GPIO1をLow
}
```

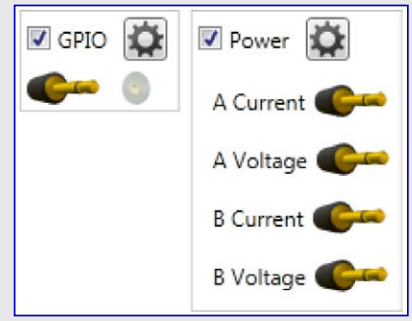
board_init.cファイルを見つけてGPIO0と1の初期化を追加してください。これはsystem_board_ini(void)でのLED_0_PIN初期化を複写することによって追加することができます。

```
    port_pin_set_config(EXT1_PIN_GPIO_0, &pin_conf);      // GPIO0を出力に設定
    port_pin_set_config(EXT1_PIN_GPIO_1, &pin_conf);      // GPIO1を出力に設定
```

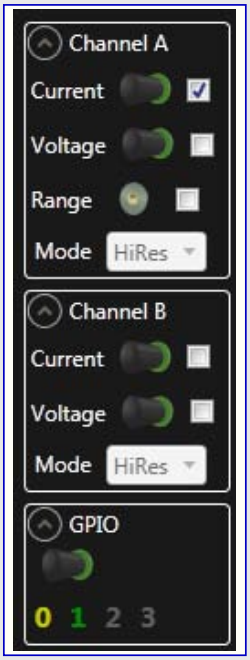
ここでそれが関連しないように、LED交互切り替えを注釈にすることもできます。

```
void ui_process(uint16_t framenumber)
{
    /*if (0 == framenumber) {
        LED_On(LED_0_PIN);
    }
    if (1000 == framenumber) {
        LED_Off(LED_0_PIN);
    }*/
}
```

すべきこと:
- プロジェクト/解決策を構築(F7)してください。
- Start Without Debugging(デバッグなしで開始) (Ctrl+Alt+F5)を使い、目的対象デバイスに応用を書いてください。
- 結果を見るためにデータ可視器(Data Visualizer)に切り替えてください。
- Power(電力)インターフェースとGPIOの両方を選択して可視化を開始してください。



すべきこと:
- Control panel(制御盤)を開いてください。
- Channel B(Bチャネル)のチェックを外してください。
- GPIO2と3のチェックを外してください。



電力図は今や以前の課題と同様に見えるでしょう。さて、Windowsを通してﾃﾞｨｽｸ ﾌｫｰﾏｯﾄを実行してGPIO0と1の信号がどう交互切り替えするかに注意してください。ﾃﾞｰﾀ可視器(Data Visualizer)を停止するか、または自動ｽｸﾛｰﾙを禁止し、着目する小片を拡大してください。読みと書きの各ｱｸｾｽが開始と停止の状態を示すためにどうGPIOの交互切り替えを持つかをはっきりと見ることができます。

これが完全準拠のﾃﾞｨｽｸ ﾄﾞﾗｲﾌﾞのため、以下のようないくつかの他の操作をしてみて読みと書きのｱｸｾｽの事象がどう影響を及ぼされるかを見てください。

- 新しいﾃｷｽﾄ ﾌｧｲﾙを作成してください。
- ﾉｰﾄﾊﾟｯﾄﾞを使って内容を追加してください。
- ﾌｧｲﾙを保存してください。
- ﾌｧｲﾙを改めて開いてください。
- ﾌｧｲﾙを削除してください。

助言: ｵﾍﾟﾚｰﾃｨﾝｸﾞ ｼｽﾃﾑによる読み込みｷｬｯｼｭは書き込み後にｷｬｯｼｭが掃き出されるまで読み込みｱｸｾｽを実行させないかもしれません。

4.2.11. ｺｰﾄﾞの相互関係

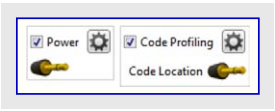
次にｺｰﾄﾞ実行を電力消費に関連付けするためにﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀのﾎﾟｰﾘﾝｸﾞがどう使われ得るかを実演します。

Power Debuggerは実行中に可能な限り速くﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀを繰り返しﾎﾟｰﾘﾝｸﾞするようにﾃﾞｰﾀ可視器(Data Visualizer)によって指示され得ます。これは非常に低い割合の追跡されるｺｰﾄﾞ行をもたらす一方で、意図されるよりももっと電力を使うｺｰﾄﾞの領域を捕らえるのに有用です。

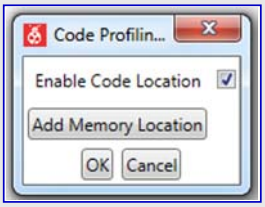
4.2.11.1. ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ ﾎﾟｰﾘﾝｸﾞ許可

電流測定ﾃﾞｰﾀと共にﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ採取を見るにはPower(電力)ｲﾝﾀｰﾌｪｰｽとCode Profiling(ｺｰﾄﾞ特性分析)ｲﾝﾀｰﾌｪｰｽの両方が許可されなければなりません。

すべきこと:
- DGI Control panel(DGI制御盤)でPower(電力)ｲﾝﾀｰﾌｪｰｽとCode Profiling(ｺｰﾄﾞ特性分析)ｲﾝﾀｰﾌｪｰｽの両方を許可してください。



すべきこと:
- Code Profiling(ｺｰﾄﾞ特性分析)ｲﾝﾀｰﾌｪｰｽの歯車釦を押すことによってCode Profiling Configuration(ｺｰﾄﾞ特性分析構成設定)ﾀﾞｲｱﾛｸﾞを開いてください。
- Enable Code Location(ｺｰﾄﾞ位置許可)を選択してください。



4.2.11.2. 走行

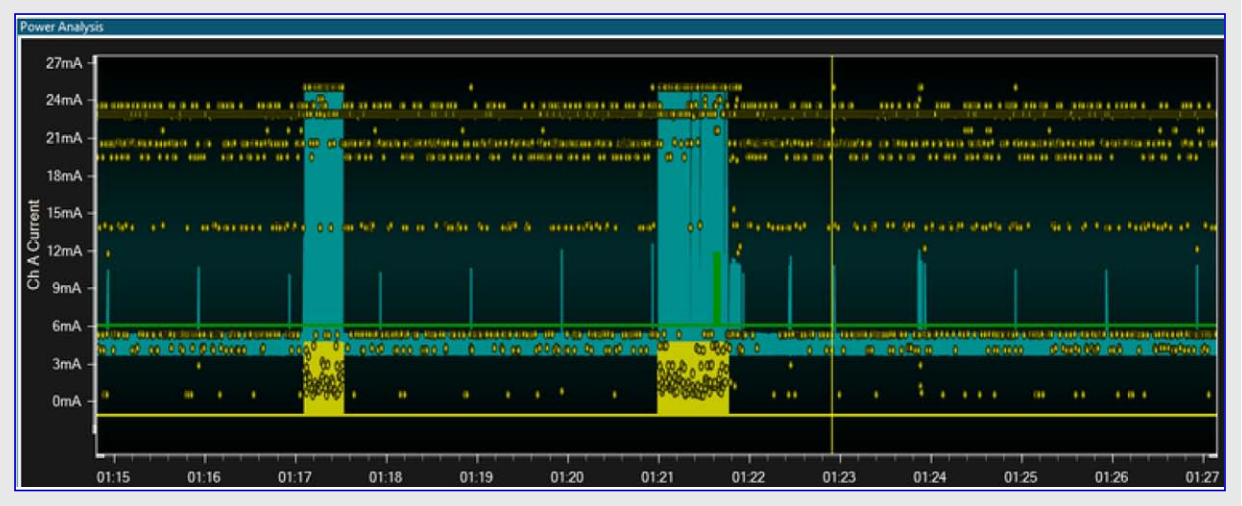
重要: ｺｰﾄﾞ相関はﾃﾞｰﾀ可視器(Data Visualizer)がAtmel Studioの拡張として動く時にだけ利用可能です。これはﾃﾞﾊﾞｯｶﾞが動いている時にだけｼﾝﾎﾞﾘｯｸ情報へのｱｸｾｽが必要だからです。

重要: ｺｰﾄﾞ相関はﾃﾞﾊﾞｯｶﾞが動いていることが必要とされ、故にAtmel StudioでStart Debugging(ﾃﾞﾊﾞｯｸﾞ開始)を使ってください。単に目的対象のﾌﾟﾛｸﾞﾗﾐﾝｸﾞやStart Without Debugging(ﾃﾞﾊﾞｯｸﾞなしで開始)の使用では動きません。

重要: 正しいﾃﾞｨｽｸをﾌｫｰﾏｯﾄするように注意してください。

すべきこと:
- ﾃﾞｰﾀ可視器(Data Visualizer)作業を開始してそれを走らせましょう。
- ﾃﾞｨｽｸ ﾌｫｰﾏｯﾄを実行して結果の描画を捕獲してください。
- 作業を停止するか、または自動ｽｸﾛｰﾙを禁止してください。
- GPIO2と3をOFFに切り替えてください。

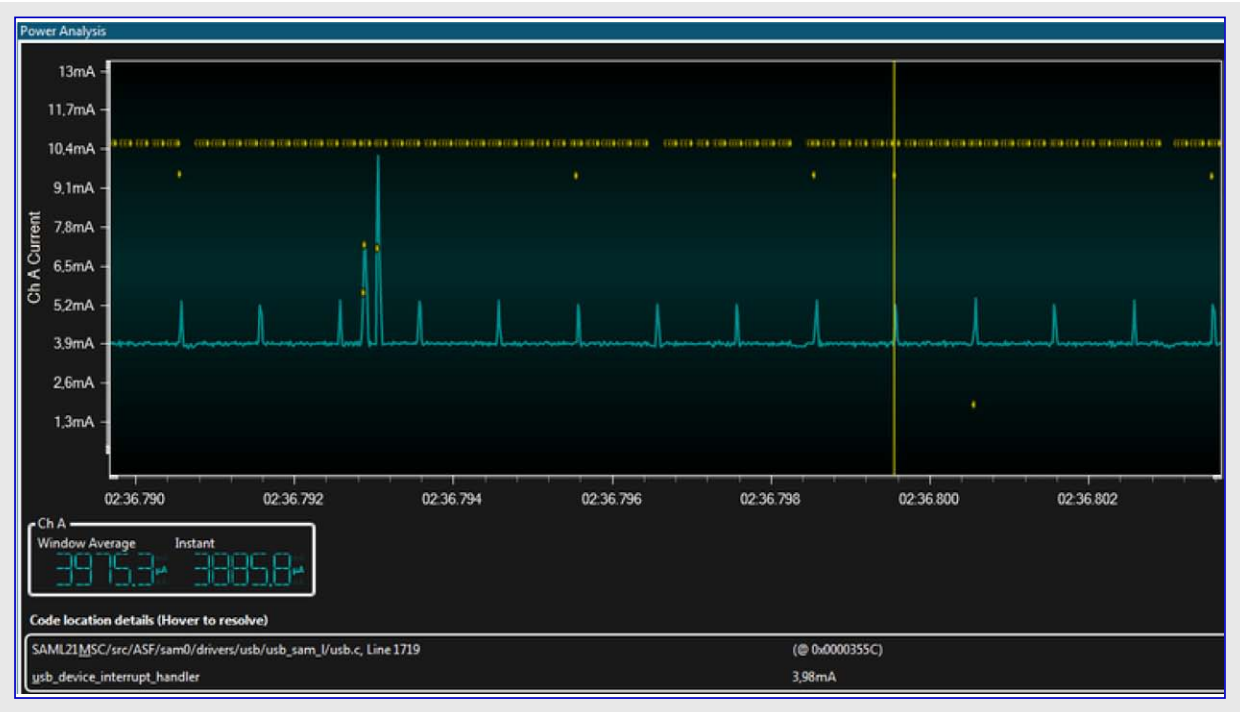
前に識別されたフォーマット活動で拡大すると、このような描画を見るでしょう。



図上で描かれた黄色の点はポーリングしたプログラム カウンタ値を表します。Y軸上のそれらの位置は目的対象デバイスのコード空間でのそれらの位置の視覚表現です。試料の相対的な群分けは違う関数の実行を示します。様式はこの技法を使って容易に見ることができます。(GPIOの活動によって見られるように)フォーマット活動がGPIO活動なしの部分と比べてどうして異なるプログラム カウンタ値の分配を持つかに注目してください。

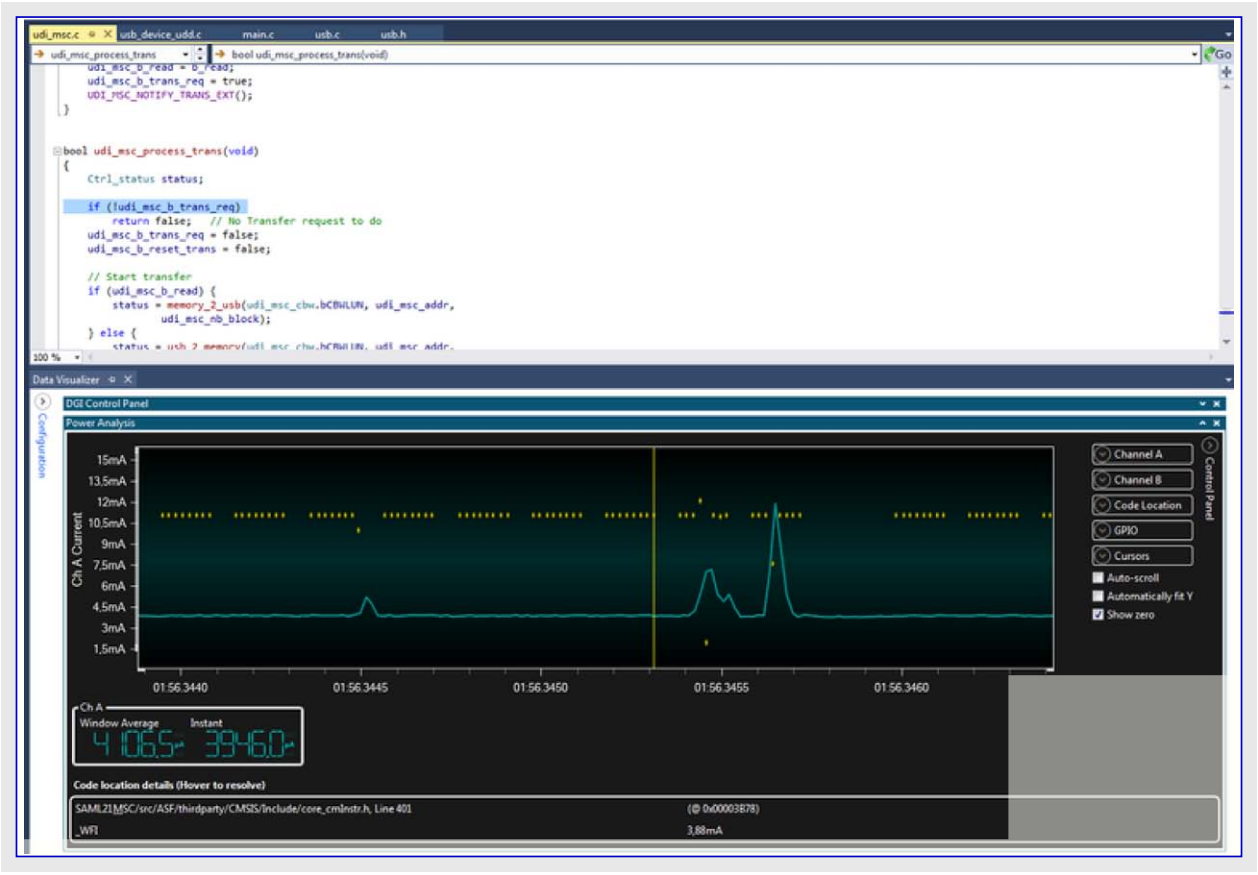
試料の1つの上でのマウス浮かしは図の下のCode Location Details(コード位置詳細)箱でその試料の位置だけでなく、その点の電流消費試料の値も示します。

より大きな電流頂点の1つを拡大すると、このようなものを見るかもしれません。



より小さな電流尖頭波形は非常に規則的で、ms毎に発生します。これらのいくつかは、これが一時的にCPUを起こすUSBのms刻時パルスであることを確かめるusb_device_interrupt_handlerとして解決されるプログラム カウンタ採取値を持ちます。割り込みは全ての実行で捕らわれるには素早く実行されすぎです。より大きな電流尖頭波形は度々別のUSB関数(csw_send/csw_receive)と連携されます。

ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ試料の1つ上のﾀﾞﾌﾞﾙ ｸﾘｯｸはｴﾃﾞｨﾀを開いてその試料に対応する行を強調表示します。



助言: いくつかの試料は"Could not resolve(解決不能)"として記されます。これは位置に対して利用可能なｿｰｽがない時、例えば、試料が予めｺﾝ;ﾊﾟｲﾙされたﾗｲﾌﾞﾗﾘ関数に対応する場合に起き得ます。

4.2.12. ﾃﾞｰﾀ ﾎﾟｰﾘﾝｸﾞ

次にﾃﾞｰﾀ可視器(Data Visualizer)が目的対象からの変数をﾎﾟｰﾘﾝｸﾞし、それらの値を図表形式で表示することができる方法を見ます。

重要: ﾃﾞｰﾀ ﾎﾟｰﾘﾝｸﾞはﾃﾞｰﾀ可視器(Data Visualizer)がAtmel Studioの拡張として動く時にだけ利用可能です。これはAtmel Studioﾃﾞﾊﾞｯｶﾞ後処理部を通してﾃﾞﾊﾞｲｽ上のﾃﾞﾊﾞｯｸﾞ ｼｽﾃﾑをｱｸｾｽすることが必要だからです。

最初にﾎﾟｰﾘﾝｸﾞするための変数を含むｺｰﾄﾞの数行を追加します。

すべきこと: ui.cを開いてﾌｧｲﾙの先頭に2つの全域変数を追加してください。

```
volatile uint32_t write_count = 0;
volatile uint32_t read_count = 0;
```

重要: ﾎﾟｰﾘﾝｸﾞに関係する変数をvolatileとして宣言することはｺﾝﾊﾟｲﾗによってそれらがSRAMに配置され、それらの値がﾚｼﾞｽﾀで貯蔵(ｷｬｯｼｭ)されないことを保証します。ﾚｼﾞｽﾀはﾎﾟｰﾘﾝｸﾞすることができず、SRAM位置だけができます。

助言: ﾃﾞｰﾀ ﾎﾟｰﾘﾝｸﾞは絶対SRAM位置で動作します。故にそれらが常にSRAMの同じ位置で利用可能なようにこの目的に対して全域変数を使うことが推奨されます。ｽﾀｯｸ内の位置のﾎﾟｰﾘﾝｸﾞはﾎﾟｰﾘﾝｸﾞの時間に於いて予測不能なｽﾀｯｸの前後関係に基づく結果を生じる可能があります。

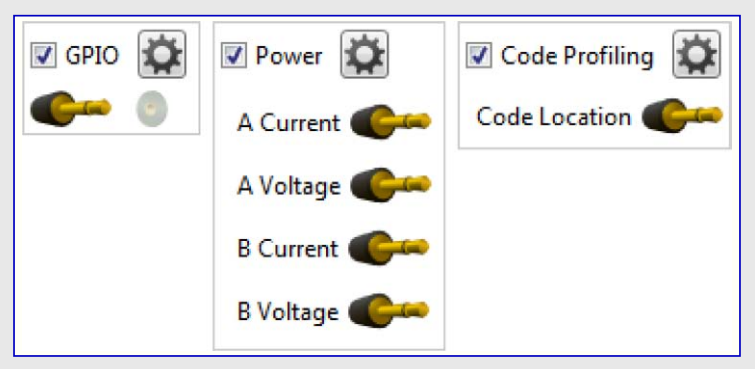
すべきこと: 各開始されたｱｸｾｽで読みと書きの計数器を増すようにui.c内の2つ'start'関数を変更してください。

```
void ui_start_read(void)
{
    port_pin_set_output_level(EXT1_PIN_GPIO_0, true);
    read_count++;
}
```

```
void ui_start_write(void)
{
    port_pin_set_output_level(EXT1_PIN_GPIO_1, true);
    write_count++;
}
```

すべきこと:
- ﾌﾟﾛｼﾞｪｸﾄ/解決策を構築(F7)してください。
- ﾃﾞｰﾀ可視器(Data Visualizer)を開いてください。
- 接続してください。

ﾃﾞｰﾀ ﾎﾟｰﾘﾝｸﾞ機能のためにCode Profiling(ｺｰﾄﾞ特性分析)ｲﾝﾀｰﾌｪｰｽを許可することが必要です。



すべきこと:
- ﾃﾞｰﾀ可視器(Data Visualizer)作業を開始してください。
- Debugging and Break(ﾃﾞﾊﾞｯｸﾞと中断) (Alt+F5)を使ってﾃﾞﾊﾞｯｸﾞ作業を始めてください。

ﾃﾞｰﾀ ﾎﾟｰﾘﾝｸﾞはSRAM位置で動き、故に変数がSRAMに置かれた場所を見つけ出すのにAtmel Studioの監視(Watch)ｳｨﾝﾄﾞｳを使うことが必要です。

すべきこと:
- ui.cに追加された2つの全域変数の位置を突き止めてください。
- 各変数を右ｸﾘｯｸしてAdd to Watch(監視に追加)を選んでください。
- それの位置を見つけるために監視ｳｨﾝﾄﾞｳで各変数のtype(型)領域を調査してください。

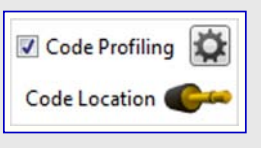
Watch 1

Name	Value	Type
read_count	0	volatile uint32_t(static storage at address 0x200000e8.)
write_count	0	volatile uint32_t(static storage at address 0x200000ec.)

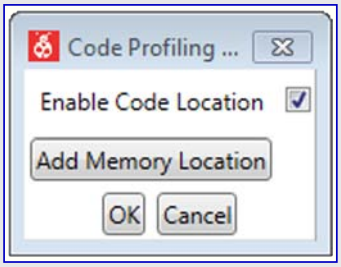
Code Profiling(ｺｰﾄﾞ特性分析)ｲﾝﾀｰﾌｪｰｽを構成設定して2つの変数を図表に接続するためにﾃﾞｰﾀ可視器(Data Visualizer)へ切り替え戻してください。

4.2.12.1. ｺｰﾄﾞ特性分析構成設定ﾀﾞｲｱﾛｸﾞを開く

すべきこと:
- DGI Control Panel(DGI制御盤)内のCode Profiling(ｺｰﾄﾞ特性分析)ｲﾝﾀｰﾌｪｰｽ用ﾁｪｯｸ枠をﾁｪｯｸすることによってｺｰﾄﾞ特性分析ｲﾝﾀｰﾌｪｰｽを許可してください。

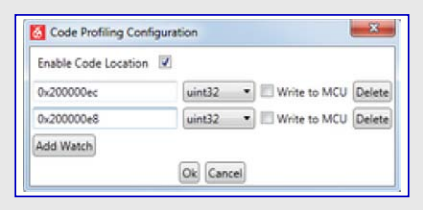


すべきこと:
- 歯車釦を押すことによってCode Profiling Configuration(ｺｰﾄﾞ特性分析構成設定)ｳｨﾝﾄﾞｳを開いてください。



4.2.12.2. ﾃﾞｰﾀ ﾎﾟｰﾘﾝｸﾞ位置を追加

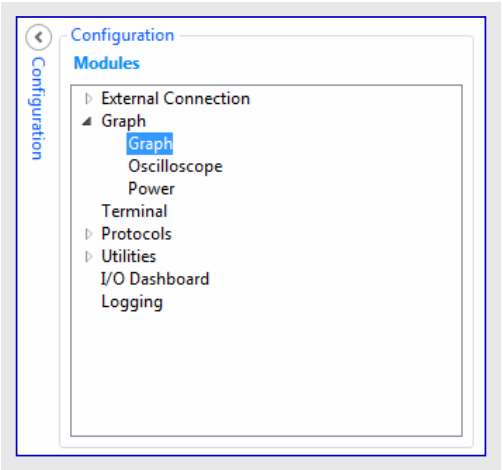
すべきこと:
- 追加される各ﾒﾓﾘ位置に対してAdd memory Loacation(ﾒﾓﾘ位置追加)釦を押してください。
- 各位置のｱﾄﾞﾚｽと形式を満たしてください。



4.2.12.3. 図で変数表示

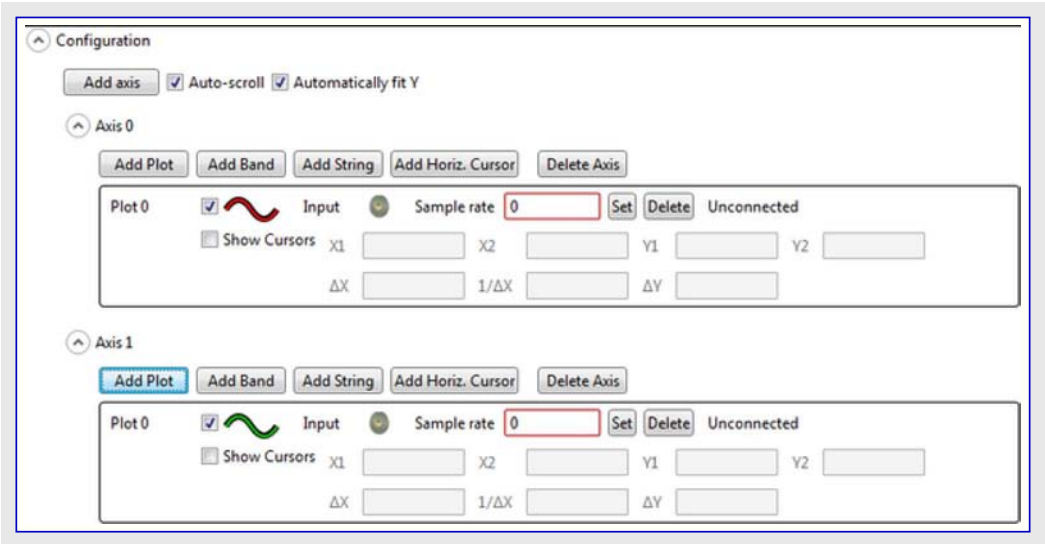
今や監視するための2つの変数を持ち、それらを図で表示することが必要です。

すべきこと:
- ﾃﾞｰﾀ可視器(Data Visualizer)でConfiguration(構成設定)盤を開いてください。
- Graph(図)単位部をﾀﾞﾌﾞﾙ ｸﾘｯｸすることによって図を追加してください。



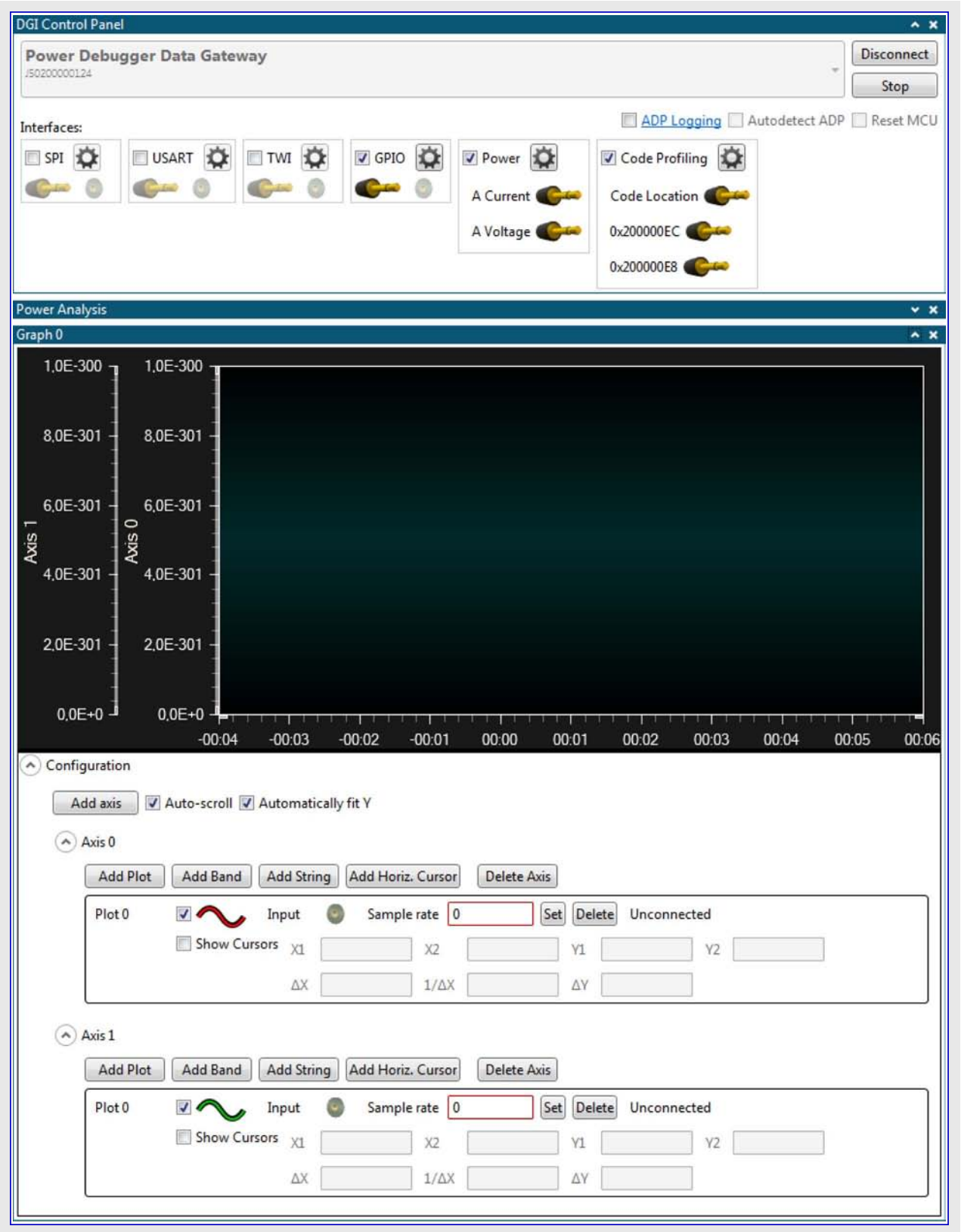
1つの構成設定したY軸でPlot(図式)要素が開きます。けれども、監視のための2つの無関係な変数があり、故に2つの軸が必要です。

すべきこと:
- 更なる軸を追加するためにAdd axis(軸追加)釦をｸﾘｯｸしてください。
- Axis(軸)の各々に対して、Add Plot(描画追加)釦をｸﾘｯｸしてください。



今や、供給元(変数)と吸い込み先(軸)を持ち、故にそれは単にそれらを共に繋げるだけです。

すべきこと: • Code Profiling(ｺｰﾄﾞ特性分析)ｲﾝﾀｰﾌｪｰｽのSource(供給元)ﾌﾟﾗｸﾞの各々をPlot(図式)の入力(sink)ｼﾞｬｯｸに引き摺ってください。



すべきこと: 実行を再開するためにAtmel StudioでContinue(継続) (F5)をｸﾘｯｸしてください。

助言: 停止(HALT)状態のUSB装置はもはやWindows事象に応答せず、長すぎる間この状態を保たれる場合にﾊﾞｽから切断されるかもしれません。これを救済するにはAtmel Studioで単に実行をﾘｾｯﾄしてください。

ﾃﾞｰﾀ可視器(Data Visualizer)の図での出力を考察してください。ﾃﾞｨｽｸをﾌｫｰﾏｯﾄして書き込み周回計数器がどう増えるかを監視してください。両方の値が独立した軸で描かれ、故にそれらはそれによって大きさ調整することができます。出力はこのようなものに見えるでしょう。

4.2.13. 計器盤制御を使う応用の相互作用

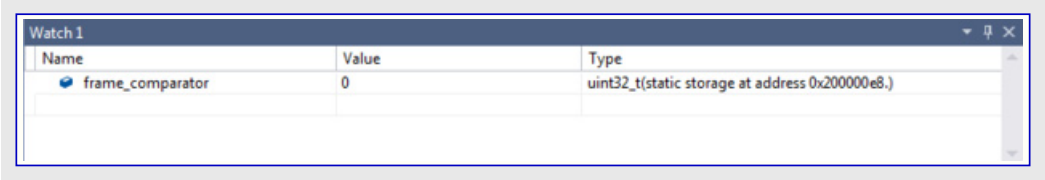
次に、ﾃﾞｰﾀ可視器(Data Visualizer)の計器盤に置かれた部品が応用内の変数にどう繋ぐことができるのかと、従って走行時に計器盤が応用とどう相互作用し得るかを見ます。

予め定義された1000 USB同期ﾊﾟﾙｽ(1秒)の代わりに、元のｺｰﾄﾞに変数比較参照を追加します。

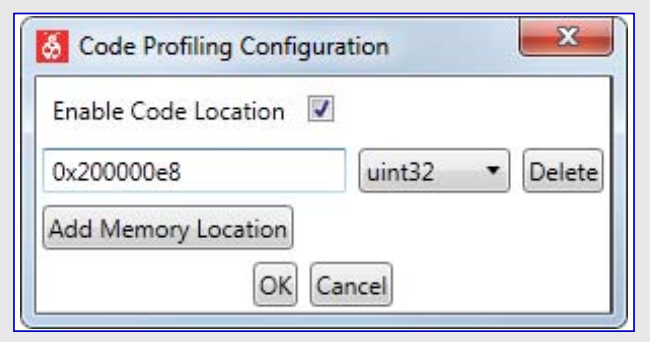
すべきこと: ここで示されるようにui_process()処理部にLED点滅部を含めるようにui.cを変更してください。

```
volatile uint32_t frame_comparator = 100;
volatile uint32_t frames_received = 0;
void ui_process(uint16_t framenumber)
{    frames_received++;
     if (frames_received >= frame_comparator) {
          LED_Toggle(LED_0_PIN);
          frames_received = 0;
     }
}
```

すべきこと:
- ﾌﾟﾛｼﾞｪｸﾄ/解決策を構築(F7)してください。
- Start Debugging and break(ﾃﾞﾊﾞｯｸﾞ開始と中断) (Alt+F5)を使ってﾃﾞﾊﾞｯｸﾞ作業を始めてください。
- uint32_t frame_comparator変数の位置を見つけてください。



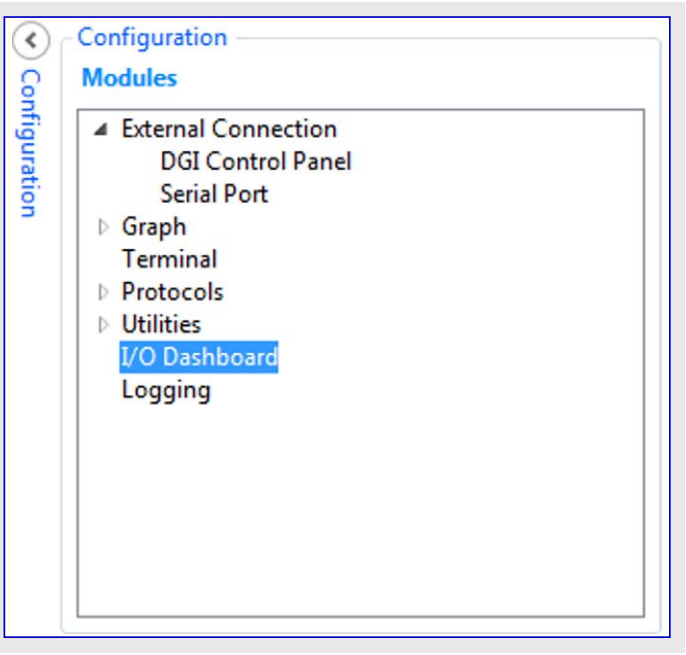
すべきこと:
- ﾃﾞｰﾀ可視器(Data Visualizer)を開いてください。
- 接続してください。
- Code Profiling Configuration(ｺｰﾄﾞ特性分析構成設定)ｳｨﾝﾄﾞｳでframe_comparatorの位置を追加してください。



ﾃﾞｰﾀ可視器の計器盤は今やこの変数の値を操作する制御部を含めて作ることができます。

4.2.13.1. 入出力計器盤を追加

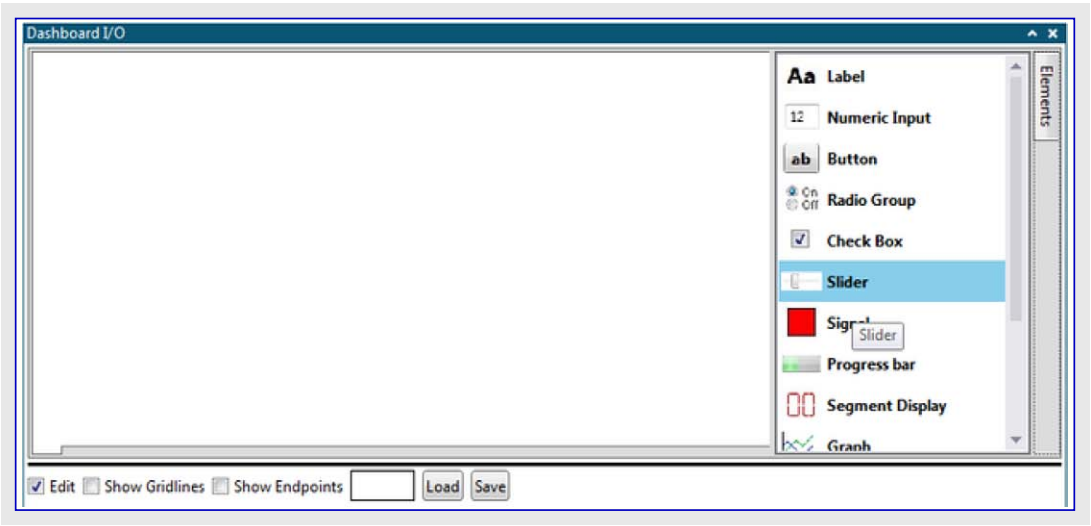
すべきこと:
- Configuration(構成設定)盤を開いてください。
- I/O Dashboard(入出力計器盤)単位部をﾀﾞﾌﾞﾙｸﾘｯｸすることによって新しい入出力計器盤部品を追加してください。



4.2.13.2. 計器盤を構成設定

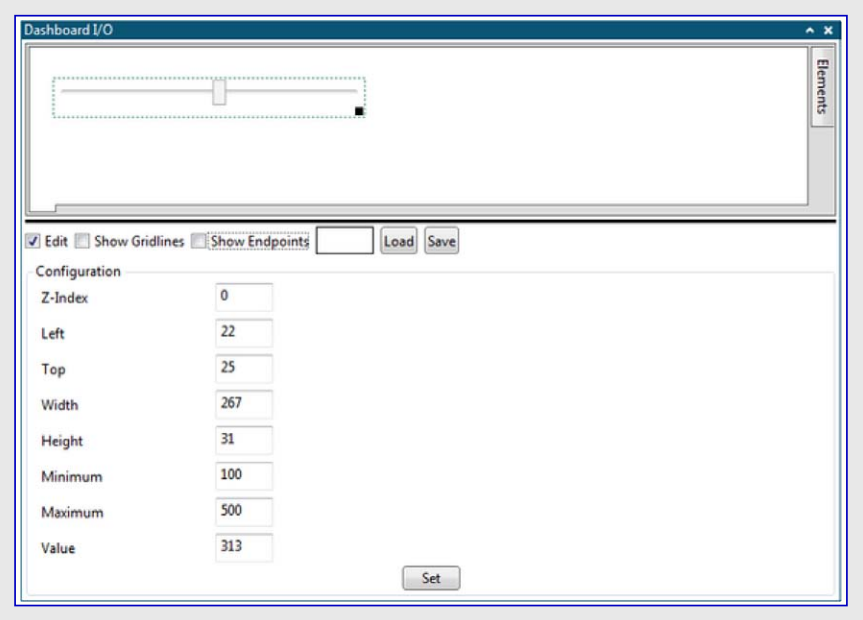
今や計器盤へSlider(摺動子)制御部を追加することができます。

すべきこと:
- Edit(編集)ﾁｪｯｸ枠をﾁｪｯｸしてください。
- Elements(要素)ﾀﾌﾞを開いてください。
- 計器盤にSlider(摺動子)要素を引き摺ってください。



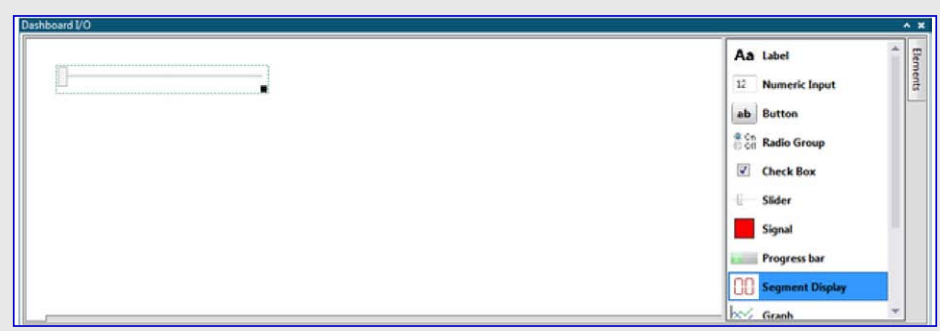
Slider(摺動子)制御部はいくつかの構成設定ﾊﾟﾗﾒｰﾀを持つことが必要です。

すべきこと: Slider(摺動子)要素を選択してそれの特性を設定してください。
- Maximum(最大)=500
- Minimum(最小)=100



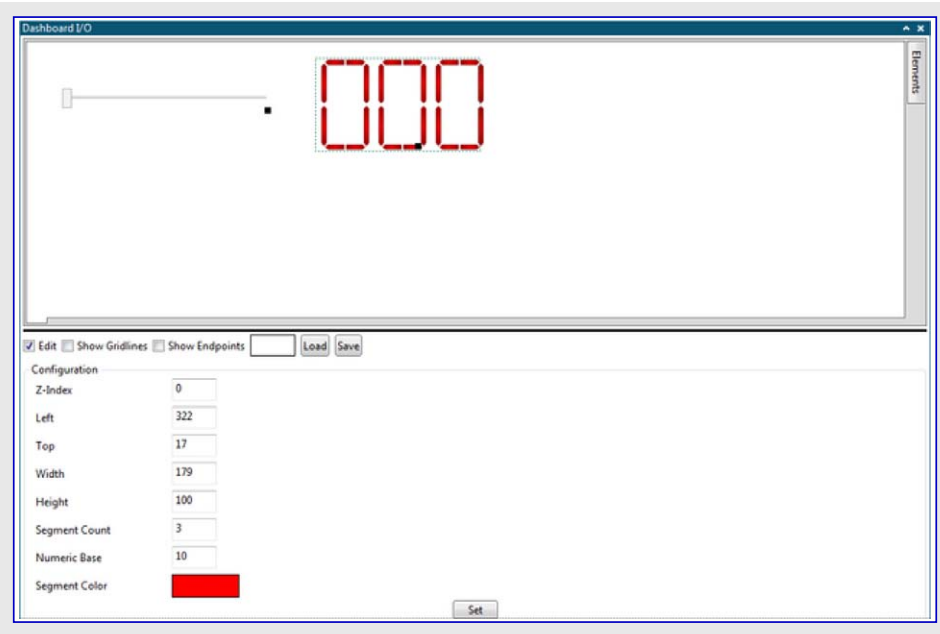
今や、計器盤にSegment Display(7ｾｸﾞ表示)制御部を追加することができます。

すべきこと:
- Edit(編集)ﾁｪｯｸ枠をﾁｪｯｸしてください。
- Elements(要素)ﾀﾌﾞを開いてください。
- 計器盤にSegment Display(7ｾｸﾞ表示)要素を引き摺ってください。



Segment Display(7ｾｸﾞ表示)制御部はいくつかの構成設定ﾊﾟﾗﾒｰﾀを持つことが必要です。

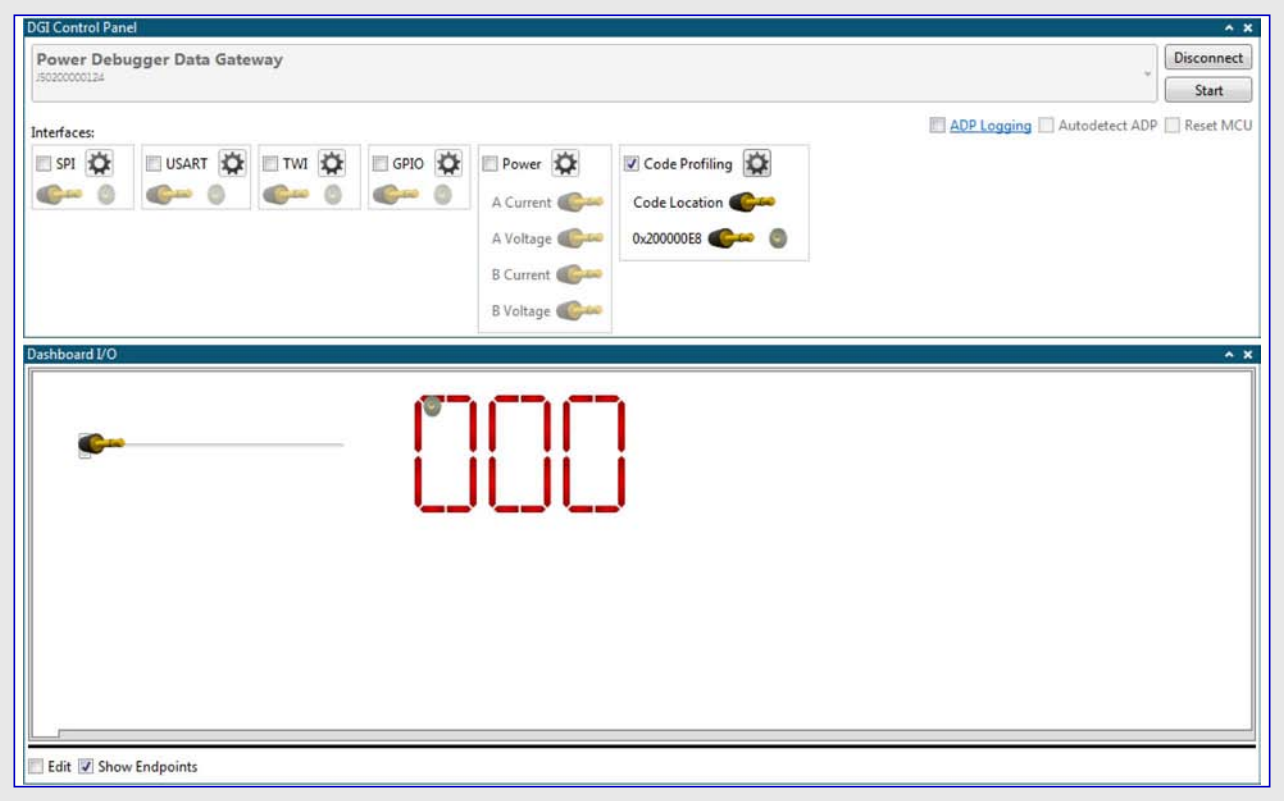
すべきこと: Segment Display(7ｾｸﾞ表示)要素を選択してそれの特性を設定してください。
- Segment Count(7ｾｸﾞ桁数)=3



Slider(摺動子)制御部は今やﾃﾞｰﾀ可視器(Data Visualizer)の何れかの適切な吸い込み先に接続することができる供給元として使うことができます。Segment Display(7ｾｸﾞ表示)は同様に何れかの適切な供給元に接続するために吸い込み先として使うことができます。

ｺｰﾄﾞ特性分析(Code profiling)ﾃﾞｰﾀ ﾎﾟｰﾘﾝｸﾞ ｲﾝﾀｰﾌｪｰｽは供給元と吸い込み先の両方のﾃﾞｰﾀを提供します。今や吸い込み先に対して摺動子、供給元に対して7ｾｸﾞ表示を接続することができます。

すべきこと:
- Edit(編集)ﾁｪｯｸ枠のﾁｪｯｸを外してください。
- Show Endpoints(評価項目表示)ﾁｪｯｸ枠をﾁｪｯｸしてください。
- 各Source(供給元)ﾌﾟﾗｸﾞを引き摺ってSink(吸い込み先)でそれを落とすことによって供給元を吸い込み先に接続してください。

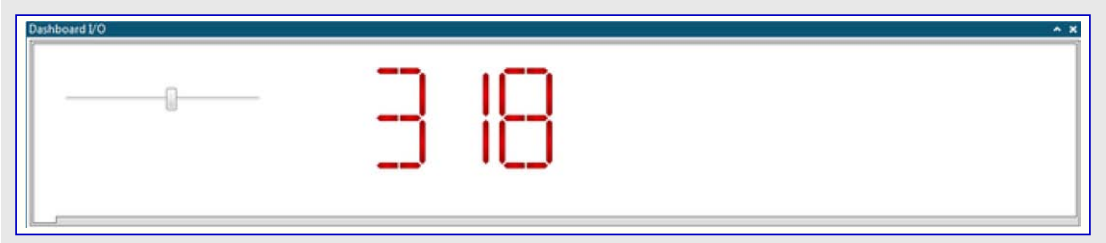


4.2.13.3. 走行

今やﾃﾞｰﾀ可視器(Data Visualizer)で接続が作られたので、ｼｽﾃﾑを走行状態に置いてGUIを通して変数との相互作用をすることができます。

すべきこと:
- Show Endpoints(評価項目表示)ﾁｪｯｸ枠のﾁｪｯｸを外してください。
- ﾃﾞｰﾀ可視器(Data Visualizer)を開始してください。
- Atmel Studioで実行を再開(F5)してください。

摺動子(Slider)は今や応用ｺｰﾄﾞ内のframe_comparator変数の制御部です。摺動子を引き摺り、LED点滅周波数が変わることに注目してください。摺動子位置でのどの変化もﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽを通して目的対象ﾃﾞﾊﾞｲｽに送られ、新しい値が変数に格納されます。また同時にこの値は目的対象から読み戻され、7ｾｸﾞ表示(Segment Display)で表示されます。



5. チップ上デバッグ

5.1. 序説

チップ上デバッグ

チップ上デバッグ単位部は通常、デバッガまたはデバッグ アダプタとして知られる装置を通して外部開発基盤からデバイスでの実行を監視して制御することを開発者に許すシステムです。

OCDシステムとで応用は目的対象システムで正確な電気的及びタイミングの特性を保ちつつ実行することができる一方で、条件付きや手動での実行停止とプログラムの流れとメモリの調査を行うことができます。

走行動作

走行動作時、コードの実行はPower Debuggerと完全に独立です。Power Debuggerは中断条件が発生したかを知るために目的対象デバイスを継続的に監視します。これが発生すると、OCDシステムはデバッグ インターフェースを通してデバイスに質問し、使用者にデバイスの内部状況を見ることを許します。

停止動作

中断点(ブレークポイント)到達時、プログラム実行は停止されますが、いくつかのI/O(周辺機能)は中断点が起きなかったように走行を継続するかもしれません。例えば中断点発生時に丁度USART送信が初期化(/設定)されたと仮定します。この場合は例えコアが停止動作であっても、USARTは送信を完了する全速で走行を継続します。

ハードウェア中断点

目的対象OCD部はハードウェアで実装された多数のプログラム カウンタ比較器を含みます。プログラム カウンタが比較器レジスタの1つに格納された値と一致すると、OCDは停止動作へ移行します。ハードウェア中断点はOCD部で専用のハードウェアを必要とするため、利用可能な中断点数は目的対象AVRに実装されるOCD部の大きさに依存します。通常、デバッガによって内部使用のために1つのこのようなハードウェア比較器が”予約”されます。

ソフトウェア中断点

ソフトウェア中断点は目的対象デバイス上のプログラム メモリに配置されたBREAK命令です。この命令が読み込まれると、プログラム実行が中断され、OCDは停止動作へ移行します。実行を継続するには、OCDから”start”命令を与えなければなりません。全てのAtmelデバイスがBREAK命令を支援するOCD部を持っている訳ではありません。

5.2. JTAG/SWDを持つSAMデバイス

全てのSAMデバイスはプログラミングとデバッグ用のSWDインターフェースが特徴です。加えて、いくつかのSAMデバイスは同様の機能を持つJTAGインターフェースも特徴です。そのデバイスで支援されるインターフェースについてはデバイスのデータシートを調べてください。

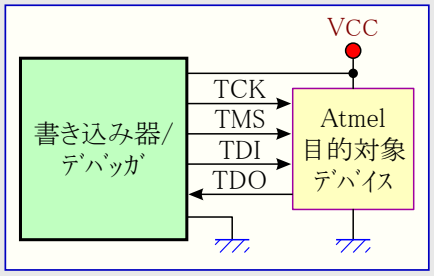
5.2.1. ARM CoreSight構成部

Atmel ARM Cortex-Mに基づくマイクロ コントローラはCoreSight準拠OCD構成部を実装します。これらの構成部の機能はデバイス毎に変わり得ます。更なる情報についてはデバイスのデータシートだけでなく、ARMによって提供されるCoreSight資料も調べてください。

5.2.2. JTAG物理インターフェース

JTAGインターフェースはIEEE® 1149.1規格に適合する4線検査入出力ポート(TAP:Test Access Port)制御器から成ります。IEEE規格は回路基板の接続性(境界走査)を効率的に検査する工業標準的な方法を提供するために開発されました。AtmelのAVRとSAMのデバイスは完全なプログラミングとチップ上デバッグの支援を含むように拡張されたこの機能を持ちます。

図5-1. JTAGインターフェースの基本



5.2.2.1. SAM JTAGピン配列 (Cortex-Mデバッグ コネクタ)

JTAGインターフェースを持つAtmel SAMを含む応用PCB設計時は右図で示されるようなピン配列を使うことが推奨されます。特定のキットで含まれるケーブルとアダプタに応じて、このピン配列の100milと50milの両変種が支援されます。

図5-2. SAM JTAGヘッダ ピン配列

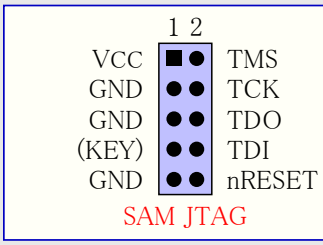


表5-1. SAM JTAGﾋﾟﾝ説明

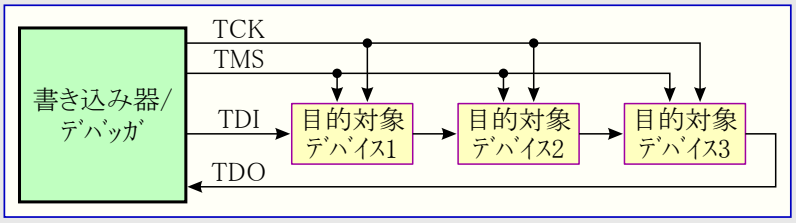
名前	ﾋﾟﾝ番号	説明
TCK	4	検査ｸﾛｯｸ (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへのｸﾛｯｸ信号)
TMS	3	検査種別選択 (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへの制御信号)
TDI	8	検査ﾃﾞｰﾀ入力 (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへ送出されるﾃﾞｰﾀ)
TDO	6	検査ﾃﾞｰﾀ出力 (目的対象ﾃﾞﾊﾞｲｽからPower Debuggerへ送出されるﾃﾞｰﾀ)
nRESET	10	ﾘｾｯﾄ (任意)。目的対象ﾃﾞﾊﾞｲｽのﾘｾｯﾄに使用。或る筋書でのﾃﾞﾊﾞｯｸﾞを必要とし得る、目的対象ﾃﾞﾊﾞｲｽをﾘｾｯﾄ状態で保持することをPower Debuggerに許すために、このﾋﾟﾝの接続が推奨されます。
VTG	1	目的対象電圧基準。Power Debuggerは正しくﾚﾍﾞﾙ変換器を給電するために、このﾋﾟﾝで目的対象ﾃﾞﾊﾞｲｽ電圧を採取します。Power Debuggerはこの動作でこのﾋﾟﾝから1mA未満を引き出します。
GND	3,5,9	接地。Power Debuggerと目的対象ﾃﾞﾊﾞｲｽが同じ接地基準を共用するのを保証するために全てが接続されなければなりません。
KEY	7	内部的にAVRｺﾈｸﾀのTRSTﾋﾟﾝに接続。無接続としてが推奨されます。

助言: 1番ﾋﾟﾝとGND間に雑音分離(ﾃﾞｶｯﾌﾟ)ｺﾝﾃﾞﾝｻを含むことを忘れないでください。

5.2.2.2. JTAGﾃﾞｨｰｼﾞｰ ﾁｪｰﾝ

JTAGｲﾝﾀｰﾌｪｰｽは多数のﾃﾞﾊﾞｲｽに対してﾃﾞｨｰｼﾞｰ ﾁｪｰﾝ構成設定内で単一ｲﾝﾀｰﾌｪｰｽへ接続することを許します。目的対象ﾃﾞﾊﾞｲｽは全てが同じ供給電圧によって給電され、共通接地節を共用しなければならず、右図で示されるように接続されなければなりません。

図5-3. JTAGﾃﾞｨｰｼﾞｰ ﾁｪｰﾝ



ﾃﾞｨｰｼﾞｰ ﾁｪｰﾝでのﾃﾞﾊﾞｲｽ接続時、以下の点が考慮されなければなりません。

- 全てのﾃﾞﾊﾞｲｽはPower Debugger探針のGNDに接続された共通接地(GND)を共用しなければなりません。
- 全てのﾃﾞﾊﾞｲｽは同じ目的対象電圧で動作しなければなりません。Power DebuggerのVTGはこの電圧に接続されなければなりません。
- TMSとTCKは並列で接続されます。TDIとTDOは直列連鎖で接続されます。
- ﾁｪｰﾝ内のﾃﾞﾊﾞｲｽのどれかがそれのJTAGﾎﾟｰﾄを禁止する場合、Power Debugger探針のnSRSTはﾃﾞﾊﾞｲｽのRESETに接続されなければなりません。
- ”Devices before”はTDI信号が目的対象ﾃﾞﾊﾞｲｽに到達する前にﾃﾞｨｰｼﾞｰ ﾁｪｰﾝ内を通って通過しなければならないJTAGﾃﾞﾊﾞｲｽ数を参照します。同様に”Devices after”は信号がPower DebuggerのTDOﾋﾟﾝに到達する前に目的対象ﾃﾞﾊﾞｲｽの後を通って通過しなければならないﾃﾞﾊﾞｲｽ数です。
- ”Instruction bits before”と”Instruction bits after”はﾃﾞｨｰｼﾞｰ ﾁｪｰﾝ内で目的対象ﾃﾞﾊﾞｲｽの前後に接続される全てのJTAGﾃﾞﾊﾞｲｽの命令ﾚｼﾞｽﾀ(IR)長の総合計を示します。
- 総IR長(Instruction bits before+Atmel目的対象ﾃﾞﾊﾞｲｽIR長+Instruction bits after)は最大256ﾋﾞｯﾄに制限されます。ﾁｪｰﾝ内のﾃﾞﾊﾞｲｽ数は前が15、後ろが15に制限されます。

助言: ﾃﾞｨｰｼﾞｰ ﾁｪｰﾝ例 : TDI ⇒ ATmega1280 ⇒ ATxmega128A1 ⇒ ATUC3A0512 ⇒ TDO

Atmel AVR XMEGA®ﾃﾞﾊﾞｲｽに接続するためのﾃﾞｨｰｼﾞｰ ﾁｪｰﾝ設定は次のとおりです。

- Devices before : 1
- Devices after : 1
- Instruction bits before : 4 (8ﾋﾞｯﾄ AVRﾃﾞﾊﾞｲｽは4ﾋﾞｯﾄのIRを持ちます。)
- Instruction bits after : 5 (32ﾋﾞｯﾄ AVRﾃﾞﾊﾞｲｽは5ﾋﾞｯﾄのIRを持ちます。)

表5-2. Atmel MCUのIR長

ﾃﾞﾊﾞｲｽ型	IR長
8ﾋﾞｯﾄ AVR	4ﾋﾞｯﾄ
32ﾋﾞｯﾄ AVR	5ﾋﾞｯﾄ
SAM	4ﾋﾞｯﾄ

5.2.3. JTAG目的対象への接続

Power Debuggerは50mil10ﾋﾟﾝJTAGｺﾈｸﾀと共に出荷されます。AVR JTAGﾍｯﾀﾞとARM Cortexﾃﾞﾊﾞｯｸﾞ ﾍｯﾀﾞの両ｺﾈｸﾀが直接電気的に接続されますが、2つの異なるﾋﾟﾝ配列に適応します。ｺﾈｸﾀは目的対象MCU型ではなく、目的対象基板のﾋﾟﾝ配列に基づいて選ばれるべきで、例えば、AVR STK600階層で実装されたSAMﾃﾞﾊﾞｲｽはAVRﾍｯﾀﾞを使うべきです。

加えてPower DebuggerはそれのPCB上のAVRﾋﾟﾝ配列で未実装の100mil 10ﾋﾟﾝJTAGｺﾈｸﾀも持ちます。これは50mil AVRﾍｯﾀﾞに直接的に配線されます。

10ﾋﾟﾝAVR JTAGｺﾈｸﾀ用の推奨ﾋﾟﾝ配列は図5-6.で示されます。

10ﾋﾟﾝARM Cortexﾃﾞﾊﾞｯｸﾞ ｺﾈｸﾀ用の推奨ﾋﾟﾝ配列は図5-2.で示されます。

標準10ﾋﾟﾝ50milﾍｯﾀﾞへの直接接続

このﾍｯﾀﾞ形式を支援する基板への直接接続するには(いくつかのｷｯﾄに含まれる)50mil 10ﾋﾟﾝ ﾌﾗｯﾄ ｹｰﾌﾞﾙを使ってください。AVRﾋﾟﾝ配列を持つﾍｯﾀﾞに対してはPower Debugger上のAVRｺﾈｸﾀ ﾎﾟｰﾄを、ARM Cortexﾃﾞﾊﾞｯｸﾞ ﾍｯﾀﾞ ﾋﾟﾝ配列に従うﾍｯﾀﾞに対してはSAMｺﾈｸﾀ ﾎﾟｰﾄを使ってください。

両10ﾋﾟﾝ ｺﾈｸﾀ ﾎﾟｰﾄ用のﾋﾟﾝ配列は下で示されます。

標準10ﾋﾟﾝ100milﾍｯﾀﾞへの接続

100milﾍｯﾀﾞへ接続するには標準50mil-100milｱﾀﾞﾌﾟﾀを使ってください。(いくつかのｷｯﾄに含まれる)ｱﾀﾞﾌﾟﾀ基板はこの目的に使うことができ、代わりにAVR目的対象についてはJTAGICE3ｱﾀﾞﾌﾟﾀを使うことができます。

重要: JTAGICE3 100milｱﾀﾞﾌﾟﾀはｱﾀﾞﾌﾟﾀで2番ﾋﾟﾝと10番(AVR GND)ﾋﾟﾝが接続されているためSAMｺﾈｸﾀ ﾎﾟｰﾄで使うことはできません。

独自100milﾍｯﾀﾞへの接続

目的対象基板が50または100milの適合10ﾋﾟﾝJTAGﾍｯﾀﾞを持たない場合、(いくつかのｷｯﾄで含まれる)10ﾋﾟﾝ”ﾊﾞﾗ線”ｹｰﾌﾞﾙを使って独自ﾋﾟﾝ配列を割り当てることができ、これは10個の個別100milｿｹｯﾄへの入出力を与えます。

20ﾋﾟﾝ100milﾍｯﾀﾞへの接続

20ﾋﾟﾝ100milﾍｯﾀﾞを持つ目的対象へ接続するには(いくつかのｷｯﾄに含まれる)ｱﾀﾞﾌﾟﾀ基板を使ってください。

表5-3. Power Debugger JTAGﾋﾟﾝ説明

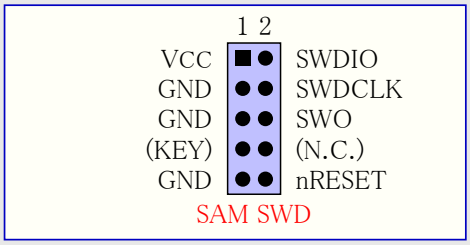
名前	ﾎﾟｰﾄのﾋﾟﾝ番号		説明
	AVR	SAM	
TCK	1	4	検査ｸﾛｯｸ (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへのｸﾛｯｸ信号)
TMS	5	2	検査種別選択 (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへの制御信号)
TDI	9	8	検査ﾃﾞｰﾀ入力 (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへ送出されるﾃﾞｰﾀ)
TDO	3	6	検査ﾃﾞｰﾀ出力 (目的対象ﾃﾞﾊﾞｲｽからPower Debuggerへ送出されるﾃﾞｰﾀ)
nTRST	8	-	検査ﾘｾｯﾄ(任意、いくつかのAVRﾃﾞﾊﾞｲｽのみ)。JTAG TAP制御器のﾘｾｯﾄに使用
nSRST	6	10	ﾘｾｯﾄ (任意)。目的対象ﾃﾞﾊﾞｲｽのﾘｾｯﾄに使用。或る筋書でのﾃﾞﾊﾞｯｸﾞを必要とし得る、目的対象ﾃﾞﾊﾞｲｽをﾘｾｯﾄ状態で保持することをPower Debuggerに許すために、このﾋﾟﾝの接続が推奨されます。
VTG	4	1	目的対象基準電圧。Power Debuggerは正しくﾚﾍﾞﾙ変換器を給電するために、このﾋﾟﾝで目的対象電圧を採取します。Power DebuggerはこのﾋﾟﾝからﾃﾞﾊﾞｯｸﾞWIRE動作で3mA未満、他の動作で1mA未満を引き出します。
GND	2,10	3,5,9	接地。Power Debuggerと目的対象ﾃﾞﾊﾞｲｽが同じ接地基準を共用するのを保証するために全てが接続されなければなりません。

5.2.4. SWD物理ｲﾝﾀｰﾌｪｰｽ

ARM SWDｲﾝﾀｰﾌｪｰｽはJTAGｲﾝﾀｰﾌｪｰｽの一部分で、TCKとTMSのﾋﾟﾝを利用します。けれどもARM JTAGとAVR JTAGのｺﾈｸﾀはﾋﾟﾝ互換ではなく、故にSWDまたはJTAGのｲﾝﾀｰﾌｪｰｽを持つSAMﾃﾞﾊﾞｲｽを使う応用PCBを設計する時に右図で示されるARMﾋﾟﾝ配列を使うことが推奨されます。Power DebuggerのASMｺﾈｸﾀ ﾎﾟｰﾄはこのﾋﾟﾝ配列に直接接続することができます。

Power DebuggerはUART形式ITM追跡をﾎｽﾄ ｺﾝﾋﾟｭｰﾀへ流す能力があります。追跡は10ﾋﾟﾝ ﾍｯﾀﾞのTRACE/SWO(JTAGのTDO)ﾋﾟﾝで捕獲されます。ﾃﾞｰﾀはPower Debuggerで内部的に緩衝され、HIDｲﾝﾀｰﾌｪｰｽ上でﾎｽﾄ ｺﾝﾋﾟｭｰﾀへ送られます。信頼できる最大ﾃﾞｰﾀ速度は約3MB/sです。

図5-4. 推奨ARM SWDﾍｯﾀﾞ ﾋﾟﾝ配列



5.2.5. SWD目的対象への接続

ARM SWDインターフェースはTCKとTMSのピンを利用するJTAGインターフェースの一部分で、SWDデバイスに接続する時に技術的に10ピンJTAGコネクタを使うことができることを意味します。けれどもARM JTAGとAVR JTAGのコネクタはピン互換ではなく、故にこれは使う目的対象基板の配置に依存します。STK600使用またはAVR JTAGピン配列を利用する基板の時、Power DebuggerのAVRコネクタ ポートが使われなければなりません。ARM JTAGピン配列を利用する基板への接続時、Power DebuggerのSAMコネクタ ポートが使われなければなりません。

10ピンCortexデバッグ コネクタ用の推奨ピン配列は図5-4.で示されます。

10ピン50mil Cortexヘッダへの接続

標準50mil Cortexヘッダへ接続するには(いくつかのキットに含まれる)フラット ケーブルを使ってください。

10ピン100mil Cortexピン配列ヘッダへの接続

100mil Cortexピン配列ヘッダへ接続するには(いくつかのキットに含まれる)アダプタ基板を使ってください。

20ピン100mil SAMヘッダへの接続

20ピン100mil SAMヘッダへ接続するには(いくつかのキットに含まれる)アダプタ基板を使ってください。

独自100milヘッダへの接続

Power DebuggerのAVRまたはSAMコネクタ ポートと目的対象基板を接続するには10ピンのミニ バラ線ケーブルが使われるべきです。下表で記述されるように6つの接続が必要とされます。

表5-4. Power Debugger SWDピン割り当て

名前	ポートのピン番号		説明
	AVR	SAM	
SWDCLK	1	4	直列線デバッグ(SWD:Serial Wire Debug)クロック
SWDIO	5	2	直列線デバッグ データ入出力
SWO	3	6	直列線デバッグ出力 (任意-全てのデバイスで実装されるとは限りません。)
nSRST	6	10	リセット
VTG	4	1	目的対象電圧基準
GND	2,10	3,5	接地

5.2.6. 特別な考慮

ERASEピン

いくつかのSAMデバイスは保護ビットが設定されたデバイスを完全にチップ消去して解錠を実行することを明示されるERASEピンを含みます。この機能はデバイス自身とフラッシュ制御器に結合され、ARMコアの一部ではありません。

ERASEピンはどのデバッグ ヘッダの一部でもなく、従ってPower Debuggerはデバイスを解錠するためにこの信号を活性にすることができません。このような場合、デバッグ作業を開始する前に手動で消去を実行すべきです。

物理インターフェース

JTAGインターフェース

RESET線はPower DebuggerがJTAGインターフェースを許可することができるよう、常に接続されるべきです。

SWDインターフェース

RESET線はPower DebuggerがSWDインターフェースを許可することができるよう、常に接続されるべきです。

5.3. JTAG/aWireを持つAVR UC3デバイス

全てのAVR UC3デバイスはプログラミングとデバッグ用のJTAGインターフェースが特徴です。加えて、いくつかのAVR UC3デバイスは単線を使って同様の機能を持つaWireインターフェースも特徴です。そのデバイスで支援されるインターフェースについてはデバイスのデータシートを調べてください。

5.3.1. Atmel AVR UC3チップ上デバッグ システム

Atmel AVR UC3 OCDシステムは高い柔軟性と強力で開かれた32ビット マイクロ コントローラ用のチップ上デバッグ規格であるNexus 2.0規格(IEEE-ISTO 5001™-2003)に従って設計されています。これは以下の機能を支援します。

- Nexus適合デバッグ解決策
- どのCPU速度も支援するOCD
- 6つのプログラム カウンタ ハードウェア中断点(ブレーク ポイント)
- 2つのデータ中断点
- 監視点として構成設定できる中断点
- 範囲での中断を与えるように結合することができるハードウェア中断点

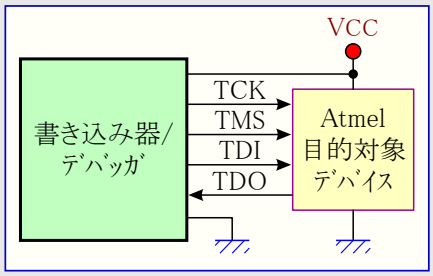
・(BREAK命令を使う)無制限数の使用者ﾌﾟﾛｸﾞﾗﾑ中断点
・(並列追跡捕獲ﾎﾟｰﾄを持つﾃﾞﾊﾞｯｶﾞによってのみ支援される)実時間ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ分岐追跡、ﾃﾞｰﾀ追跡、処理追跡

AVR UC3 OCDｼｽﾃﾑに関するより多くの情報についてはwww.atmel.com/uc3に置かれているAVR32UC技術参考書を調べてください。

5.3.2. JTAG物理ｲﾝﾀｰﾌｪｰｽ

JTAGｲﾝﾀｰﾌｪｰｽはIEEE® 1149.1規格に適合する4線検査入出力ﾎﾟｰﾄ(TAP:Test Access Port)制御器から成ります。IEEE規格は回路基板の接続性(境界走査)を効率的に検査する工業標準的な方法を提供するために開発されました。AtmelのAVRとSAMのﾃﾞﾊﾞｲｽは完全なﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾁｯﾌﾟ上ﾃﾞﾊﾞｯｸﾞの支援を含むように拡張されたこの機能を持ちます。

図5-5. JTAGｲﾝﾀｰﾌｪｰｽの基本



5.3.2.1. AVR JTAGﾋﾟﾝ配列

JTAGｲﾝﾀｰﾌｪｰｽを持つAtmel AVRを含む応用PCB設計時は右図で示されるようなﾋﾟﾝ配列を使うことが推奨されます。特定のｷｯﾄで含まれるｹｰﾌﾞﾙとｱﾀﾞﾌﾟﾀに応じて、このﾋﾟﾝ配列の100milと50milの両変種が支援されます。

図5-6. AVR JTAGﾍｯﾀﾞ ﾋﾟﾝ配列

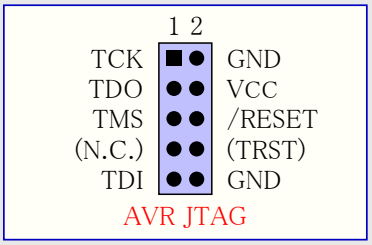


表5-5. Power Debugger JTAGﾋﾟﾝ説明

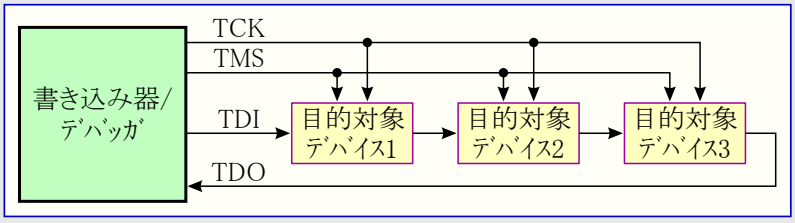
名前	ﾎﾟｰﾄのﾋﾟﾝ番号		説明
	AVR	SAM	
TCK	1	4	検査ｸﾛｯｸ (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへのｸﾛｯｸ信号)
TMS	5	2	検査種別選択 (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへの制御信号)
TDI	9	8	検査ﾃﾞｰﾀ入力 (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへ送出されるﾃﾞｰﾀ)
TDO	3	6	検査ﾃﾞｰﾀ出力 (目的対象ﾃﾞﾊﾞｲｽからPower Debuggerへ送出されるﾃﾞｰﾀ)
nTRST	8	-	検査ﾘｾｯﾄ(任意、いくつかのAVRﾃﾞﾊﾞｲｽのみ)。JTAG TAP制御器のﾘｾｯﾄに使用
nSRST	6	10	ﾘｾｯﾄ (任意)。目的対象ﾃﾞﾊﾞｲｽのﾘｾｯﾄに使用。或る筋書でのﾃﾞﾊﾞｯｸﾞを必要とし得る、目的対象ﾃﾞﾊﾞｲｽをﾘｾｯﾄ状態で保持することをPower Debuggerに許すために、このﾋﾟﾝの接続が推奨されます。
VTG	4	1	目的対象基準電圧。Power Debuggerは正しくﾚﾍﾞﾙ変換器を給電するために、このﾋﾟﾝで目的対象電圧を採取します。Power DebuggerはこのﾋﾟﾝからﾃﾞﾊﾞｯｸﾞWIRE動作で3mA未満、他の動作で1mA未満を引き出します。
GND	2,10	3,5,9	接地。Power Debuggerと目的対象ﾃﾞﾊﾞｲｽが同じ接地基準を共用するのを保証するために全てが接続されなければなりません。

助言: 4番ﾋﾟﾝとGND間に雑音分離(ﾃﾞｶｯﾌﾟ)ｺﾝﾃﾞﾝｻを含むことを覚えて置いてください。

5.3.2.2. JTAGﾃﾞｨｰｼﾞｰ ﾁｪｰﾝ

JTAGｲﾝﾀｰﾌｪｰｽは多数のﾃﾞﾊﾞｲｽに対してﾃﾞｨｰｼﾞｰﾁｪｰﾝ構成設定内で単一ｲﾝﾀｰﾌｪｰｽへ接続することを許します。目的対象ﾃﾞﾊﾞｲｽは全てが同じ供給電圧によって給電され、共通接地節を共用しなければならず、右図で示されるように接続されなければなりません。

図5-7. JTAGﾃﾞｨｰｼﾞｰ ﾁｪｰﾝ



ﾃﾞｨｰｼﾞｰ ﾁｪｰﾝでのﾃﾞﾊﾞｲｽ接続時、以下の点が考慮されなければなりません。

・全てのﾃﾞﾊﾞｲｽはPower Debugger探針のGNDに接続された共通接地(GND)を共用しなければなりません。

- 全てのﾃﾞﾊﾞｲｽは同じ目的対象電圧で動作しなければなりません。Power DebuggerのVTGはこの電圧に接続されなければなりません。
- TMSとTCKは並列で接続されます。TDIとTDOは直列連鎖で接続されます。
- チェーン内のﾃﾞﾊﾞｲｽのどれかがそれのJTAGﾎﾟｰﾄを禁止する場合、Power Debugger探針のnSRSTはﾃﾞﾊﾞｲｽのRESETに接続されなければなりません。
- ”Devices before”はTDI信号が目的対象ﾃﾞﾊﾞｲｽに到達する前にﾃﾞｨｰｼﾞｰ チェーン内を通って通過しなければならないJTAGﾃﾞﾊﾞｲｽ数を参照します。同様に”Devices after”は信号がPower DebuggerのTDOﾋﾟﾝに到達する前に目的対象ﾃﾞﾊﾞｲｽの後を通って通過しなければならないﾃﾞﾊﾞｲｽ数です。
- ”Instruction bits before”と”Instruction bits after”はﾃﾞｨｰｼﾞｰ チェーン内で目的対象ﾃﾞﾊﾞｲｽの前後に接続される全てのJTAGﾃﾞﾊﾞｲｽの命令ﾚｼﾞｽﾀ(IR)長の総合計を示します。
- 総IR長(Instruction bits before+Atmel目的対象ﾃﾞﾊﾞｲｽIR長+Instruction bits after)は最大256ﾋﾞｯﾄに制限されます。チェーン内のﾃﾞﾊﾞｲｽ数は前が15、後ろが15に制限されます。

助言: ﾃﾞｨｰｼﾞｰ チェーン例：TDI ⇒ ATmega1280 ⇒ ATxmega128A1 ⇒ ATUC3A0512 ⇒ TDO

Atmel AVR XMEGA®ﾃﾞﾊﾞｲｽに接続するためのﾃﾞｨｰｼﾞｰ チェーン設定は次のとおりです。

- Devices before : 1
- Devices after : 1
- Instruction bits before : 4 (8ﾋﾞｯﾄ AVRﾃﾞﾊﾞｲｽは4ﾋﾞｯﾄのIRを持ちます。)
- Instruction bits after : 5 (32ﾋﾞｯﾄ AVRﾃﾞﾊﾞｲｽは5ﾋﾞｯﾄのIRを持ちます。)

表5-6. Atmel MCUのIR長

ﾃﾞﾊﾞｲｽ型	IR長
8ﾋﾞｯﾄ AVR	4ﾋﾞｯﾄ
32ﾋﾞｯﾄ AVR	5ﾋﾞｯﾄ
SAM	4ﾋﾞｯﾄ

5.3.3. JTAG目的対象への接続

Power Debuggerは2つの50mil 10ﾋﾟﾝのJTAGｺﾈｸﾀが装備されます。両ｺﾈｸﾀは電気的に直接的に接続されていますが、AVR JTAGﾍｯﾀﾞとARM Cortexﾃﾞﾊﾞｯｸﾞ ﾍｯﾀﾞの2つの異なるﾋﾟﾝ配列を確認してください。ｺﾈｸﾀは目的対象MCU型式ではなく、目的対象基板のﾋﾟﾝ配列に基づいて選択されるべきです。例えばAVR STK600階層に装着されたSAMﾃﾞﾊﾞｲｽはAVRﾍｯﾀﾞが使われるべきです。

加えてPower DebuggerはそれのPCB上のAVRﾋﾟﾝ配列で未実装の100mil 10ﾋﾟﾝJTAGｺﾈｸﾀも持ちます。これは50mil AVRﾍｯﾀﾞに直接的に配線されます。

10ﾋﾟﾝAVR JTAGｺﾈｸﾀの推奨ﾋﾟﾝ配列は図5-6.で示されます。

10ﾋﾟﾝARM Cortexﾃﾞﾊﾞｯｸﾞ ｺﾈｸﾀの推奨ﾋﾟﾝ配列は図5-2.で示されます。

標準10ﾋﾟﾝ50milﾍｯﾀﾞへの直接接続

このﾍｯﾀﾞ形式を支援する基板へ直接接続するには(いくつかのｷｯﾄに含まれる)50mil 10ﾋﾟﾝ ﾌﾗｯﾄ ｹｰﾌﾞﾙを使ってください。AVRﾋﾟﾝ配列を持つﾍｯﾀﾞについてはPower DebuggerのAVRｺﾈｸﾀ ﾎﾟｰﾄを、ARM Cortexﾃﾞﾊﾞｯｸﾞ ﾍｯﾀﾞに従うﾍｯﾀﾞについてはSAMｺﾈｸﾀ ﾎﾟｰﾄを使ってください。

10ﾋﾟﾝの両ｺﾈｸﾀのﾋﾟﾝ配列は以降で示されます。

標準10ﾋﾟﾝ100milﾍｯﾀﾞへの接続

100milﾍｯﾀﾞに接続するには標準50mil⇒100milｱﾀﾞﾌﾟﾀを使ってください。(いくつかのｷｯﾄに含まれる)ｱﾀﾞﾌﾟﾀ基板がこの目的に使うことができ、または代わりにAVR目的対象に対してJTAGICE3のｱﾀﾞﾌﾟﾀを使うことができます。

重要: ｱﾀﾞﾌﾟﾀの2と10番ﾋﾟﾝ(AVR GND)が接続されるため、JTAGICE3の100milｱﾀﾞﾌﾟﾀはSAMｺﾈｸﾀ ﾎﾟｰﾄで使うことはできません。

独自100milﾍｯﾀﾞへの接続

目的対象基板が50milか100milの適合10ﾋﾟﾝJTAGﾍｯﾀﾞを持たない場合、10個の個別100milｿｹｯﾄへの入出力を与える(いくつかのｷｯﾄに含まれる)10ﾋﾟﾝの”ﾐﾆ ﾊﾞﾗ線”ｹｰﾌﾞﾙを使って独自ﾋﾟﾝ配列に割り当てることができます。

20ﾋﾟﾝ100milﾍｯﾀﾞへの接続

20ﾋﾟﾝ100milﾍｯﾀﾞを持つ目的対象へ接続するには(いくつかのｷｯﾄに含まれる)ｱﾀﾞﾌﾟﾀ基板を使ってください。

表5-7. Power Debugger JTAGﾋﾟﾝ説明

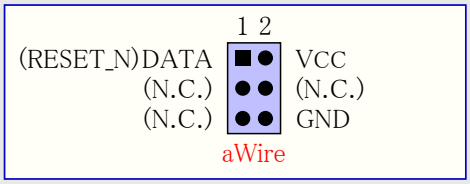
名前	ﾎﾟｰﾄのﾋﾟﾝ番号		説明
	AVR	SAM	
TCK	1	4	検査ｸﾛｯｸ (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへのｸﾛｯｸ信号)
TMS	5	2	検査種別選択 (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへの制御信号)
TDI	9	8	検査ﾃﾞｰﾀ入力 (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへ送出されるﾃﾞｰﾀ)
TDO	3	6	検査ﾃﾞｰﾀ出力 (目的対象ﾃﾞﾊﾞｲｽからPower Debuggerへ送出されるﾃﾞｰﾀ)
nTRST	8	-	検査ﾘｾｯﾄ(任意、いくつかのAVRﾃﾞﾊﾞｲｽのみ)。JTAG TAP制御器のﾘｾｯﾄに使用
nSRST	6	10	ﾘｾｯﾄ (任意)。目的対象ﾃﾞﾊﾞｲｽのﾘｾｯﾄに使用。或る筋書でのﾃﾞﾊﾞｯｸﾞを必要とし得る、目的対象ﾃﾞﾊﾞｲｽをﾘｾｯﾄ状態で保持することをPower Debuggerに許すために、このﾋﾟﾝの接続が推奨されます。
VTG	4	1	目的対象基準電圧。Power Debuggerは正しくﾚﾍﾞﾙ変換器を給電するために、このﾋﾟﾝで目的対象電圧を採取します。Power DebuggerはこのﾋﾟﾝからﾃﾞﾊﾞｯｸﾞWIRE動作で3mA未満、他の動作で1mA未満を引き出します。
GND	2,10	3,5,9	接地。Power Debuggerと目的対象ﾃﾞﾊﾞｲｽが同じ接地基準を共用するのを保証するために全てが接続されなければなりません。

5.3.4. aWire物理ｲﾝﾀｰﾌｪｰｽ

aWireｲﾝﾀｰﾌｪｰｽはﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞの機能を許すためにAVRﾃﾞﾊﾞｲｽのRESET線を利用します。Power Debuggerによって特別な許可手順が送出され、これはﾋﾟﾝの既定RESET機能を禁止します。

aWireｲﾝﾀｰﾌｧｰｽを持つAtmel AVRを含む応用PCBの設計時には図5-8.で示されるﾋﾟﾝ配列を使うことが推奨されます。特定のｷｯﾄで含まれるｹｰﾌﾞﾙとｱﾀﾞﾌﾟﾀに応じて、このﾋﾟﾝ配列の100milと50milの両変種が支援されます。

図5-8. aWireﾍｯﾀﾞ ﾋﾟﾝ配列



助言: aWireが半二重ｲﾝﾀｰﾌｪｰｽなので、方向変更時の誤った開始ﾋﾞｯﾄ検出を避けるため、約47kΩのRESETでのﾌﾟﾙｱｯﾌﾟ抵抗が推奨されます。

aWireｲﾝﾀｰﾌｪｰｽはﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞの両ｲﾝﾀｰﾌｪｰｽとして使うことができます。10ﾋﾟﾝJTAGｲﾝﾀｰﾌｪｰｽを通して利用可能なOCDｼｽﾃﾑの全ての機能もaWireを使ってｱｸｾｽすることができます。

5.3.5. aWire目的対象への接続

aWireｲﾝﾀｰﾌｪｰｽはVCCとGNDに加えて1つのﾃﾞｰﾀ線だけが必要です。例えﾃﾞﾊﾞｯｶﾞがﾃﾞｰﾀ線としてJTAG TDOを使っても、目的対象でこの線はnRESET線です。

6ﾋﾟﾝaWireｺﾈｸﾀ用の推奨ﾋﾟﾝ配列は図5-8.で示されます。

6ﾋﾟﾝ100mil aWireﾍｯﾀﾞへの接続

標準100mil aWireﾍｯﾀﾞへ接続するには(いくつかのｷｯﾄに含まれる)ﾌﾗｯﾄ ｹｰﾌﾞﾙ上の6ﾋﾟﾝ100mil引き出しを使ってください。

6ﾋﾟﾝ50mil aWireﾍｯﾀﾞへの接続

標準50mil aWireﾍｯﾀﾞへ接続するには(いくつかのｷｯﾄに含まれる)ｱﾀﾞﾌﾟﾀ基板を使ってください。

独自100milﾍｯﾀﾞへの接続

Power DebuggerのAVRｺﾈｸﾀ ﾎﾟｰﾄと目的対象基板を接続するには10ﾋﾟﾝのﾐﾆ ﾊﾞﾗ線ｹｰﾌﾞﾙが使われるべきです。右表で記述されるように3つの接続が必要とされます。

表5-8. Power DebuggerのaWireﾋﾟﾝ割り当て

Power Debugger AVRﾎﾟｰﾄ ﾋﾟﾝ	目的対象ﾋﾟﾝ	ﾐﾆ ﾊﾞﾗ線ﾋﾟﾝ	aWireﾋﾟﾝ
1 (TCK)		1	
2 (GND)	GND	2	6
3 (TDO)	DATA	3	1
4 (VTG)	VTG	4	2
5 (TMS)		5	
6 (nSRST)		6	
7 (未接続)		7	
8 (nTRST)		8	
9 (TDI)		9	
10 (GND)		0	

5.3.6. 特別な考慮

JTAGインターフェース

いくつかのAtmel AVR UC3デバイスでJTAGポートは既定で許可されません。これらのデバイス使用時、Power DebuggerがJTAGインターフェースを許可することをできるようにRESET線を正しくすることが重要です。

aWireインターフェース

aWire通信のボーレートはデータがそれら2つの領域間で同期されなければならないので、システム クロックの周波数に依存します。Power Debuggerはシステム クロックがより低いことを自動的に検知し、それに従ってボーレートを再校正します。自動校正は8kHzのシステム クロック周波数へ落とすように動くだけです。デバッグ作業中のより低いシステム クロックへの切り替えは目的対象との交信を失わせるかもしれません。

必要とされるなら、aWireのボーレートはaWireクロック項目を設定することによって制限することができます。自動検出は未だ動きますが、上限値は結果を強制されるでしょう。

RESETピンに接続されるどの安定化コンデンサも、それらがインターフェースの正しい動作を妨げるため、aWire使用時に切断されなければなりません。この線上の弱い(10kΩまたはより高い)外部プルアップが推奨されます。

遮断休止動作

いくつかのAVR UC3デバイスは1.8Vに調整された入出力線とで3.3V供給動作で使うことができる内部調整器を持ちます。これは内部調整器がコアと殆どの入出力の両方に給電することを意味します。Atmel AVR ONE!デバッガだけがこの調整器がOFFの休止動作を使っている間のデバッグを支援します。

5.3.7. EVTI/EVTOの使い方

EVTIとEVTOのピンはPower Debuggerでアクセスできません。けれども、それらは他の外部装置と共に未だ使うことができます。

EVTIは以下の目的で使うことができます。

- 目的対象は外部事象への応答で実行の停止を強制することができます。DCレジスタ内の制御での事象(EIC:Event In Control)ビットが'01'を書かれる場合、EVTIピンでのHigh⇒Low遷移が中断点(ブレーク ポイント)状態を生成します。これが起こる時にDS内の外部中断点(EXB:External Breakpoint)ビットが設定(1)されます。
- 追跡同期メッセージ生成。Power Debuggerによって使われません。

EVTOは以下の目的で使うことができます。

- CPUがデバッグ動作へ移行したことを指示。DC内のEOSビットの'01'設定は、目的対象デバイスがデバッグ動作移行時に1CPUクロック周期間Lowへ引かれることをEVTOピンに起こします。この信号は外部オシロスコープ用の起動元として使うことができます。
- CPUが中断点または監視点に到達したことを指示。対応する中断点/監視点制御レジスタ内のEOCビットの設定により、中断点または監視点の状態がEVTOピンで示されます。DC内のEOSビットはこの機能を許可するために'10'に設定されなければなりません。そして監視点タイミングを調べるためにEVTOピンは外部オシロスコープに接続することができます。
- 追跡タイミング信号生成。Power Debuggerによって使われません。

5.4. tinyAVR、megaAVR、XMEGAデバイス

AVRデバイスは様々なプログラミングとデバッグのインターフェースが特徴です。そのデバイスで支援されるインターフェースについてはデバイスのデータシートを調べてください。

- いくつかのtinyAVR®デバイスはTPIインターフェースを持ちます。TPIはデバイスをプログラミングすることだけに使うことができ、これらのデバイスは全てでチップ上デバッグ能力を持ちません。
- いくつかのtinyAVRデバイスといくつかのmegaAVRデバイスはtinyOCDとして知られるチップ上デバッグ システムに接続するデバッグWIREインターフェースを持ちます。デバッグWIREを持つ全てのデバイスは実装プログラミングのためにSPIインターフェースも持ちます。
- いくつかのmegaAVRデバイスはmegaOCDとして知られるチップ上デバッグ システムを持つ、プログラミングとデバッグ用のJTAGインターフェースを持ちます。JTAGを持つ全てのデバイスは実装プログラミングのための代替インターフェースとしてSPIインターフェースも特徴です。
- 全てのAVR XMEGAデバイスはプログラミングとデバッグのためのPDIインターフェースを持ちます。いくつかのAVR XMEGAデバイスは同様の機能を持つJTAGインターフェースも持ちます。
- 新しいtinyAVRデバイスはプログラミングとデバッグに使うUPDIインターフェースを持ちます。

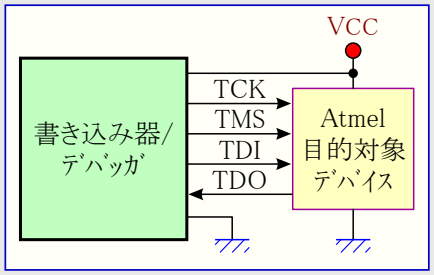
表5-9. ﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞ用インターフェース要約

ﾃﾞﾊﾞｲｽ区分	UPDI	TPI	SPI	ﾃﾞﾊﾞｯｸﾞWIRE	JTAG	PDI	aWire	SWD
tinyAVR	新ﾃﾞﾊﾞｲｽ	いくつかのﾃﾞﾊﾞｲｽ	いくつかのﾃﾞﾊﾞｲｽ	いくつかのﾃﾞﾊﾞｲｽ				
megaAVR			全ﾃﾞﾊﾞｲｽ	いくつかのﾃﾞﾊﾞｲｽ	いくつかのﾃﾞﾊﾞｲｽ			
AVR XMEGA					いくつかのﾃﾞﾊﾞｲｽ	全ﾃﾞﾊﾞｲｽ		
AVR UC					全ﾃﾞﾊﾞｲｽ		いくつかのﾃﾞﾊﾞｲｽ	
SAM					いくつかのﾃﾞﾊﾞｲｽ			全ﾃﾞﾊﾞｲｽ

5.4.1. JTAG物理インターフェース

JTAGインターフェースはIEEE® 1149.1規格に適合する4線検査入出力ﾎﾟｰﾄ(TAP:Test Access Port)制御器から成ります。IEEE規格は回路基板の接続性(境界走査)を効率的に検査する工業標準的な方法を提供するために開発されました。AtmelのAVRとSAMのﾃﾞﾊﾞｲｽは完全なﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾁｯﾌﾟ上ﾃﾞﾊﾞｯｸﾞの支援を含むように拡張されたこの機能を持ちます。

図5-9. JTAGインターフェースの基本



5.4.2. JTAG目的対象への接続

Power Debuggerは2つの50mil 10ﾋﾟﾝのJTAGｺﾈｸﾀが装備されます。両ｺﾈｸﾀは電気的に直接的に接続されていますが、AVR JTAGﾍｯﾀﾞとARM Cortexﾃﾞﾊﾞｯｸﾞ ﾍｯﾀﾞの2つの異なるﾋﾟﾝ配列を確認してください。ｺﾈｸﾀは目的対象MCU型式ではなく、目的対象基板のﾋﾟﾝ配列に基づいて選択されるべきです。例えばAVR STK600階層に装着されたSAMﾃﾞﾊﾞｲｽはAVRﾍｯﾀﾞが使われるべきです。

加えてPower DebuggerはそれのPCB上のAVRﾋﾟﾝ配列で未実装の100mil 10ﾋﾟﾝJTAGｺﾈｸﾀも持ちます。これは50mil AVRﾍｯﾀﾞに直接的に配線されます。

10ﾋﾟﾝAVR JTAGｺﾈｸﾀの推奨ﾋﾟﾝ配列は図5-6.で示されます。

10ﾋﾟﾝARM Cortexﾃﾞﾊﾞｯｸﾞ ｺﾈｸﾀの推奨ﾋﾟﾝ配列は図5-2.で示されます。

標準10ﾋﾟﾝ50milﾍｯﾀﾞへの直接接続

このﾍｯﾀﾞ形式を支援する基板へ直接接続するには(いくつかのｷｯﾄに含まれる)50mil 10ﾋﾟﾝ ﾌﾗｯﾄ ｹｰﾌﾞﾙを使ってください。AVRﾋﾟﾝ配列を持つﾍｯﾀﾞについてはPower DebuggerのAVRｺﾈｸﾀ ﾎﾟｰﾄを、ARM Cortexﾃﾞﾊﾞｯｸﾞ ﾍｯﾀﾞに従うﾍｯﾀﾞについてはSAMｺﾈｸﾀ ﾎﾟｰﾄを使ってください。

10ﾋﾟﾝの両ｺﾈｸﾀのﾋﾟﾝ配列は以降で示されます。

標準10ﾋﾟﾝ100milﾍｯﾀﾞへの接続

100milﾍｯﾀﾞに接続するには標準50mil⇒100milｱﾀﾞﾌﾟﾀを使ってください。(いくつかのｷｯﾄに含まれる)ｱﾀﾞﾌﾟﾀ基板がこの目的に使うことができ、または代わりにAVR目的対象に対してJTAGICE3のｱﾀﾞﾌﾟﾀを使うことができます。

重要: ｱﾀﾞﾌﾟﾀの2と10番ﾋﾟﾝ(AVR GND)が接続されるため、JTAGICE3の100milｱﾀﾞﾌﾟﾀはSAMｺﾈｸﾀ ﾎﾟｰﾄで使うことはできません。

独自100milﾍｯﾀﾞへの接続

目的対象基板が50milか100milの適合10ﾋﾟﾝJTAGﾍｯﾀﾞを持たない場合、10個の個別100milｿｹｯﾄへの入出力を与える(いくつかのｷｯﾄに含まれる)10ﾋﾟﾝの"ﾐﾆ ﾊﾞﾗ線"ｹｰﾌﾞﾙを使って独自ﾋﾟﾝ配列に割り当てることができます。

20ﾋﾟﾝ100milﾍｯﾀﾞへの接続

20ﾋﾟﾝ100milﾍｯﾀﾞを持つ目的対象へ接続するには(いくつかのｷｯﾄに含まれる)ｱﾀﾞﾌﾟﾀ基板を使ってください。

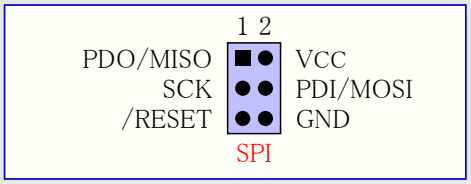
表5-10. Power Debugger JTAGﾋﾟﾝ説明

名前	ﾎﾟｰﾄのﾋﾟﾝ番号		説明
	AVR	SAM	
TCK	1	4	検査ｸﾛｯｸ (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへのｸﾛｯｸ信号)
TMS	5	2	検査種別選択 (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへの制御信号)
TDI	9	8	検査ﾃﾞｰﾀ入力 (Power Debuggerから目的対象ﾃﾞﾊﾞｲｽへ送出されるﾃﾞｰﾀ)
TDO	3	6	検査ﾃﾞｰﾀ出力 (目的対象ﾃﾞﾊﾞｲｽからPower Debuggerへ送出されるﾃﾞｰﾀ)
nTRST	8	-	検査ﾘｾｯﾄ(任意、いくつかのAVRﾃﾞﾊﾞｲｽのみ)。JTAG TAP制御器のﾘｾｯﾄに使用
nSRST	6	10	ﾘｾｯﾄ (任意)。目的対象ﾃﾞﾊﾞｲｽのﾘｾｯﾄに使用。或る筋書でのﾃﾞﾊﾞｯｸﾞを必要とし得る、目的対象ﾃﾞﾊﾞｲｽをﾘｾｯﾄ状態で保持することをPower Debuggerに許すために、このﾋﾟﾝの接続が推奨されます。
VTG	4	1	目的対象基準電圧。Power Debuggerは正しくﾚﾍﾞﾙ変換器を給電するために、このﾋﾟﾝで目的対象電圧を採取します。Power DebuggerはこのﾋﾟﾝからﾃﾞﾊﾞｯｸﾞWIRE動作で3mA未満、他の動作で1mA未満を引き出します。
GND	2,10	3,5,9	接地。Power Debuggerと目的対象ﾃﾞﾊﾞｲｽが同じ接地基準を共用するのを保証するために全てが接続されなければなりません。

5.4.3. SPI物理ｲﾝﾀｰﾌｪｰｽ

実装書き込み(ISP:In-System Programming)はﾌﾗｯｼｭ ﾒﾓﾘとEEPROM内にｺｰﾄﾞを書き込むのに目的対象のAtmel AVRの内部SPI(Serial Peripheral Interface)を使います。これはﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽではありません。SPIｲﾝﾀｰﾌｪｰｽを持つAVRを含む応用PCBの設計時、右図で示されるﾋﾟﾝ配列が使われるべきです。

図5-10. SPIﾍｯﾀﾞ ﾋﾟﾝ配列



5.4.4. SPI目的対象への接続

6ﾋﾟﾝSPIｺﾈｸﾀ用の推奨ﾋﾟﾝ配列は図5-10.で示されます。

6ﾋﾟﾝ100mil SPIﾍｯﾀﾞへの接続

標準100mil SPIﾍｯﾀﾞへ接続するには(いくつかのｷｯﾄに含まれる)ﾌﾗｯﾄ ｹｰﾌﾞﾙ上の6ﾋﾟﾝ100mil引き出しを使ってください。

6ﾋﾟﾝ50mil SPIﾍｯﾀﾞへの接続

標準50mil SPIﾍｯﾀﾞへ接続するには(いくつかのｷｯﾄに含まれる)ｱﾀﾞﾌﾟﾀ基板を使ってください。

独自100milﾍｯﾀﾞへの接続

Power DebuggerのAVRｺﾈｸﾀ ﾎﾟｰﾄと目的対象基板を接続するには10ﾋﾟﾝのﾐﾆ ﾊﾞﾗ線ｹｰﾌﾞﾙが使われるべきです。下表で記述されるように6つの接続が必要とされます。

重要: ﾃﾞﾊﾞｯｸﾞWIRE許可(DWEN)ﾋｭｰｽﾞがﾌﾟﾛｸﾞﾗﾑ(0)されている時に、例えSPIENﾋｭｰｽﾞもﾌﾟﾛｸﾞﾗﾑ(0)されていても、SPIｲﾝﾀｰﾌｪｰｽは事実上禁止されます。SPIｲﾝﾀｰﾌｪｰｽを再許可するには、ﾃﾞﾊﾞｯｸﾞWIREでのﾃﾞﾊﾞｯｸﾞ作業中に'disable debugWIRE'命令が発行されなければなりません。この方法でのﾃﾞﾊﾞｯｸﾞWIRE禁止はSPIENﾋｭｰｽﾞが既にﾌﾟﾛｸﾞﾗﾑ(0)されていることが必要です。Atmel StudioがﾃﾞﾊﾞｯｸﾞWIRE禁止を失敗する場合、多分SPIENﾋｭｰｽﾞがﾌﾟﾛｸﾞﾗﾑ(0)にされていないからです。その場合、SPIENﾋｭｰｽﾞをﾌﾟﾛｸﾞﾗﾑ(0)するのに高電圧ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ｲﾝﾀｰﾌｪｰｽを使うことが必要です。

情報: SPIｲﾝﾀｰﾌｪｰｽはそれがAtmel AVR製品の最初の実装書き込み(In System Programming)ｲﾝﾀｰﾌｪｰｽだったため、度々"ISP"として参照されます。今や他のｲﾝﾀｰﾌｪｰｽが実装書き込みに利用可能です。

表5-11. Power DebuggerのSPIﾋﾟﾝ割り当て

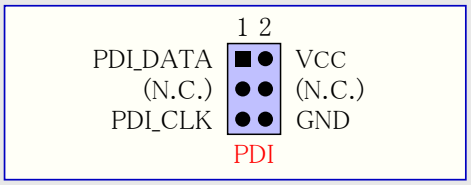
Power Debugger AVRﾎﾟｰﾄ ﾋﾟﾝ	目的対象ﾋﾟﾝ	ﾐﾆ ﾊﾞﾗ線ﾋﾟﾝ	SPIﾋﾟﾝ
1 (TCK)	SCK	1	3
2 (GND)	GND	2	6
3 (TDO)	MISO	3	1
4 (VTG)	VTG	4	2
5 (TMS)		5	
6 (nSRST)	/RESET	6	5
7 (未接続)		7	
8 (nTRST)		8	
9 (TDI)	MOSI	9	4
10 (GND)		0	

5.4.5. PDI

ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ/ﾃﾞﾊﾞｯｸﾞ用ｲﾝﾀｰﾌｪｰｽ(PDI:Program and Debug Interface)はﾃﾞﾊﾞｲｽの外部ﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾁｯﾌﾟ上ﾃﾞﾊﾞｯｸﾞ用のAtmel専有ｲﾝﾀｰﾌｪｰｽです。PDI物性は目的対象ﾃﾞﾊﾞｲｽとの双方向半二重同期通信を提供する2ﾋﾟﾝ ｲﾝﾀｰﾌｪｰｽです。

PDIｲﾝﾀｰﾌｪｰｽを持つAtmel AVRを含む応用PCBの設計時、右図で示されるﾋﾟﾝ配列が使われるべきです。そしてPower Debugger探針を応用PCBへ接続するのに、Power Debuggerｷｯﾄと共に提供される6ﾋﾟﾝ ｱﾀﾞﾌﾟﾀの1つを使うことができます。

図5-11. PDIﾍｯﾀﾞ ﾋﾟﾝ配列



5.4.6. PDI目的対象への接続

6ﾋﾟﾝPDIｺﾈｸﾀ用の推奨ﾋﾟﾝ配列は図5-11.で示されます。

6ﾋﾟﾝ100mil PDIﾍｯﾀﾞへの接続

標準100mil PDIﾍｯﾀﾞへ接続するには(いくつかのｷｯﾄに含まれる)ﾌﾗｯﾄ ｹｰﾌﾞﾙ上の6ﾋﾟﾝ100mil引き出しを使ってください。

6ﾋﾟﾝ50mil PDIﾍｯﾀﾞへの接続

標準50mil PDIﾍｯﾀﾞへ接続するには(いくつかのｷｯﾄに含まれる)ｱﾀﾞﾌﾟﾀ基板を使ってください。

独自100milﾍｯﾀﾞへの接続

Power DebuggerのAVRｺﾈｸﾀ ﾎﾟｰﾄと目的対象基板を接続するには10ﾋﾟﾝのﾐﾆ ﾊﾞﾗ線ｹｰﾌﾞﾙが使われるべきです。下表で記述されるように4つの接続が必要とされます。

重要: 必要とされるﾋﾟﾝ配列はPDI_DATAが9番ﾋﾟﾝに接続されるJTAGICE mkⅡのJTAG探針とは違います。Power DebuggerはAtmel-ICE、JTAGICE3、AVR ONE!、AVR Dragon™製品によって使われるﾋﾟﾝ配列と互換性があります。

表5-12. Poweer DebuggerのPDIﾋﾟﾝ割り当て

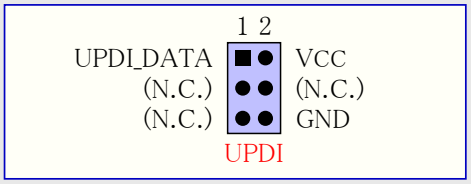
Power Debugger AVRﾎﾟｰﾄ ﾋﾟﾝ	目的対象ﾋﾟﾝ	ﾐﾆ ﾊﾞﾗ線ﾋﾟﾝ	STK600 PDIﾋﾟﾝ
1 (TCK)		1	
2 (GND)	GND	2	6
3 (TDO)	PDI_DATA	3	1
4 (VTG)	VTG	4	2
5 (TMS)		5	
6 (nSRST)	PDI_CLK	6	5
7 (未接続)		7	
8 (nTRST)		8	
9 (TDI)		9	
10 (GND)		0	

5.4.7. UPDI物理ｲﾝﾀｰﾌｪｰｽ

統一ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ/ﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽ(UPDI:Unified Program and Debug Interface)はﾃﾞﾊﾞｲｽの外部ﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾁｯﾌﾟ上ﾃﾞﾊﾞｯｸﾞ用のAtmel専有ｲﾝﾀｰﾌｪｰｽです。これは全てのAVR XMEGAﾃﾞﾊﾞｲｽで見つかるPDI 2線物理ｲﾝﾀｰﾌｪｰｽの後継です。UPDIはﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞの目的のために目的対象ﾃﾞﾊﾞｲｽと双方向半二重非同期通信を提供する単線ｲﾝﾀｰﾌｪｰｽです。

UPDIｲﾝﾀｰﾌｪｰｽを持つAtmel AVRを含む応用PCBの設計時、右で示されるﾋﾟﾝ配列が使われるべきです。そしてPower Debugger探針を応用PCBへ接続するのに、Power Debuggerｷｯﾄと共に提供される6ﾋﾟﾝ ｱﾀﾞﾌﾟﾀの1つを使うことができます。

図5-12. UPDIﾍｯﾀﾞ ﾋﾟﾝ配列



5.4.7.1. UPDIと/RESET

UPDI単線ｲﾝﾀｰﾌｪｰｽは目的対象AVRﾃﾞﾊﾞｲｽに応じて専用ﾋﾟﾝまたは共用ﾋﾟﾝになり得ます。更なる情報についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄを調べてください。

UPDIｲﾝﾀｰﾌｪｰｽが共用ﾋﾟﾝの場合、そのﾋﾟﾝはRSTPINCFG1,0ﾋｭｰｽﾞの設定によってUPDI、/RESET、汎用入出力(GPIO)のどれかに構成設定することができます。

RSTPINCFG1,0ﾋｭｰｽﾞはﾃﾞｰﾀｼｰﾄで記述されるように以下の構成設定を持ちます。各選択の実際の実装はここで与えられます。

表5-13. RSTPINCFG1,0ﾋｭｰｽﾞ構成設定

RSTPINCFG1,0	構成設定	使用法
0 0	GPIO	汎用入出力(GPIO)ﾋﾟﾝ。UPDIにｱｸｾｽするためにはこのﾋﾟﾝに12Vﾊﾟﾙｽが印加されなければなりません。外部ﾘｾｯﾄ元は利用不可です。
0 1	UPDI	ﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞの専用ﾋﾟﾝ。外部ﾘｾｯﾄ元は利用不可です。
1 0	RESET	ﾘｾｯﾄ信号入力。UPDIにｱｸｾｽするためにはこのﾋﾟﾝに12Vﾊﾟﾙｽが印加されなければなりません。
1 1	(予約)	利用不可

注: 古いAVRﾃﾞﾊﾞｲｽは(直列と並列の両変種が存在する)"高電圧ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ"として知られるﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ｲﾝﾀｰﾌｪｰｽを持ちます。一般的にこのｲﾝﾀｰﾌｪｰｽはﾌﾟﾛｸﾞﾗﾐﾝｸﾞ作業の間中/RESETﾋﾟﾝに印加される12Vを必要とします。UPDIｲﾝﾀｰﾌｪｰｽは完全に異なるｲﾝﾀｰﾌｪｰｽです。UPDIﾋﾟﾝは主にﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞのﾋﾟﾝで、これは(/RESETまたはGPIOの)代替機能を持つようにﾋｭｰｽﾞ切り替えすることができます。代替機能が選ばれる場合、UPDI機能を再活性するためにはそのﾋﾟﾝで12Vﾊﾟﾙｽが必要とされます。

注: ﾋﾟﾝの制限のために設計がUPDI信号の共有を必要とする場合、ﾃﾞﾊﾞｲｽをﾌﾟﾛｸﾞﾗﾐﾝｸﾞすることができることを保証するために手順が取られなければなりません。UPDI信号が正しく機能するだけでなく、外部部品に対する12Vﾊﾟﾙｽからの損傷を避けるため、ﾃﾞﾊﾞｲｽのﾌﾟﾛｸﾞﾗﾐﾝｸﾞやﾃﾞﾊﾞｯｸﾞを試みる時にこのﾋﾟﾝ上のどの部品も切断することが推奨されます。これは既定によって実装され、ﾃﾞﾊﾞｯｸﾞの間に取り外されるか、またはﾋﾟﾝ ﾍｯﾀﾞによって置き換えられる0Ω抵抗器を用いて行うことができます。この構成設定は事実上、ﾃﾞﾊﾞｲｽを実装する前にﾌﾟﾛｸﾞﾗﾐﾝｸﾞが行われるべきことを意味します。

5.4.8. UPDI目的対象への接続

6ﾋﾟﾝ UPDIｺﾈｸﾀ用の推奨ﾋﾟﾝ配列は図5-12.で示されます。

6ﾋﾟﾝ100mil UPDIﾍｯﾀﾞへの接続

標準100mil UPDIﾍｯﾀﾞへ接続するには(いくつかのｷｯﾄに含まれる)ﾌﾗｯﾄ ｹｰﾌﾞﾙ上の6ﾋﾟﾝ100mil引き出しを使ってください。

6ﾋﾟﾝ50mil UPDIﾍｯﾀﾞへの接続

標準50mil UPDIﾍｯﾀﾞへ接続するには(いくつかのｷｯﾄに含まれる)ｱﾀﾞﾌﾟﾀ基板を使ってください。

独自100milﾍｯﾀﾞへの接続

Power DebuggerのAVRｺﾈｸﾀ ﾎﾟｰﾄと目的対象基板を接続するには10ﾋﾟﾝのﾐﾆ ﾊﾞﾗ線ｹｰﾌﾞﾙが使われるべきです。右表で記述されるように3つの接続が必要とされます。

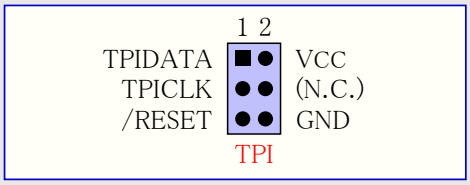
表5-14. Power DebuggerのUPDIﾋﾟﾝ割り当て

Power Debugger AVRﾎﾟｰﾄ ﾋﾟﾝ	目的対象ﾋﾟﾝ	ﾐﾆ ﾊﾞﾗ線ﾋﾟﾝ	STK600 UPDIﾋﾟﾝ
1 (TCK)		1	
2 (GND)	GND	2	6
3 (TDO)	UPDI_DATA	3	1
4 (VTG)	VTG	4	2
5 (TMS)		5	
6 (nSRST)	(/RESET検知)	6	5
7 (未接続)		7	
8 (nTRST)		8	
9 (TDI)		9	
10 (GND)		0	

5.4.9. TPI物理ｲﾝﾀｰﾌｪｰｽ

TPIはいくつかのAVR ATtinyﾃﾞﾊﾞｲｽ用のﾌﾟﾛｸﾞﾗﾐﾝｸﾞ専用ｲﾝﾀｰﾌｪｰｽです。これはﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽではなく、これらのﾃﾞﾊﾞｲｽはOCD能力を持ちません。TPIｲﾝﾀｰﾌｪｰｽを持つAVRを含む応用PCBの設計時、右図で示されるﾋﾟﾝ配列が使われるべきです。

図5-13. TPIﾍｯﾀﾞ ﾋﾟﾝ配列



5.4.10. TPI目的対象への接続

6ﾋﾟﾝTPIｺﾈｸﾀ用の推奨ﾋﾟﾝ配列は図5-13.で示されます。

6ﾋﾟﾝ100mil TPIﾍｯﾀﾞへの接続

標準100mil TPIﾍｯﾀﾞへ接続するには(いくつかのｷｯﾄに含まれる)ﾌﾗｯﾄ ｹｰﾌﾞﾙ上の6ﾋﾟﾝ100mil引き出しを使ってください。

6ﾋﾟﾝ50mil TPIﾍｯﾀﾞへの接続

標準50mil TPIﾍｯﾀﾞへ接続するには(いくつかのｷｯﾄに含まれる)ｱﾀﾞﾌﾟﾀ基板を使ってください。

独自100milﾍｯﾀﾞへの接続

Power DebuggerのAVRｺﾈｸﾀ ﾎﾟｰﾄと目的対象基板を接続するには10ﾋﾟﾝのﾐﾆ ﾊﾞﾗ線ｹｰﾌﾞﾙが使われるべきです。右表で記述されるように5つの接続が必要とされます。

表5-15. Power DebuggerのTPIﾋﾟﾝ割り当て

Power Debugger AVRﾎﾟｰﾄ ﾋﾟﾝ	目的対象ﾋﾟﾝ	ﾐﾆ ﾊﾞﾗ線ﾋﾟﾝ	TPIﾋﾟﾝ
1 (TCK)	CLOCK	1	3
2 (GND)	GND	2	6
3 (TDO)	DATA	3	1
4 (VTG)	VTG	4	2
5 (TMS)		5	
6 (nSRST)	/RESET	6	5
7 (未接続)		7	
8 (nTRST)		8	
9 (TDI)		9	
10 (GND)		0	

5.4.11. 高度なﾃﾞﾊﾞｯｸﾞ (AVR JTAG/ﾃﾞﾊﾞｯｸﾞWIREﾃﾞﾊﾞｲｽ)

入出力周辺機能

殆どの周辺機能は例えﾌﾟﾛｸﾞﾗﾑ実行が中断点(ﾌﾞﾚｰｸ ﾎﾟｲﾝﾄ)によって停止されても、走行を続けます。例: UART送信中に中断点に到達した場合、その送信は完了されて対応するﾋﾞｯﾄが設定されます。送信完了(TXC)ﾌﾗｸﾞが設定(1)され、例え実際のﾃﾞﾊﾞｲｽでは通常、より遅くに起こるとしても、ｺｰﾄﾞの次の単一段実行で利用可能になります。

全ての入出力単位部は次の2つの例外付きで停止動作で走行を継続します。

- ﾀｲﾏ/ｶｳﾝﾀ (ｿﾌﾄｳｪｱ前処理部を使って構成設定可能)
- ｳｫｯﾁﾄﾞｯｸﾞ ﾀｲﾏ (ﾃﾞﾊﾞｯｸﾞ中のﾘｾｯﾄを防ぐため、常に停止)

単一段実行の入出力ｱｸｾｽ

入出力が停止動作で走行を続けるため、或るﾀｲﾐﾝｸﾞの問題を避けるために注意が払われるべきです。例えば次のｺｰﾄﾞです。

```
OUT     PORTB, $AA
IN      TEMP, PINB
```

このｺｰﾄﾞの通常走行時、ﾃﾞｰﾀはIN操作によって採取される時に物理的に未だﾋﾟﾝにﾗｯﾁされていないため、TEMPﾚｼﾞｽﾀは$AAを読み戻しません。PINﾚｼﾞｽﾀに正しい値が存在するのを保証するには、OUTとINの命令間にNOP命令が置かれなければなりません。

けれども、OCDを通してこの機能を単一段実行すると、例え単一段実行中にｺｱが停止されていても、入出力が全速で走行するので、このｺｰﾄﾞは常に$AAを与えます。

単一段実行とﾀｲﾐﾝｸﾞ

或るﾚｼﾞｽﾀは制御信号許可後に与えられた周期数内に読みまたは書きが必要です。停止動作でI/Oｸﾛｯｸと周辺機能が全速度走行を続けるため、そのようなｺｰﾄﾞを通した単一段実行はﾀｲﾐﾝｸﾞ必要条件に合致しないでしょう。2つの単一段実行間で、I/Oｸﾛｯｸは100万周期走行するかもしれません。このようなﾀｲﾐﾝｸﾞ必要条件で成功裏のﾚｼﾞｽﾀを読みまたは書きを行うには、全速でﾃﾞﾊﾞｲｽを走らせる非分断操作として読みまたは書きの全体手順が実行されるべきです。これはｺｰﾄﾞを実行するのにﾏｸﾛまたは関数を使う、またはﾃﾞﾊﾞｯｸﾞ環境でｶｰｿﾙまで実行の機能を使うことによって行うことができます。

16ﾋﾞｯﾄ ﾚｼﾞｽﾀのｱｸｾｽ

Atmel AVR周辺機能は代表的に8ﾋﾞｯﾄ ﾃﾞｰﾀ ﾊﾞｽ経由でｱｸｾｽすることができる多数の16ﾋﾞｯﾄ ﾚｼﾞｽﾀ(例えば、16ﾋﾞｯﾄ ﾀｲﾏ/ｶｳﾝﾀのTCNTn)を含みます。16ﾋﾞｯﾄ ﾚｼﾞｽﾀは2つの読みまたは書きの操作を用いてﾊﾞｲﾄ ｱｸｾｽされなければなりません。16ﾋﾞｯﾄ ｱｸｾｽの中間での中断、またはこの状況を通しての単一段実行は誤った値に帰着するかもしれません。

制限されたI/Oﾚｼﾞｽﾀ ｱｸｾｽ

或るﾚｼﾞｽﾀはそれらの内容に影響を及ぼさずに読むことができません。このようなﾚｼﾞｽﾀは読むことによって解除(0)されるﾌﾗｸﾞを含むそれらや、緩衝されたﾃﾞｰﾀ ﾚｼﾞｽﾀ(例えば、UDR)を含みます。OCDﾃﾞﾊﾞｯｸﾞの意図された非侵入的性質を守るため、ｿﾌﾄｳｪｱ前処理部は停止動作時にこれらのﾚｼﾞｽﾀの読み込みを防ぎます。加えて、いくつかのﾚｼﾞｽﾀは副作用を起こすことなく、安全に書くことができません。それらのﾚｼﾞｽﾀは読み込み専用です。例えば、以下です。

- 何れかのﾋﾞｯﾄへの'1'書き込みによって解除(0)されるﾌﾗｸﾞのﾌﾗｸﾞ ﾚｼﾞｽﾀ。これらのﾚｼﾞｽﾀは読み込み専用です。
- UDRとSPDRは単位部の状態に影響を及ぼすことなく読むことができません。これらのﾚｼﾞｽﾀはｱｸｾｽ不能です。

5.4.12. megaAVR特別な考慮

ソフトウェア中断点

ATmega128[A]は初期版のOCD単位部を含むため、ソフトウェア中断点(ブレーク ポイント)用BREAK命令の使用を支援しません。

JTAGクロック

目的対象クロック周波数はデバッグ作業を始める前にソフトウェア前処理部で正確に指定されなければなりません。同期化の理由のため、JTAGのTCK信号は信頼できるデバッグのために目的対象クロック周波数の1/4未満でなければなりません。JTAGインターフェース経由のプログラミング時、TCK周波数は目的対象デバイスの最大周波数評価によって制限され、使われる実際のクロック周波数ではありません。

内部RC発振器使用時、周波数がデバイス毎に変わり得て、温度やVCCの変化によって影響を及ぼされることに注意してください。目的対象クロック周波数を指定する時は控え目に(確実性を高く)してください。

JTAGENとOCDENのヒューズ

JTAGインターフェースは既定によってプログラム(0)されているJTAGENヒューズによって許可されます。これはJTAGプログラミング インターフェースへのアクセスを許します。この機構を通して、OCDENヒューズをプログラム(0)することができます(既定でのOCDENは非プログラム(1))。これはデバイスのデバッグを容易にするためにOCDへのアクセスを許します。ソフトウェア前処理部は作業終了時にOCDENヒューズが非プログラム(1)にさせられることを常に保証し、それによってOCD単位部による不必要な電力消費を制限します。JTAGENヒューズが意図せずに禁止された場合、SPIまたは高電圧プログラミング法を使ってだけ再許可することができます。

JTAGENヒューズがプログラム(0)されたなら、JTAGインターフェースはJTDビットを設定(1)することによって未だファームウェアで禁止することができます。これはデバッグ不可コードにし、デバッグ作業を試みる時に実行されべきではありません。デバッグ作業開始時にこのようなコードが既に実行された場合、Power Debuggerは接続中にRESET線を有効にします。この線が正しく配線されているなら、目的対象AVRデバイスをリセットに強制し、それによってJTAG接続を許します。

JTAGインターフェースが許可された場合、JTAGピンは代替ピン機能に使うことができません。それらはプログラム コードからJTDビットの設定(1)、またはプログラミング インターフェースを通してJTAGENヒューズの解除(1)のどちらかによってJTAGインターフェースが禁止されるまでJTAG専用ピンに留まります。

助言: JTDビットの設定(1)によってJTAGインターフェースを禁止するコードを走行するデバイスでRESET線を有効にしてJTAGインターフェースを再許可することをPower Debuggerに許すために、プログラミング ダイアログとデバッグ任意選択ダイアログの両方で"use external reset"チェック枠のチェックを確実にしてください。

IDR/OCDR事象

IDR(In-out Data Register)はOCDR(On Chip Debug Register)としても知られ、デバッグ作業中の停止動作の時にMCUに対して情報を読み書きするため、デバッガによって広範囲に使われます。デバッグされつつあるAVRデバイスのOCDRレジスタに応用プログラムがバイトデータを書く時に、Power Debuggerはこの値を読み出してそれをソフトウェア前処理部のメッセージ ウィンドウに表示します。OCDRレジスタは50ms毎にポーリングされ、故により高い繰り返しでの書き込みは信頼に足る結果を生じません。デバッグ中にAVRデバイスが電力を失うと、偽のOCDR事象が報告され得ます。これは目的対象電圧がAVRの最低動作電圧以下に落ちる時にPower Debuggerが未だデバイスをポーリングし得るために起こります。

5.4.13. AVR XMEGA特別な考慮

OCDとクロック駆動

MCUが停止動作へ移行すると、MCUクロックとしてOCDクロックが使われます。OCDクロックはJTAGインターフェースが使われている場合にJTAGのTCK、またはPDIインターフェースが使われている場合にPDI_CLKのどちらかです。

停止動作でのI/O単位部

初期のAtmel megaAVRデバイスと対照的に、XMEGAの入出力単位部は停止動作で停止されます。これはUSART送信が中断され、計時器が停止されることを意味します。

ハードウェア中断点

2つのアドレス比較器と2つの値比較器で4つのハードウェア中断点(ブレーク ポイント)比較器があります。これらは或る制限を持ちます。

- 全ての中断点は同じ形式(プログラムまたはデータ)でなければなりません。
- 全てのデータ中断点は同じメモリ領域(I/O、SRAM、またはXRAM)でなければなりません。
- アドレス範囲が使われる場合は1つの中断点だけしかできません。

以下は設定し得る各種組み合わせです。

- 2つの単一データまたはプログラム アドレスの中断点
- 1つのデータまたはプログラム アドレスの範囲中断点
- 単一値比較を持つ2つの単一データ アドレス中断点
- アドレス範囲、値範囲、または両方を持つ1つのデータ中断点

Atmel Studioは中断点を設定できない場合に何故かを告げます。ソフトウェア中断点が利用可能な場合、データ中断点がプログラム中断点より上の優先権を持ちます。

外部リセットとPDI物性

PDI物理インターフェースはクロックとしてリセット線を使います。デバッグ時、リセットのプルアップは10kΩに、より大きく、または取り外されるべきです。どのリセット コンデンサも取り去られるべきです。他の外部リセット元は切断されるべきです。

ATxmegaA1改訂Hとそれ以前版に対するデバッグ

ATxmegaA1系の初期版でデバイスが或る休止動作中に許可されつつあることからOCDが妨げられるバグが存在しました。OCDを再許可するには2つの対処があります。

- Tools(ツール)メニューでPower Debugger Options(Power Debugger任意選択)へ行き、”Always activate external reset when reprogramming device(デバイス再プログラミング時に常に外部リセットを活性)”を許可してください。
- チップ消去を実行してください。

このバグを起動する休止動作は以下です。

- パワーダウン
- パワーセーブ
- スタンバイ
- 拡張スタンバイ

5.4.14. デバッグWIRE特別な考慮

デバッグWIRE(dW)ピンは物理的に外部リセット(RESET)と同じピンに配置されます。従ってデバッグWIREインターフェースが許可される時に外部リセット元は支援されません。

デバッグWIREインターフェースが機能するためには、目的対象デバイスでデバッグWIRE許可(DWEN)ヒューズが設定(0)されなければなりません。このヒューズはAtmel AVRデバイスが工場から出荷される時に既定によって非プログラム(1)にされます。デバッグWIREインターフェースそれ自身はこのヒューズを設定することができません。DWENヒューズを設定するにはSPI動作が使われなければなりません。ソフトウェア前処理部は必要なSPIピンが接続されていればこれを自動的に処理します。Atmel Studioのプログラミング ダイアログからSPIプログラミングを使うこともできます。以下の2つのどちらかを行います。

- デバッグWIRE部でデバッグ作業の開始を試みてください。デバッグWIREインターフェースが許可されていなければ、Atmel Studioは再試行を提供するか、またはSPIプログラミングを使ってデバッグWIREを許可しようと試みます。完全なSPIヘッダ接続があれば、デバッグWIREが許可され、目的対象で電源のOFF/ONを問われるでしょう。これは効果を得るべくヒューズを変更するのに必要とされます。
- SPI動作でプログラミング ダイアログを開き、識票が正しいデバイスと一致することを確認してください。デバッグWIREを許可するためにDWENヒューズをチェックしてください。

重要: SPIENヒューズがプログラム(0)され、RSTDISBLヒューズが非プログラム(1)のままにされていることが重要です。これを行わないことはデバッグWIRE動作でデバイスを動作停止にし、DWEN設定を元に戻すのに高電圧プログラミングが必要とされます。

デバッグWIREインターフェースを禁止するには、DWENヒューズを非プログラム(1)にするのに高電圧プログラムを使ってください。代わりに、デバッグWIREを一時的に禁止するのにデバッグWIREそれ自身を使い、SPIENヒューズが設定(0)されていれば、SPIプログラミング実行を許します。

重要: SPIENヒューズがプログラム(0)のままにされていない場合、Atmel Studioはこの操作を緩衝することができず、高電圧プログラミングが使われなければなりません。

デバッグ作業の間、’Debug(デバッグ)’メニューから’Disable debugWIRE and Close(デバッグWIREを禁止して閉じる)’メニュー任意選択を選んでください。デバッグWIREが一時的に禁止され、Atmel StudioはDWENヒューズを非プログラム(1)にするのにSPIプログラミングを使います。

プログラム(0)されたDWENヒューズがあることは全ての休止動作で走行することをクロック系のいくつかの部分に許します。これは休止動作間のAVRの消費電力を増やします。従ってデバッグWIREが使われない時は常にDWENヒューズが禁止されるべきです。

デバッグWIREが使われる目的対象応用PCBの設計時、正しい動作のために以下の考慮をしなければなりません。

- dW(RESET)線のプルアップ抵抗は10kΩよりも小さく(強力で)あってはなりません。プルアップ抵抗はデバッガ ツールがこれを提供するため、機能的にデバッグWIREに必要ではありません。
- RESETピンに接続されるどの安定化コンデンサも、それらがインターフェースの正しい動作を妨げるため、デバッグWIRE使用時に切断されなければなりません。
- 全ての外部リセット元またはREEST線上の他の活性な駆動部は、それらがインターフェースの正しい動作を妨げるため、切断されなければなりません。

目的対象デバイスの施錠ビットを決してプログラム(0)してはなりません。デバッグWIREインターフェースは正しく機能するために施錠ビットの解除(1)を必要とします。

5.4.15. ﾃﾞﾊﾞｯｸﾞWIREｿﾌﾄｳｪｱ中断点

ﾃﾞﾊﾞｯｸﾞWIRE OCDはAtmel megaAVR (JTAG) OCDと比べた時に劇的に縮小されています。これはﾃﾞﾊﾞｯｸﾞ目的のために使用者に対して利用可能などのﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ中断点比較器も持たないことを意味します。このような1つの比較器はｶｰｿﾙまで実行と単一段階実行の操作の目的のために存在しますが、追加の使用者中断点(ﾌﾞﾚｰｸ ﾎﾟｲﾝﾄ)はﾊｰﾄﾞｳｪｱで支援されません。

代わりに、ﾃﾞﾊﾞｯｶﾞはAVRのBREAK命令を利用しなければなりません。この命令はﾌﾗｯｼｭ ﾒﾓﾘに置くことができ、実行のためにそれが取得された時にAVR CPUを停止動作へ移行させます。ﾃﾞﾊﾞｯｸﾞ中に中断点を支援するには、使用者が求める中断点の位置でﾃﾞﾊﾞｯｶﾞがﾌﾗｯｼｭ ﾒﾓﾘ内にBREAK命令を挿入しなければなりません。元の命令は後で置き換えるために保存されなければなりません。BREAK命令上で単一段階実行をすると、ﾃﾞﾊﾞｯｶﾞはﾌﾟﾛｸﾞﾗﾑの動きを守るために保存された元の命令を実行しなければなりません。極端な場合、BREAKはﾌﾗｯｼｭ ﾒﾓﾘから取り去られて後で置き換えられなければなりません。これら全ての筋書きは中断点からの単一段階実行時に明白な遅延を引き起こし得て、これは目的対象ｸﾛｯｸ周波数が非常に低い時に激化されます。

従って、可能であれば、以下の指針に従うことが推奨されます。

- ﾃﾞﾊﾞｯｸﾞ中に常に可能な限り高い周波数で目的対象を走らせてください。ﾃﾞﾊﾞｯｸﾞWIRE物理ｲﾝﾀｰﾌｪｰｽは目的対象ｸﾛｯｸでｸﾛｯｸ駆動されます。
- 1つ毎に目的対象で置き換えるべきﾌﾗｯｼｭ ﾍﾟｰｼﾞを必要とするため、中断点の追加と削除の数を最小にしてみてください。
- ﾌﾗｯｼｭ ﾍﾟｰｼﾞの書き込み操作を最小とするため、同時に少数の中断点を追加または削除するようにしてください。
- 可能なら、2語命令に中断点を置くことを避けてください。

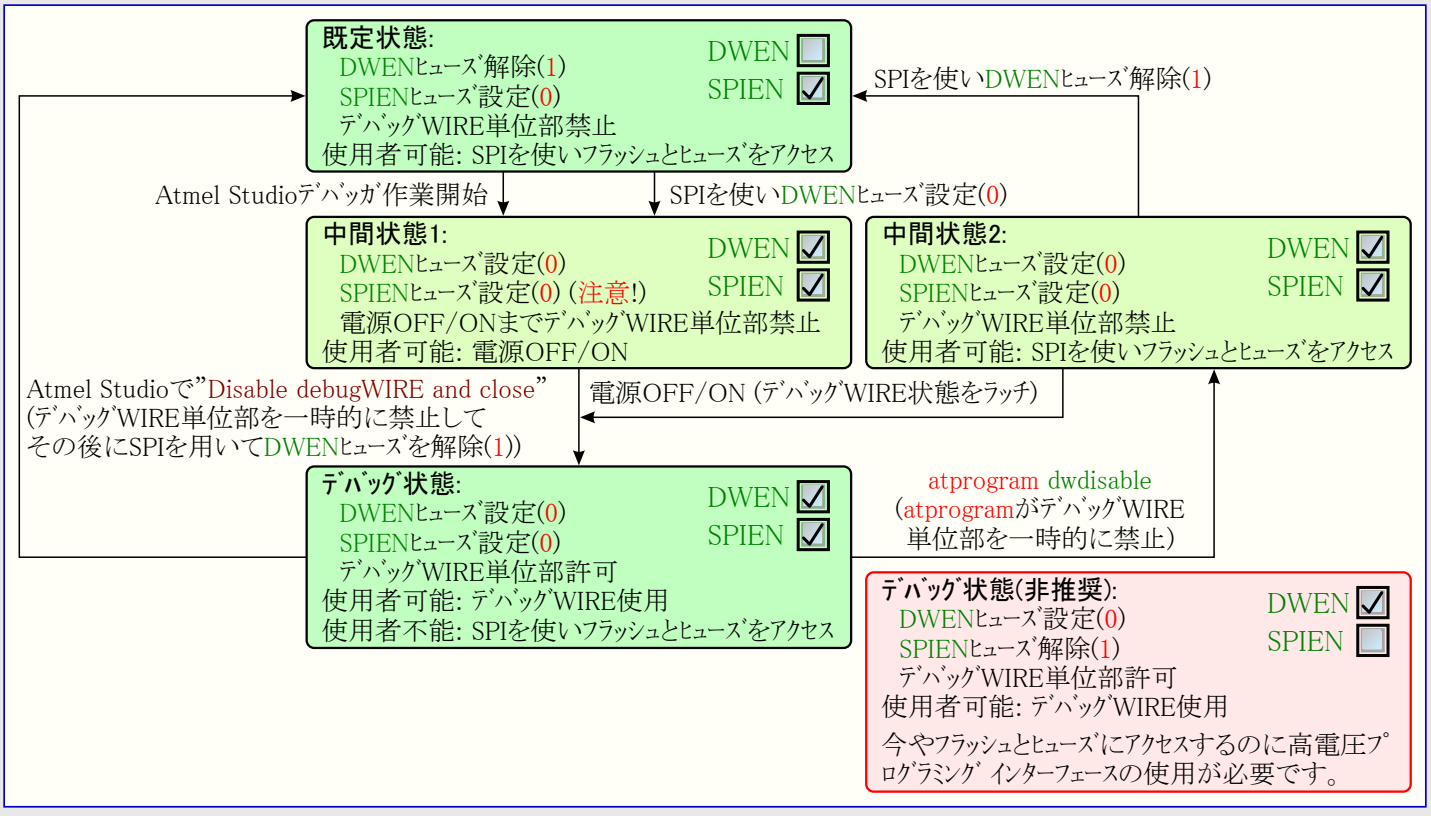
5.4.16. ﾃﾞﾊﾞｯｸﾞWIREとDWENﾋｭｰｽﾞの理解

許可されると、ﾃﾞﾊﾞｯｸﾞWIREｲﾝﾀｰﾌｪｰｽはﾃﾞﾊﾞｲｽの/RESETﾋﾟﾝの制御を取り、これはこのﾋﾟﾝも必要なSPIｲﾝﾀｰﾌｪｰｽに対して相互に排他的にします。ﾃﾞﾊﾞｯｸﾞWIRE単位部の許可と禁止する時はこれら2つの手法の1つに従ってください。

- 事態をAtmel Studioに処理させてください。(推奨)
- DWENを手動で設定(0)と解除(1)をしてください。(注意してください。上級使用者のみ!)

重要: 手動でDWEN操作時、高電圧ﾌﾟﾛｸﾞﾗﾐﾝｸﾞを使用しなければならないことを避けるために、SPIENﾋｭｰｽﾞが設定(0)に留まることが重要です。

図5-14. ﾃﾞﾊﾞｯｸﾞWIREとDEWNﾋｭｰｽﾞの理解



5.4.17. TinyX-OCD(UPDI)特別な考慮

UPDISﾃﾞｰﾀ(UPDI_DATA)ﾋﾟﾝは目的対象AVRﾃﾞﾊﾞｲｽに応じて専用ﾋﾟﾝまたは共用ﾋﾟﾝです。共用されるUPDIﾋﾟﾝは12V許容で、/RESETまたは汎用入出力(GPIO)として使うように構成設定することができます。これらの構成設定でのﾋﾟﾝ使用法の更なる詳細については「UPDI物理ｲﾝﾀｰﾌｪｰｽ」をご覧ください。

巡回冗長検査ﾒﾓﾘ走査(CRCSCAN:Cyclic Redundancy Check Memory Scan)単位部を含むﾃﾞﾊﾞｲｽではﾃﾞﾊﾞｯｸﾞ中にこの単位部が継続背面動作で使われるべきではありません。OCD単位部は制限されたﾊｰﾄﾞｳｪｱ中断点比較器資源を持ち、故により多くの中断点(ﾌﾞﾚｰｸ ﾎﾟｲﾝﾄ)が必要とされる時、またはｿｰｽ ﾚﾍﾞﾙ ｺｰﾄﾞ段階実行中でも、BREAK命令がﾌﾗｯｼｭ ﾒﾓﾘに挿入される(ｿﾌﾄｳｪｱ中断点)かもしれません。CRC単位部はこの中断点をﾌﾗｯｼｭ ﾒﾓﾘ内容の破損として不正に検出し得ます。

CRCSCAN単位部は起動の前にCRC走査を実行するように構成設定することもできます。CRC不一致の場合、ﾃﾞﾊﾞｲｽは起動せず、固まった状態に見えます。この状態からﾃﾞﾊﾞｲｽを回復する唯一の方法は完全なﾁｯﾌﾟ消去を実行して、有効なﾌﾗｯｼｭ ｲﾒｰｼﾞを書き込むか、または事前起動CRCSACNを禁止するかのどちらかにすることです(簡単なﾁｯﾌﾟ消去は無効なCRCを持つ空白ﾌﾗｯｼｭ ﾒﾓﾘに帰着し、従ってﾃﾞﾊﾞｲｽは未だ起動しないでしょう)。Atmel Studioはこの状態でﾃﾞﾊﾞｲｽをﾁｯﾌﾟ消去する時にCRCSCANﾋｭｰｽﾞを自動的に禁止します。

UPDIｲﾝﾀｰﾌｪｰｽが使われる目的対象PCBの設計時、正しい動作のために以下の考慮が行われなければなりません。

- UPDI線のﾌﾟﾙｱｯﾌﾟ抵抗器は10kΩよりも小さく(強く)てはなりません。ﾌﾟﾙﾀﾞｳﾝ抵抗器は使われるべきではなく、またそれはUPDI使用時に取り去られるべきです。UPDI物理層はﾌﾟｯｼｭﾌﾟﾙ能力があり、故に線がｱｲﾄﾞﾙの時に誤った開始ﾋﾞｯﾄ起動を防ぐために弱いﾌﾟﾙｱｯﾌﾟ抵抗器だけが必要とされます。
- UPDIﾋﾟﾝがRESETｲﾝとして使われるべきなら、それがｲﾝﾀｰﾌｪｰｽの正しい動作を妨害するため、UPDI使用時にどの安定化ｺﾝﾃﾞﾝｻも切断されなければなりません。
- UPDIﾋﾟﾝがRESETまたはGPIOのﾋﾟﾝとして使われる場合、それらがｲﾝﾀｰﾌｪｰｽの正しい動作を妨害するかもしれないため、ﾌﾟﾛｸﾞﾗﾐﾝｸﾞやﾃﾞﾊﾞｯｸﾞの間に線上の全ての外部駆動部が切断されなければなりません。

6. ﾊｰﾄﾞｳｪｱ説明

6.1. 概要

Atmel Power Debuggerﾊｰﾄﾞｳｪｱはﾃﾞﾊﾞｯｶﾞ、ｱﾅﾛｸﾞ前処理部、ﾃﾞｰﾀ交換器のこれら主要部分から成ります。ﾂｰﾙの論理的な構成がここで示されます。

図6-1. Power Debuggerの論理的な構成

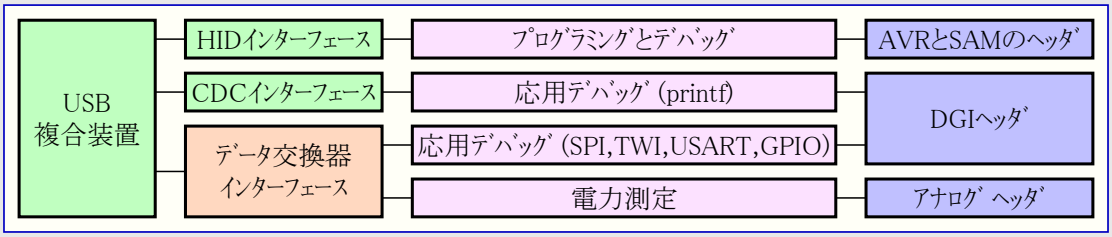
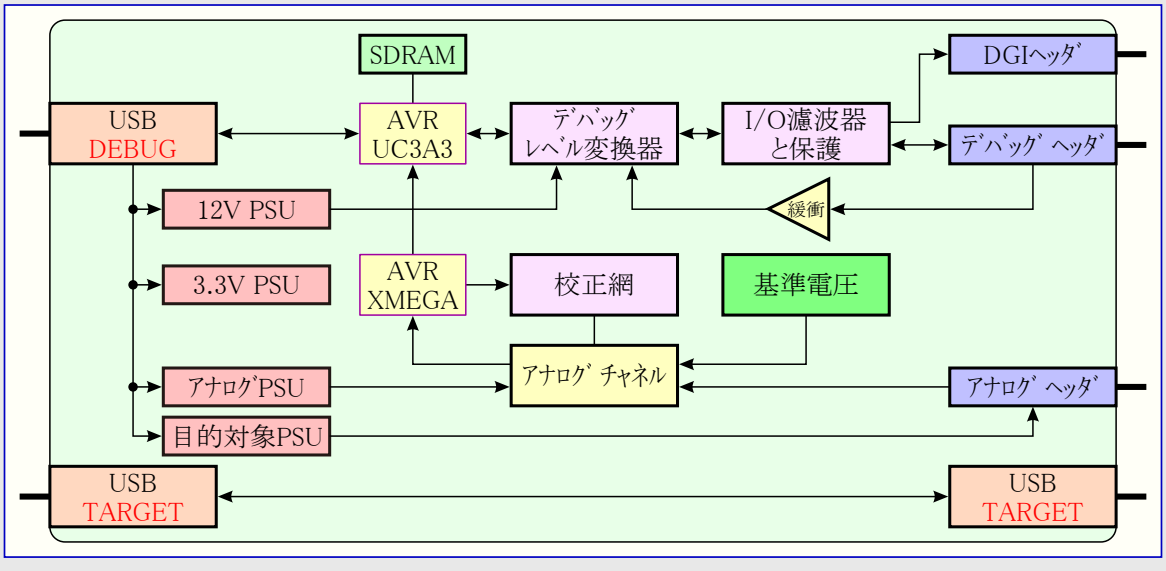


図6-2. Power Debugger構成図



電力はDEBUGﾎﾟｰﾄに接続されたUSBﾊﾞｽからPower Debuggerへ供給され、降下ｽｲｯﾁﾝｸﾞ動作調整器によって3.3Vに調整されます。VTGﾋﾟﾝは参照基準入力専用として使われ、独立した電源が基板上のﾚﾍﾞﾙ変換器の可変電圧側に供給します。Power Debugger主基板の心臓部は処理する作業に応じて最大84MHzで動くAtmel UC3ﾏｲｸﾛ ｺﾝﾄﾛｰﾗのAT32UC3A3256です。このﾏｲｸﾛ ｺﾝﾄﾛｰﾗはﾂｰﾙに対して高いﾃﾞｰﾀの単位処理量を許す、ﾁｯﾌﾟ上のUSB 2.0高速単位部を含みます。

6.2. ﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞ

Power Debuggerのﾃﾞﾊﾞｯｶﾞ部分はAtmel-ICEﾊｰﾄﾞｳｪｱに酷似しています。ﾃﾞﾊﾞｯｶﾞの心臓部はそれのﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｶﾞのAPI用USB HIDｲﾝﾀｰﾌｪｰｽを実装するAtmel AVR UC3ﾏｲｸﾛ ｺﾝﾄﾛｰﾗです。

Power Debuggerと目的対象ﾃﾞﾊﾞｲｽ上のﾌﾟﾛｸﾞﾗﾐﾝｸﾞとﾃﾞﾊﾞｯｸﾞのｲﾝﾀｰﾌｪｰｽ間の通信は目的対象の動作電圧とPower Debuggerの内部電圧水準間の信号を移し変える一揃いのﾚﾍﾞﾙ変換器を通して行われます。また信号経路にはPTCﾋｭｰｽﾞ、接地への過電圧保護ﾂｪﾅｰ ﾀﾞｲｵｰﾄﾞ、直列終端抵抗、誘導性濾波器、ESD保護ﾀﾞｲｵｰﾄﾞがあります。Power Debuggerのﾊｰﾄﾞｳｪｱそれ自体は5.0Vよりも高い電圧を駆動することができませんが、全ての信号ﾁｬﾈﾙは1.62～5.5Vの範囲で動作することができます。最大動作周波数は使う目的対象ｲﾝﾀｰﾌｪｰｽに従って変わります。

Power Debuggerは利便性のために3つの電気的に接続された出力ﾎﾟｰﾄを含みます。2つの50mil水平ﾍｯﾀﾞは各々、AVRの10ﾋﾟﾝ ﾋﾟﾝ配列とARM Cortexﾃﾞﾊﾞｯｸﾞ ﾍｯﾀﾞで使うためです。これらのﾎﾟｰﾄはAtmel-ICEで見つかるそれらと同じです。加えてPower Debuggerは独自接続用AVRﾋﾟﾝ配列で未実装の100milﾍｯﾀﾞを含みます。

重要: このﾍｯﾀﾞでの半田付けは使用者の責任で行われ、ESD予防処置が取られなければなりません。

6.3. ｱﾅﾛｸﾞ ﾊｰﾄﾞｳｪｱ

Power Debuggerのｱﾅﾛｸﾞ前処理部は'A'ﾁｬﾈﾙと'B'ﾁｬﾈﾙとして参照される2つのﾁｬﾈﾙを含みます。それらは同様の仕組みで動きますが、2つのﾁｬﾈﾙは釣り合ってなく、異なる目的に使われるべきです。

両ﾁｬﾈﾙはAVR XMEGA ATxmega128A1Uﾏｲｸﾛ ｺﾝﾄﾛｰﾗの独立したADCﾁｬﾈﾙに供給されます。このADC単位部は'A'ﾁｬﾈﾙに最大採取速度で完全に独立した動作を許す一方で、'B'ﾁｬﾈﾙのADCは両ﾁｬﾈﾙの田與測定で共用され、それは低い頻度で採取されます。

'A'ﾁｬﾈﾙは低電流を正確に測定するのに推奨されるﾁｬﾈﾙです。これは最大限界で100mAから1μAに落として測定する能力がある回路を提供する迂回切り替え器を通す2つの分流段が特徴です。範囲切り替えは自動で行われ、これが必要とされるべき場合にだけ、採取を高い範囲に固定化することが可能です。

1. 'A'ﾁｬﾈﾙ高範囲：100mA～500μA、約3μAの分解能
2. 'A'ﾁｬﾈﾙ低範囲：1mA～1μA、約30nAの分解能

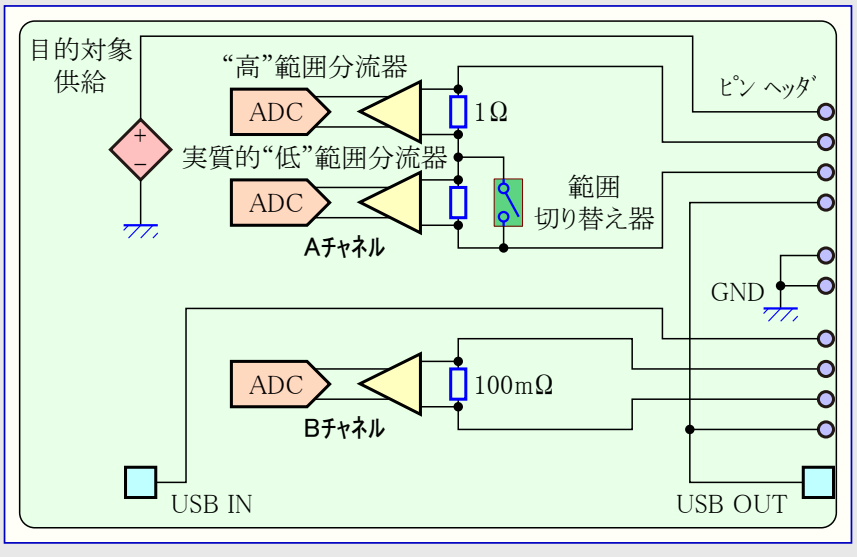
'A'ﾁｬﾈﾙの採取速度は63.5kHzで、ﾎｽﾄ ｺﾝﾋﾟｭｰﾀへ16ﾋﾞｯﾄ ﾌﾚｰﾑで送られます。校正された'A'ﾁｬﾈﾙは3%よりも悪くなく、約1μAに至る精度を持ちます。

'B'ﾁｬﾈﾙは'A'ﾁｬﾈﾙよりも低い分解能でより高い電流を測定するのに推奨されるﾁｬﾈﾙです。これは1Aまでと1mAに至る電流の測定を許す単一分流抵抗に基づきます。

1. 'B'ﾁｬﾈﾙ単一範囲：1A、約500μAの分解能

ﾃﾞｰﾀは62.5kHzで採取され、12ﾋﾞｯﾄ ﾌﾚｰﾑでｺﾝﾋﾟｭｰﾀへ転送されます。

図6-3. ｱﾅﾛｸﾞ構成図



6.3.1. アナログ ハードウェア校正

Power Debuggerの両アナログ チャネルは完全な測定精度を達成するのに校正を必要とします。ハードウェアは製造中に校正されますが、最高に正確な測定を得るために再校正を行うことも可能です。校正手順自身は自動ですが、校正手順を開始し得るのに先立ってアナログ チャネル ピンと電源ピン(Power Debugger上の電流計記号傍らのピンと電源記号傍らのピン)から全ての外部の線やジャンパを切り離すことが必要です。校正はPower Debuggerを繋ぐのに使われるソフトウェア前処理部を通して起動されます。

6.4. 目的対象電圧供給 (VOUT)

Power DebuggerはUSB DEBUGコネクタから1.6～5.5Vの範囲で最大100mAを供給する能力がある基板上の電圧供給を持ちます。

助言: 電圧供給はAチャネルで過電流が検出されると直ちにOFFに切り替わります。これはこの供給がAチャネルを通して接続された場合に目的対象回路を保護します。

6.5. データ交換器インターフェース

Power Debuggerは完全なデータ中継器インターフェース(DGI:Data Gateway Interface)を含みます。機能はAtmel EDBG装置によって装備されたAtmel Xplained Proキットで見つかるものと同じです。

データ交換器インターフェースは目的対象デバイスからコンピュータへデータを流すためのインターフェースです。これは応用のデバッグでの手助けとしてだけでなく、目的対象デバイスで走行する応用内の機能の実演用も意味されます。

DGIはデータ流し用の複数チャネルから成ります。Power Debuggerは以下の動作を支援します。

- USART
- SPI
- TWI
- GPIO0～3

DGIヘッダ上の全ての信号は目的対象デバイスの電力領域で動くようにレベルを移し変えられます。また、信号経路に於いては直列PTCヒューズ、接地への過電圧保護ツェナー ダイオード、直列終端抵抗、誘導性濾波器、ESD保護ダイオードがあります。Power Debuggerハードウェア自身が5.0Vよりも高い電圧を出力することができませんが、全ての信号チャネルは1.62～5.5Vの範囲で動作することができます。

重要: USART/SPIインターフェース上のシルク スクリーンでの矢印はデータ方向を示します。左向き矢印はUSART使用に対して目的対象デバイスのTX、SPI使用に対してMOSI線が接続されるべき受信部チャネルです。同様に右向き矢印は送信部チャネルでUSART使用に対して目的対象デバイスのRX、SPI使用に対してMISO線に接続されるべきです。SPI使用に対してPower Debuggerが従装置であることを忘れないでください。

重要: DGIはデバッガ部分と同じ参照基準電圧を使います。これはDGIヘッダとデバッグ ヘッダのREFピンが電気的に接続されていることを意味します。

重要: SPI動作とUSART動作は同時に使うことはできません。

重要: (0長が有り得る)I²C書き込み単位処理はI²C読み込み単位処理がEDBGによって応答される前に行われなければなりません。

6.6. CDCインターフェース

Power Debuggerは標準CDC仮想COMポート インターフェースを含みます。例えそれが論理的に独立したインターフェースでも、CDCヘッダは20ピンDGIヘッダの一部です。

重要: シルク スクリーンでの矢印はデータ方向を示します。左向き矢印は使用者が目的対象のTXを接続する受信部チャネルです。同様に右向き矢印は目的対象デバイスのRXが接続される送信部チャネルです。

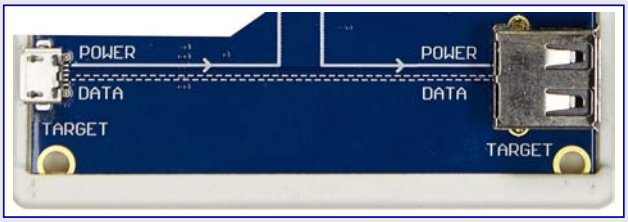
6.7. USBコネクタ

Power Debuggerはそれの下側に渡るUSB"通過"部分を含みます。これはUSB給電目的対象基板を素早く簡単な接続を許す便利な機能です。

USB通過部分は以下のように2つのUSBコネクタを持ちます。

- 左側の"TARGET"コネクタはUSBマイクロBジャックです。キットに含まれるケーブルを使ってこれをホスト コンピュータに接続してください。
- 右側の"TARGET"コネクタはUSB A型ジャックです。これをデバッグをする基板に接続してください。

図6-4. USB通過



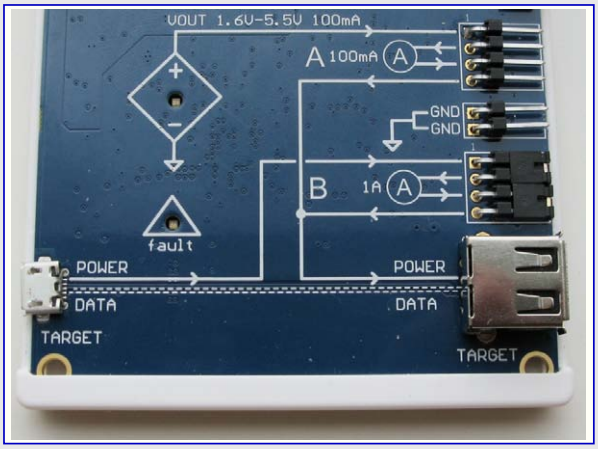
USB通過部分の配線は以下のように基板上のシルク スクリーンによってはっきりと示されます。

- DATA線はマイクロBジャックからAジャックへ無接触で通って渡されます。
- GNDは基板の接地に接続されます。
- SHIELDは1MΩの抵抗器を通して基板の接地に接続されます。
- POWER(VBUS)はマイクロBジャックからBチャネル ヘッダの1番ピンに配線されます。
- POWER(VBUS)はAとBの両チャネルの4番ピンからAジャックに配線されます。

POWER線の配線は以下の構成設定の1つを容易に接続することを使用者に許します。

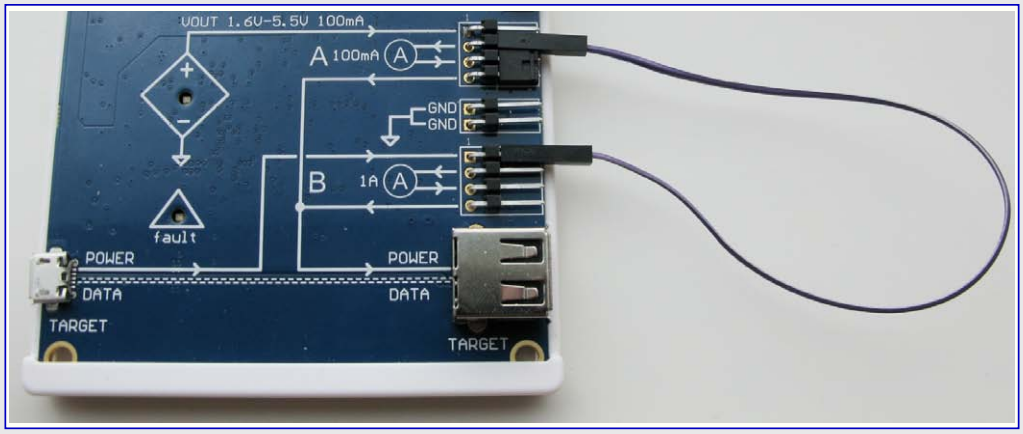
- 測定のためにBチャネルを通してTARGET USBコネクタからのUSB電力を配線。
  ここで示されるようにBチャネル ヘッダの両ジャンパを装着してください。

図6-5. ジャンパ構成設定



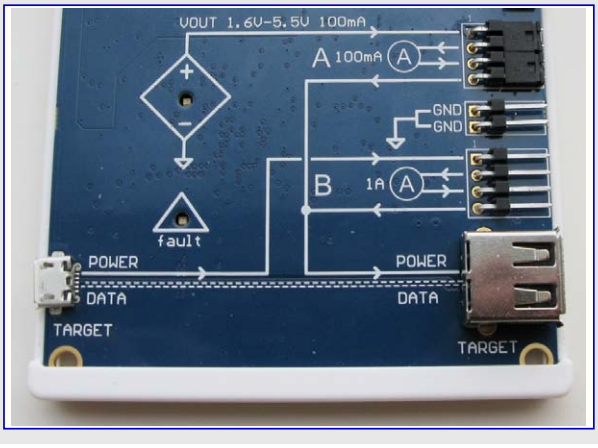
- 測定のためにAチャネルを通してTARGET USBコネクタからのUSB電力を配線。
  ここで示されるようにPOWERをAチャネルに接続するのに単独線を、出力を接続するのにジャンパを使ってください。

図6-6. 配線とジャンパ構成設定



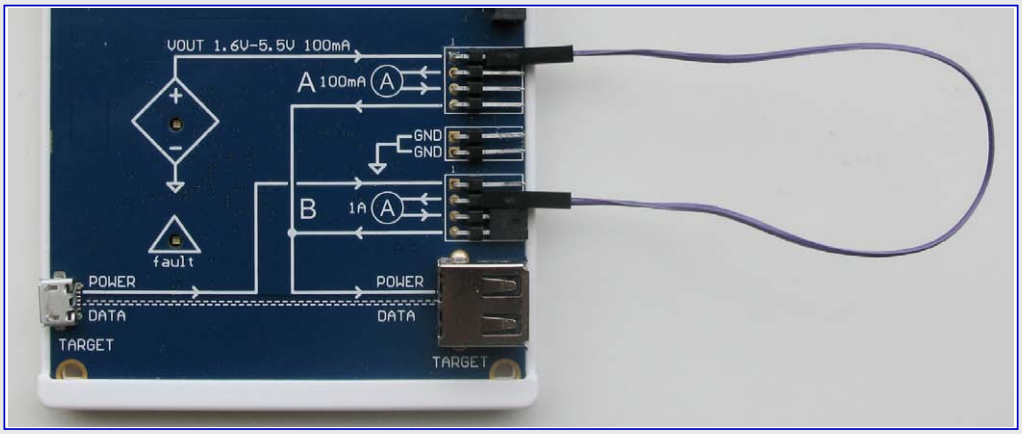
- Aチャネルを通る可変電圧源を通してDEBUG USBコネクタからのUSB電力を供給。
  ここで示されるようにAチャネルに両方のジャンパを装着してください。

図6-7. ジャンパ構成設定



・Bチャネルを通る可変電圧源を通してDEBUG USBコネクタからのUSB電力を供給。
ここで示されるようにPOWERをBチャネルに接続するのに単独線を、出力を接続するのにジャンパを使ってください。

図6-8. 配線とジャンパ構成設定



6.8. LED

Power Debuggerはツールの状態を示す5つのLEDを持ちます。デバッグLEDはAtmel-ICEと以前のデバッガ ハードウェアでのそれらの機能と同じです。Atmel-ICEに精通する人たちについてはPower Debuggerの中央部分全体がAtmel-ICEに似ています。

図6-9. デバッガLED

図6-10. 目的対象電圧供給の状態を示すのに使うVOUT LED

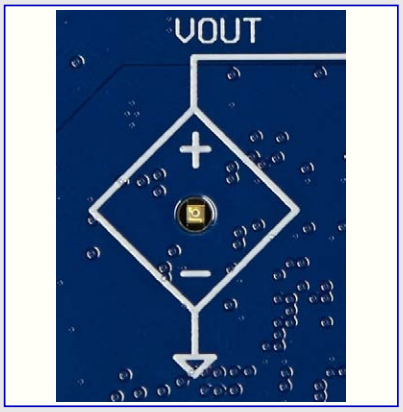


図6-11. アナログ採取回路の状態を示すのに使う障害LED

表6-1. LED

LED	機能	説明
TARGET	目的対象電力	目的対象電力がOKの時に緑。
POWER	主電力	主基板電力がOKの時に赤。
STATUS	状態	目的対象が走行/段階実行の時に点滅。目的対象が停止の時にOFF。
VOUT	目的対象電圧	目的対象出力電圧がONの時に緑。Aチャネルで過電圧が検出された時に点滅。
fault	アナログ障害	アナログ ピンで推奨動作条件外の電圧や電流が検出された時に赤。

7. 製品適法性

7.1. RoHSとWEEE

Power Debuggerと全ての付属品はRoHS指令(2002/95/EC)とWEEE指令(2002/96/EC)の両方に従って製造されます。

7.2. CEとFCC

Power Debugger本体は以下ような必須の必要条件とその他の関連指令条項に従って検査されています。

・2004/108EC(B級)指令
・FCC区分15副区分B
・2002/95/EC(RoHS,WEEE)

評価には以下の規格が使われます。

・EN 61000-1 (2007)
・EN 61000-3 (2007) + A1 (2011)

・FCC CFR 47区分15 (2013)

技術構成ﾌｧｲﾙは以下に置かれます。

Atmel Norway
Vestre Rosten 79
7075 Tiller
Norway

全ての努力がこの製品からの電磁放射を最小にさせました。けれども、或る条件下で、ｼｽﾃﾑ(目的対象応用回路に接続された本製品)は前述の規格によって許される最大値を超える個別電磁成分周波数を放射するかもしれません。この放射の周波数と大きさは使われる製品と目的対象応用の配置と配線を含む様々な要素によって決められます。

8. 公開履歴と既知の問題

8.1. ﾌｧｰﾑｳｪｱ公開履歴

表8-1. 公開ﾌｧｰﾑｳｪｱ改訂

ﾌｧｰﾑｳｪｱ版 (10進数)	日付	関連変更
1.16	2015年9月22日	Power Debuggerﾂｰﾙの初回公開
1.18	2015年11月18日	改善されたCDC機能 LED点滅部修正
1.45	2016年9月29日	UPDIｲﾝﾀｰﾌｪｰｽ(tinyXﾃﾞﾊﾞｲｽ)の支援を追加 USBｴﾝﾄﾞﾎﾟｲﾝﾄの大きさを構成設定可能にする。 全般的なﾊﾞｸﾞ修正

8.2. 既知の問題

・ﾎｽﾄまたは電源がTARGETﾏｲｸﾛ-B USBｺﾈｸﾀに接続されると同時にﾎｽﾄがDEBUG USBｺﾈｸﾀに接続されていない場合、Power DebuggerはTARGETﾏｲｸﾛ-B接続を通して部分的に給電され、POWER LEDが点滅するでしょう。TARGETﾏｲｸﾛ-B USBｺﾈｸﾀに何かを接続するのに先立って常にﾎｽﾄをDEBUG USBｺﾈｸﾀに接続することが推奨されます。

・ﾎｽﾄがDEBUG USBｺﾈｸﾀに接続されていない間に20ﾋﾟﾝDGIﾍｯﾀﾞのEVTﾋﾟﾝに電圧が印加される場合、EVTﾋﾟﾝに最大25mAの電流が流れるかもしれません。Power Debuggerのﾋﾟﾝ ﾍｯﾀﾞのどれかのﾋﾟﾝに何れかの電圧を印加するのに先立って常にﾎｽﾄをDEBUG USBｺﾈｸﾀに接続することが推奨されます。

・低ｲﾝﾀｰﾌｪｰｽ周波数/ﾎﾞｰﾚｰﾄでのﾌﾟﾛｸﾞﾗﾐﾝｸﾞやﾃﾞﾊﾞｯｸﾞ中のﾃﾞｰﾀ可視器(Daya Visualizer)での電流測定走行はﾃﾞｰﾀ可視器でPower Debuggerからの切断に終わるかもしれません。ｲﾝﾀｰﾌｪｰｽ速度の下限は目的対象型、ﾌﾗｯｼｭ ﾒﾓﾘの大きさ、ｲﾝﾀｰﾌｪｰｽ形式に応じて変わりますが、代表的に100～300kHzの範囲です。一般的に電流測定が目的対象上のﾁｯﾌﾟ上ﾃﾞﾊﾞｯｸﾞ ｼｽﾃﾑによって影響を及ぼされるため、ﾃﾞﾊﾞｯｸﾞ中に電流を測定することは推奨されません。

9. 改訂履歴

資料改訂	日付	注釈
42696A	-	初版資料公開
42696B	2015年3月	初版資料は未配布
42696C	2016年3月	・即時開始の手引きを更新 ・ﾃﾞﾊﾞｯｸﾞ ｹｰﾌﾞﾙのﾋﾟﾝ配列を追加 ・手直しされたﾌｧｰﾑｳｪｱ公開履歴表 ・いくつかの微細な更新と修正
42696D	2016年10月	UPDIｲﾝﾀｰﾌｪｰｽ追加とﾌｧｰﾑｳｪｱ公開履歴更新

本使用者の手引きはAtmelのPower Debugger使用者の手引き(Rev.42696D-10/2016)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。