AVR944 : AVR 8ビット マイクロ コントローラ用LED駆動部ライブラリ

応用記述

序説

Atmel® MSL系は簡潔で高電力のLED駆動部の組です。この応用記述はこれらのLED駆動部とインターフェースするためにAtmel AVR® 8ビット マイクロ コントローラの使い方を説明します。MSL系駆動部はそれらの内部レジスタに読み書きするためのTWIまたはSPIのインターフェースが特徴です。Atmel LED駆動部ライブラリは主装置として何れかのAVR 8ビット マイクロ コントローラを使用してそれらのレジスタへ読み書きするための使い易い覆い関数を提供します。このライブラリはTWIとSPIのインターフェースを支援します。下表は各AVRに対してこのライブラリで支援されるインターフェースを示します。

表1. Atmel LED駆動部ライブラリによって支援されるインターフェース

マイクロ コントローラ	インターフェース
megaAVR	TWI
XMEGA	TWI
tinyAVR	SPI,USI(SPI)

特徴

- Atmel LED駆動部のAtmel LED駆動部MSL系を制御するためのライブラリ
- Atmel XMEGA®とmegaAVR®用のTWIライブラリを内包
  - PhilipsのI²C規約互換
- Atmel tinyAVR®用のSPIライブラリを内包
  - tinyAVR用のSPIとして使用されるUSIライブラリも内包
- MCUと応用から独立した応用プログラミング インターフェース
- 容易な形態設定と使用

目次

1. 事前必要条件

この応用記述で使用されるLED駆動部ライブラリは以下の基本的な熟知が必要です。

- ライブラリがこのIDEを使用して書かれているため、Atmel Studio 7でのCプロジェクトのコンパイル
- SPIとTWIのインターフェースの全般的な熟知と電気的な接続の必要条件
- MSL系レジスタの基本的な知識とそれらの使い方
- AVR JTGICE mkⅡまたはAtmel-ICEのように、コンパイルされた応用をデバッグして検査するため、または応用hexファイルを目的対象デバイスに書き込むためのデバイスのプログラミングとデバッグ

2. 制限

- LED駆動部ライブラリはGCC Cコンパイラを持つAtmel Studio 7でコンパイルされて検査されました。このライブラリはIAR™または他のどのコンパラでもコンパイルされませんでした。
- LED駆動部ライブラリはAtmel megaAVRとXMEGAでのTWIとAtmel tinyAVRでのUSI(SPI)だけを支援します。
- 'ATXMEGA','MEGAAVR','TINYAVR','SPI','TWI','SPI_USI'は重要な必要語で、それらが形態設定ファイルで定義されるように使用されるため、コード内の他の場所でそれらを使用しないでください。

3. プロジェクト作成

下表は短い説明と共にこのライブラリに含まれるファイルの一覧を示します。

新しいプロジェクトにこのライブラリをインクルードするには、(以下の章で説明されるように)"atmel_led_device_config.h"が形態設定され、必要とされるドライバ ファイルとコンパイラ ファイル(avr_compiler.h)と共に追加されなければなりません。例えば、ATxmegaのTWIインターフェース用のプロジェクトを作成するには、以下のファイルがインクルードされるべきです。

- atxmega_twi_driver.c
- atxmega_twi_driver.h
- avr_compiler.h (*)
- atxmega_twi_driver.h

プロジェクトとしてAVR及びインターフェースと無関係にインクルードされなければならない形態設定とコンパイラ(*印)のファイルはそれらなしでコンパイルしません。

使用者応用ソース ファイルでは、先頭に次の文を追加することによって"atmel_led_device_config.h"をインクルードしてください。

```
#include "atmel_led_device_config.h"
```

このファイルは必要とされるファイル全てを自動的にインクルードします。"atmel_led_drvr_demo.c"はこのライブラリをインクルードして使用する方法の非常に良い例を提供します。

表3-1. Atmel LED駆動部ライブラリ内のファイルの一覧

ソース ファイル	説明
atmega_twi_driver.c	TWIインターフェースでのmegaAVR用駆動部
atmega_twi_driver.h	atmega_twi_driver.c用ヘッダ ファイル
atxmega_twi_driver.c	TWIインターフェースでのXMEGA AVR用駆動部
atxmega_twi_driver.h	atxmega_twi_driver.c用ヘッダ ファイル
tiny_avr_spi_via_usi_driver.c	USIインターフェースでのtinyAVR用駆動部
tiny_avr_spi_via_usi_driver.h	tiny_avr_spi_via_usi_driver.c用ヘッダ ファイル
tiny_avr_spi_driver.c	SPIインターフェースでのtinyAVR用駆動部
tiny_avr_spi_driver.h	tiny_avr_spi_driver.c用ヘッダ ファイル
atmel_led_device_config.h	Atmel LED駆動部ライブラリ形態設定ファイル
atmel_led_drvr_demo.c	Atmel LED駆動部ライブラリ実演応用
avr_compiler.h	AVRコンパイラ ファイル
documentation.h	Doxygenによってのみ使用

4. ライブラリ形態設定

1つのファイル”atmel_led_device_config.h”だけでライブラリを形態設定してください。このファイルは選択したAVRとインターフェースに対してだけ必要とされるソースをコンパイルするための事前コンパイル命令を含みます。ライブラリと共に提供された既定形態設定ファイルはSPIインターフェースでのAtmel ATtiny40用に構成設定されます。各種AVRマイクロ コントローラ用の形態設定ファイル設定はこの章で説明されます。

4.1. デバイスとインターフェースの選択

デバイスとインターフェースに関連する4つの定義があります。最初の2つの#defineで目的対象のAVRとインターフェースを選んでください。

```
//*****************************************************************/
// 使用者はここでデバイスを定義することができます。: ATXMEGA, MEGAAVR or TINYAVR
#define ATXMEGA
//*****************************************************************/
// ここでインターフェースを定義してください。: SPI, TWI or SPI_USI
#define TWI
```

この例はTWIインターフェースでのATxmegaとしての目的対象を示します。

’ATXMEGA’,’MEGAAVR’,’TINYAVR’,’SPI’,’TWI’,’SPI_USI’は重要な必要語で、コード内の他のどの場所でもそれらを使用しないでください。これらは必要とされるソース ファイルだけを選択的にコンパイルします。

次の2つの定義は以下を示します。

```
// AVRが動いているCPUクロック。これはボーレート設定と遅延の計算に使用されます。
#define F_CPU 20000000
//*****************************************************************
// ここで従装置アドレスを定義してください。
#define SLAVE_ADDRESS 0xA0
```

- F_CPUはAVRが動いているヘルツ(Hz)での周波数です。これは遅延とボーレートの設定を計算します。
- SLAVE_ADDRESSはMSL従装置のアドレスです。これはTWI従装置またはSPI従装置でも有り得ます。これは7ビット アドレスでビット0は無視されます。ビット0は読みまたは書きの動作を示すのに使用されます。

4.2. インターフェース設定

”atmel_led_device_config.h”ファイルには異なるインターフェースの種類に対して各々使用される4つの領域があります。使用されるAVRとインターフェースの形式に依存して4つの領域の1つがコンパイルされます。

4.2.1. TWIインターフェースでのAtmel XMEGA

以下のコード領域はこの目的対象に関連する定義を含みます。

```
//********* ATXMEGA TWIパラメータ ***********************************/
#if defined(ATXMEGA) && defined(TWI)
    #include "atxmega_twi_driver.h"
    /*! ¥brief アドレス バイトとレジスタ アドレスを於いて送受される最大メッセージの大きさ。
    配列読み書き関数使用時、計数パラメータはNUM_BYTESを超えられません。 */
    #define NUM_BYTES 16
    /*! 使用するTWIポート */
    #define TWI_PORT TWIC
    /*! TWI主装置割り込み */
    #define TWI_INT_VECTOR TWIC_TWIM_vect
    /* ボーレート 100kHz */
    #define BAUDRATE      100000
//*****************************************************************
```

- NUM_BYTESは1回の転送で送ることができる最大データ バイト数を定義します。これは配列読み/書き命令使用時に使用されます。計数パラメータはNUM_BYTESを超えることはできません。
- TWI_PORTはどのTWI単位部が使用されるのかを定義します。
- TWI_INIT_VECTORはそのTWI単位部と連携する割り込みを定義します。
- BAUDRATEはHzでの目的対象SCLクロック周波数を定義します。殆どのTWI応用について、これは100kHzです。

4.2.2. TWIインターフェースでのAtmel megaAVR

以下のｺｰﾄﾞ領域はこの目的対象に関連する定義を含みます。

```
//****************** megaAVR TWIﾊﾟﾗﾒｰﾀ ***************************/
#elif defined(MEGAAVR) && defined(TWI)
    #include "atmega_twi_driver.h"
    /*! ¥brief ｱﾄﾞﾚｽ ﾊﾞｲﾄとﾚｼﾞｽﾀ ｱﾄﾞﾚｽを於いて送受される最大ﾒｯｾｰｼﾞの大きさ。
    配列読み書き関数使用時、計数ﾊﾟﾗﾒｰﾀはNUM_BYTESを超えられません。 */
    #define NUM_BYTES 16
    // 8MHz CPUｸﾛｯｸ用ﾋﾞｯﾄ速度設定、100kHz SCL
    #define TWI_TWBR 0x20
    #define TWI_TWPS 0x00
//*************************************************************
```

- NUM_BYTESは1回の転送で送ることができる最大ﾃﾞｰﾀ ﾊﾞｲﾄ数を定義します。これは配列読み/書き命令使用時に使用されます。計数ﾊﾟﾗﾒｰﾀはNUM_BYTESを超えることはできません。
- TWI_BUFFER_SIZEは送信/受信の緩衝部の大きさを定義します。MSLでの単純な読み/書き転送では1方向で3ﾊﾞｲﾄが転送されます。
- TWI_TWBRとTWI_TWPSは下の式に従ってﾎﾞｰﾚｰﾄ ﾚｼﾞｽﾀ設定を定義します。SCLｸﾛｯｸ周波数を設定するためにこれらを慎重に入力してください。

$$\text{SCL周波数} = \frac{\text{CPUｸﾛｯｸ周波数}}{16+2\times TWBR\times(4^{TWPS})}$$

4.2.3. USIインターフェースでのAtmel tinyAVR

tinyAVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗの殆どはSPI単位部を持ちません。代わりにそれらはSPIまたはTWIのどちらかとして実装することができる多用途直列ｲﾝﾀｰﾌｪｰｽ(USI)を持ちます。このﾗｲﾌﾞﾗﾘはtinyAVRのUSIのSPI実装用のﾄﾞﾗｲﾊﾞを含みます。このﾗｲﾌﾞﾗﾘを使用してLED駆動部を制御するためにAVRのUSIﾎﾟｰﾄでMSLのSPIｲﾝﾀｰﾌｪｰｽを駆動してください。USIが3ﾋﾟﾝ ｲﾝﾀｰﾌｪｰｽのため、ｿﾌﾄｳｪｱによって追加入出力ﾋﾟﾝを$\overline{SS}$として使用してください。

以下のｺｰﾄﾞ領域はこの目的対象に関連する定義を含みます。

```
#elif defined(TINYAVR) && defined(SPI_USI)
    #include "tiny_avr_spi_via_usi_driver.h"

    /* ATtiny25用USIﾎﾟｰﾄとﾋﾟﾝの定義 */
    #define USI_OUT_REG          PORTB   //!< USIﾎﾟｰﾄ出力ﾚｼﾞｽﾀ
    #define USI_IN_REG           PINB    //!< USIﾎﾟｰﾄ入力ﾚｼﾞｽﾀ
    #define USI_DIR_REG          DDRB    //!< USIﾎﾟｰﾄ方向ﾚｼﾞｽﾀ
    #define USI_CLOCK_PIN        PB2     //!< USIｸﾛｯｸ入出力ﾋﾟﾝ
    #define USI_DATAIN_PIN       PB0     //!< USIﾃﾞｰﾀ入力ﾋﾟﾝ
    #define USI_DATAOUT_PIN      PB1     //!< USIﾃﾞｰﾀ出力ﾋﾟﾝ
    #define CSB_PIN              PB4     //!< Atmel LED駆動部CSB
    #define TC0_PRESCALER_VALUE  1       //!< 1,8,64,256,1024でなければなりません。
    #define TC0_COMPARE_VALUE    31      //!< 0～255でなければなりません。前置分周器=CLK/1で最小31
```

- USI_OUT_REGは使用されるUSIﾎﾟｰﾄの出力ﾚｼﾞｽﾀを定義します。
- USI_IN_REGはUSIﾎﾟｰﾄの入力ﾚｼﾞｽﾀを定義します。
- USI_DIR_REGはUSIﾎﾟｰﾄの方向ﾚｼﾞｽﾀを定義します。
- USI_CLOCK_PINはUSCK(SCK)ﾋﾟﾝを定義します。
- USI_DATAIN_PINはUSIﾃﾞｰﾀ入力(MISO)ﾋﾟﾝを定義します。
- USI_DATAOUT_PINはUSIﾃﾞｰﾀ出力(MOSI)ﾋﾟﾝを定義します。
- CSB_PINはｿﾌﾄｳｪｱで制御される入出力($\overline{SS}$)ﾋﾟﾝです。USIが3ﾋﾟﾝ ｲﾝﾀｰﾌｪｰｽのため、MSLxxxxCSB入力を駆動するためにｿﾌﾄｳｪｱで制御されるこのﾋﾟﾝを使用してください。
- TC0_PRESCALAR_VALUEとTC0_COMPARE_VALUEは下の式に従ってｲﾝﾀｰﾌｪｰｽのSCKのﾋﾞｯﾄ/秒(bps)を設定します。

(1,8,64,256,1024)の前置分周値と0～255の比較値を使用してください。1の前置分周値使用時は比較値を最小31で使用してください。従って最高SCKﾋﾞｯﾄ/秒はF_CPU/64です。

$$\text{bps} = \frac{F\_CPU}{TC0\_PRESCALAR\_VALUE\times(TC0\_COMPARE\_VALUE+1)\times 2}$$

4.2.4. SPIインターフェースでのAtmel tinyAVR

組み込みSPI単位部を持つtinyAVRマイクロ コントローラについては、LED駆動部ライブラリはAtmel MSL2160のようなAtmel LED駆動部とインターフェースするためのtinyAVR SPIドライバを含みます。以下のコード領域は組み込みSPI定義を含みます。

```
#elif defined(TINYAVR) && defined(SPI)
    #include "tiny_avr_spi_driver.h"
    // これはSPR[1:0]によって定義されるようなSCK分周器設定です。
    #define SPI_CLK_DIVIDER 4 //Must be 4, 16, 64, 128
    // これは設定(1)ならばSCKを2倍にするSPI2Xビット設定です。
    #define SPI_CLK_DOUBLE      1       // 1または0でなければなりません。
    #define SPI_DIR_REG         DDRC    //!< SPIポート方向レジスタ
    #define SPI_PORT            PORTC   //!< SPIポート出力レジスタ
    #define SPI_SCK             PC1     //!< SPIクロック入出力ピン
    #define SPI_MISO            PC2     //!< SPIデータ入力ピン
    #define SPI_MOSI            PC4     //!< SPIデータ出力ピン
    #define SPI_SS              PC0     //!< Atmel LED駆動部CSBピン
```

- SPI_CLOCK_DIVIDERとSPI_CLK_DOUBLEは下の式に従ってSCK周波数を決定します。(4,16,64,128)のクロック分周値と、0または1のクロック倍値を使用してください。
- SPI_DIR_REGはSPIポートの方向レジスタを定義します。
- SPI_PORTはSPIポートの出力レジスタを定義します。
- SPI_SCKはSCKピンを定義します。
- SPI_MISOはMISOピンを定義します。
- SPI_MOSIはMOSIピンを定義します。
- SPI_SSはMSLのCSBピンに接続される従装置選択ピンを定義します。

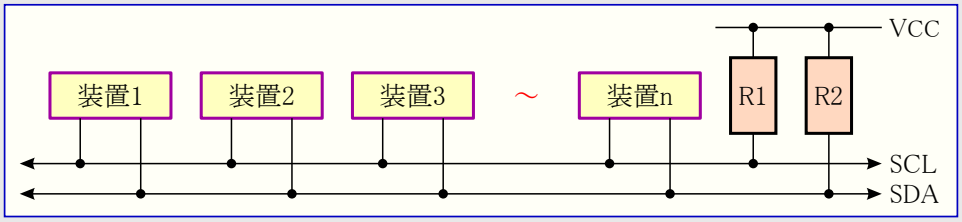
$$bps = \frac{F\_CPU}{SPI\_CLOCK\_DIVIDER} \times 2^{SPI\_CLK\_DOUBLE}$$

5. 電気的な接続

TWI

MSLをTWI従装置としてAVR(megaAVRまたはXMEGA)主装置に接続するには、下図で示されるように10kΩでSDAとSCLの線をプルアップしてください。この図ではR1とR2が10kΩの抵抗です。装置1～nは多数のMSL TWI従装置です。

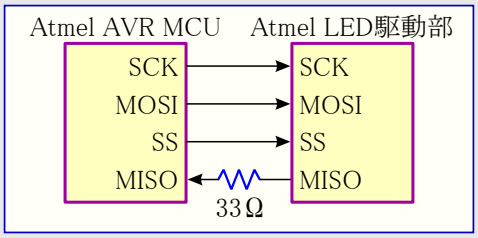
図5-1. TWI接続



SPI

MSLをSPI従装置としてtinyAVRのSPI(またはUSI)主装置に接続するには、MISO.MOSI,SCK,CSBをプルアップしませんが、下図で示されるようにAVRとMSL間のMISOに33Ωの直列抵抗を追加してください。

図5-2. SPI接続



6. ライブラリの使い方

本章はこのライブラリによって使用者に対して利用可能な関数を説明します。原型は使用されるAtmel AVRとインターフェースに関係なく同じままです。この目的のため、本章は"atmel_led_device_config.h"ファイルが正しく形態設定されると仮定します。

ライブラリは以下の応用プログラム インターフェース(API)を提供します。これらのより多くの情報は付随のDoxygen資料で見つけることができます。

atmel_led_drvr_init()

この関数はAVRを主装置として働くように形態設定します。従装置との更なる通信に先立ってこの関数を呼んでください。この関数はvoidを返します。

atmel_led_drvr_writeregister(slave_address, REG_ADDR. REG_DATA)

この関数はslave_addressでMSLxxxx従装置をアクセスし、それのREG_ADDRの内部レジスタにREG_DATAを書きます。この関数は成功の場合に1を、さもなければ0を返します。

atmel_led_drvr_readregister(slave_address, REG_ADDR, *receivedData)

この関数はslave_addressで従装置をアクセスし、それのREG_ADDRの内部レジスタのREG_DATAを読んでreceivedData位置指示子にデータを格納します。この関数は成功の場合に1を、さもなければ0を返します。

atmel_led_drvr_writearray(slave_address, REG_ADDR, *Data, count)

この関数はData位置指示子によって指示されるcount長のバイト配列をslave_addressの従装置のREG_ADDRのレジスタ アドレスで始まるレジスタに書きます。XMEGAとmegaAVRのデバイスに対して、このcountは形態設定ファイルで定義されたNUM_BYTESを超えることができません。

atmel_led_drvr_readarray(slave_address, REG_ADDR, *Data, count)

この関数はslave_addressの従装置のREG_ADDRのレジスタ アドレスで始まるレジスタからcount長のバイト配列をData位置指示子によって指示される緩衝部に読みます。XMEGAとmegaAVRのデバイスに対して、このcountは形態設定ファイルで定義されたNUM_BYTESを超えることができません。

使用者定義関数の書き方

3つの原始的関数の使用はLED駆動部の特定レジスタを読み書きするもっと使用者に友好的な関数を書くことを許します。例えば、Atmel MSL2160の全体輝度レジスタは$1Fの内部アドレスを持ちます。このレジスタは全体的なLED輝度を設定します。0～$FFの値が各々0～100%の明るさに対応します。下の例のコードは特定の従装置の全体輝度を設定するための簡単な覆い関数を示します。

```
char SetBrighntessLevel(char slave_addr, char intensity)
{
    return atmel_led_drvr_writeregister(slave_addr, 0x1F, intensity);
}
```

7. 実演プログラム

"atmel_led_drvr_demo.c"ファイルはAtmel MSL2160評価基板に接続するためのSPIインターフェースを持つAtmel ATtiny40 AVR用の実演プログラムのソースを含みます。このプログラムはMSL2160従装置をアクセスして'$00'のレジスタに'$AA'を書いてそのレジスタを読み戻します。この動作は無限繰り返しで行われます。

ハードウェア必要条件

この実演の構成設定は以下の構成品が必要です。

- MCUピンに触れさせる(Atmel STK®600のような)何れかの開発キットを持つATtiny40
- LED負荷基板と電源を持つMSL2160評価基板
- 33Ω抵抗

8. 参照

AVR151:SPIの初期設定と使用
http://www.atmel.com/images/atmel-2585-setup-and-use-of-the-spi_applicationnote_avr151.pdf

AVR311:I^{2}C従装置としてのTWI部の使い方
http://www.atmel.com/images/atmel-2565-using-the-twi-module-as-i2c-slave_applicationnote_avr311.pdf

AVR315:I^{2}C主装置としてのTWI部の使い方
http://www.atmel.com/images/atmel-2564-using-the-twi-module-as-i2c-master_applicationnote_avr315.pdf

AVR319:SPI通信へのUSI部使用法
http://www.atmel.com/images/doc2582.pdf

AVR1308:XMEGA TWIの使い方
http://www.atmel.com/images/doc8054.pdf

AVR1309:XMEGA SPIの使い方
http://www.atmel.com/images/doc8057.pdf

MSL2100ﾃﾞｰﾀｼｰﾄ
http://www.atmel.com/dyn/resources/prod_documents/F1_MSL2100_DB.pdf

MSL2160/61ﾃﾞｰﾀｼｰﾄ
http://www.atmel.com/dyn/resources/prod_documents/F1_MSL2160_DB.pdf

Atmel Studio 7
http://www.atmel.com/studio

9. 改訂履歴

資料改訂	日付	注釈
8464A	2011年11月	初版資料公開
8464B	2016年8月	新雛形といくつかの微細な更新

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA TEL:(+1)(408) 441-0311 FAX: (+1)(408) 436-4200 | www.atmel.com

---

本応用記述はAtmelのAVR944応用記述(Rev.8464B-08/2016)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。