PCのシリアルポートからPSoCのPWMを制御

pwm_uart_2

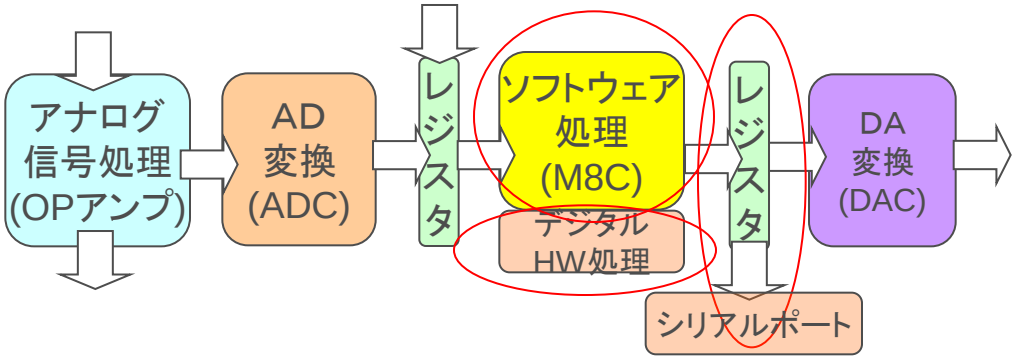
PSoC Experiment Lab

Experiment Course Material V1.03
April 9th., 2019
pwm_uart_2.pptx (13Slides)
Renji Mikami

ラボ

pwm_uart_2

シリアルポート経由の外部コマンドでPWM音源の音階発生



uart_1 ラボと timer_pwm2 ラボの統合

- timer_pwm2では,定期的に割り込みを発生させて,ハ長調の音階を発生させました。 PWM16ユーザーモジュールのwriteperiod()関数のperiod値をプログラムで書き換える(レジスタの値を書き換える)ことにより,ハードウェアを変更して音の周波数を変更して音階を発生させました。
- このlabでは, タイマー割り込みを使用せず,シリアルポートからコマンドを読み込んで,コマンドに応じた音階の音を発生させます。シリアルポートからのコマンド入力は、接続したPCのターミナルソフトウェア(TeraTerm)を使用します。
- PCのキーボードをシンセザイザの鍵盤として使用しPSoCからPWM音源の音を発生させます。
- コマンドは、a,s,d,f,g,h,j,k とlの9文字を使用し,それぞれがド,レ,ミ,ファ,ソ,ラ,シ,ドと無音に対応します。コマンドの発行は文字コードを入力し、Enterキー押すことで実行されます。

pwm_uart_2 プロジェクトの作成

- 先に作成したuart_1プロジェクトからpwm_uart_2クローン・プロジェクトを作り ,このプロジェクトにpwm16ユーザーモジュールを追加します。
- Cソースでシリアルポートから1文字を読み込みこの文字に対応した値をpwm16ユーザーモジュールのwriteperiod関数を使ってレジスタに書き込みます。
- UARTの制御手順は以下のとおりとなります
  - UART_CmdReset(); // Initialize receiver/cmd buffer
  - UART_IntCntl(UART_ENABLE_RX_INT); // Enable RX interrupts
  - UART_Start(UART_PARITY_NONE); // Enable UART
  - UART_CPutString("¥r¥nPSoC Synthesizer V1.1 ¥r¥n");
  - UART_PutString(strPtr); // Print out command
  - UART_CmdReset(); // Reset command buffer
- これら関数の詳細説明はuart_1 ラボに解説してあります.

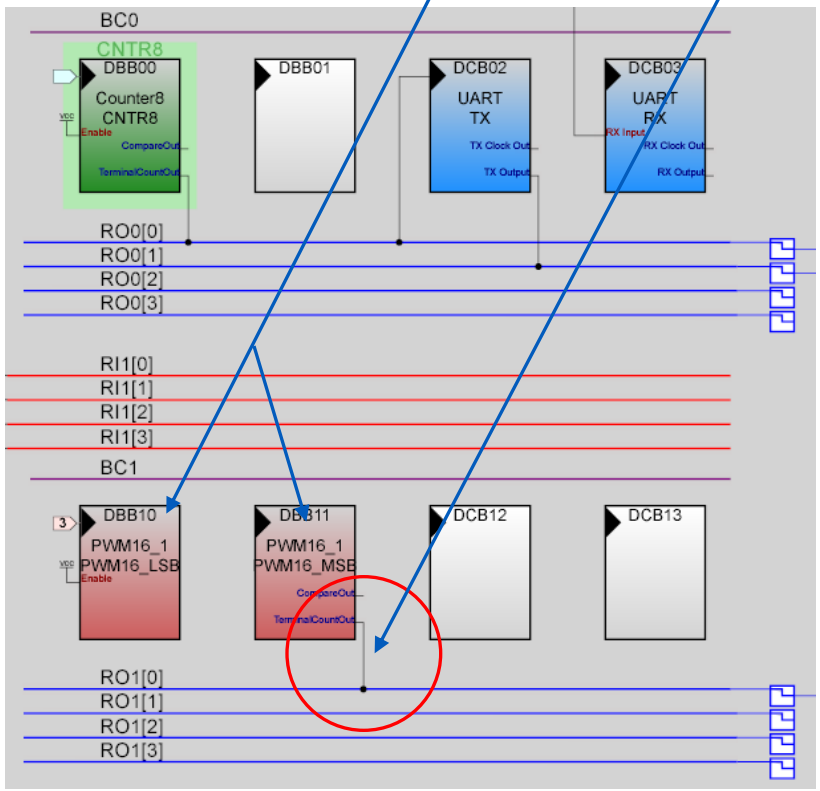
3210基板のジャンパなどの設定

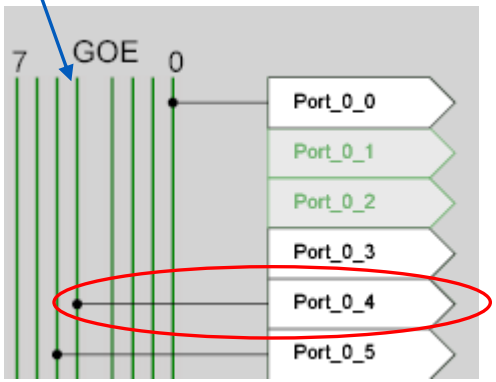
- ジャンパはJ1,J2,J3すべてオープン
- P03をRX,P05をTXに接続
- UART_2に進む場合は、P04にスピーカーの+（赤）を接続
- スピーカーの-(黒)は、GNDに接続
- EIA-574（DSUB9）ストレート通信ケーブルをPCと3210基板に接続

- MIniProgで書き込み,基板を Power Onする

PWM16ユーザーモジュールの追加配置と配線

- PWM16 UserModule をDBB10/11に配置
- PWM16のTerminal Count出力は,RO1[0]に接続し,
  ここからGOE4を経由してP0[4]ポートに接続





PSoCユーザーモジュールの設定

- 追加した PWM16の設定

Properties - PWM16_1	
Name	PWM16_1
User Module	PWM16
Version	2.5
Clock	VC3
Enable	High
CompareOut	None
TerminalCountOu	Row_1_Output_0
Period	0
PulseWidth	0
CompareType	Less Than Or Equal
InterruptType	Terminal Count
ClockSync	Sync to SysClk
InvertEnable	Normal

この値は、キーボド入力で自由に書き換えて周波数を変更します。

- uart_1 同じ設定のUM
  - Global Resource
  - Counter8
  - UART

Properties - Counter8	
Name	Counter8
User Module	Counter8
Version	2.5
Clock	SysClk*2
ClockSync	Unsynchronized
Enable	High
CompareOut	None
TerminalCountOut	Row_0_Output_0
Period	155
CompareValue	78
CompareType	Less Than Or Equal
InterruptType	Terminal Count
InvertEnable	Normal

Global Resources - uart_1	
CPU_Clock	3_MHz (SysClk/
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	1
VC2= VC1/N	16
VC3 Source	VC2
VC3 Divider	25
SysClk Source	Internal 24_MHz
SysClk*2 Disable	No
Analog Power	SC On/Ref Low
Ref Mux	(Vdd/2)+/-(Vdd/2)
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD (S	4.81V (5.00V)
LVDThrottleBack	Disable
Supply Voltage	5.0V
Watchdog Enable	Disable

Properties - UART	
Name	UART
User Module	UART
Version	5.2
Clock	Row_0_Output_0
RX Input	Row_0_Input_3
TX Output	Row_0_Output_1
TX Interrupt Mode	TXComplete
ClockSync	Unsynchronized
RxCmdBuffer	Enable
RxBufferSize	16
CommandTerminator	13
Param_Delimiter	32
IgnoreCharsBelow	32
Enable_BackSpace	Disable
RX Output	None
RX Clock Out	None
TX Clock Out	None
InvertRX Input	Normal

ソフトウェアのコーディング(1/2)

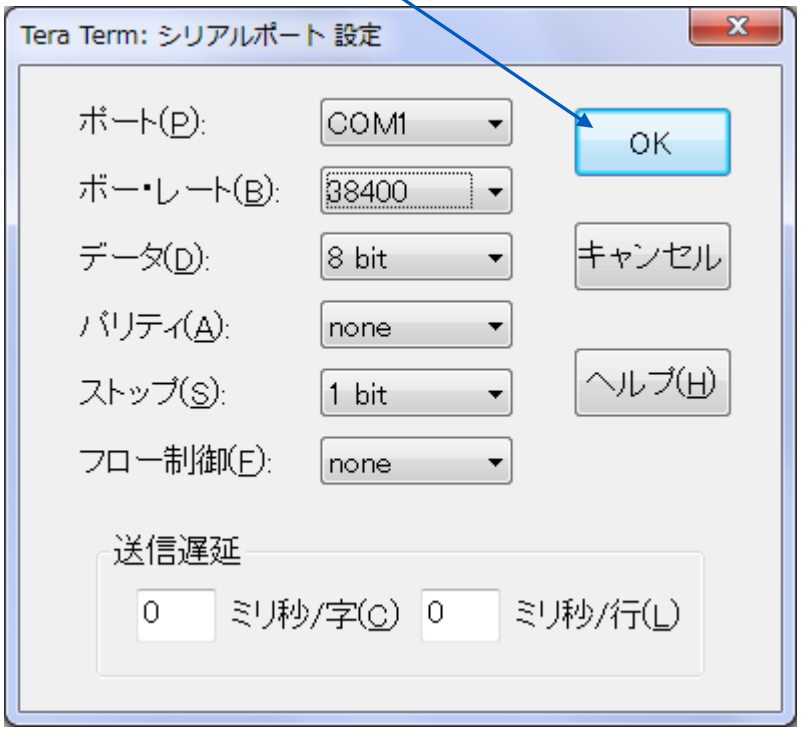
```
#include <m8c.h>
 #include "PSoCAPI.h"
 int PW=125;            // Pulse Width
 int SILENT=0, D=229, L=204, M=182, F=172, S=153, R=136, C=121, DD=115; //Scale
  void main(void)   {
  char * strPtr;                    // Parameter pointer
  UART_CmdReset();                  // Initialize receiver/cmd  buffer
    UART_IntCntl(UART_ENABLE_RX_INT);     // Enable RX interrupts
    Counter8_WritePeriod(155);          // Set up baud rate generator
    Counter8_WriteCompareValue(77);
    Counter8_Start();                 // Turn on baud rate generator
    UART_Start(UART_PARITY_NONE);       // Enable UART
     M8C_EnableGInt ;                 // Turn on interrupts
    PWM16_1_Start();
    UART_CPutString("¥r¥nPSoC Synthesizer V1.1 ¥r¥n");
     while(1) {
       if(UART_bCmdCheck())    {                          // Wait for command
          if(strPtr = UART_szGetParam())    {             // More than delimiter?
             UART_CPutString("Found valid command¥r¥nCommand =>");
             UART_PutString(strPtr);          // Print out command
             UART_CPutString("<¥r¥nParamaters:¥r¥n");
```

ソフトウェアのコーディング(2/2)

```
if (*strPtr=='a')
            PWM16_1_WritePeriod(D);
            else if (*strPtr=='s')
            PWM16_1_WritePeriod(L);
            else if (*strPtr=='d')
            PWM16_1_WritePeriod(M);
            else if (*strPtr=='f')
            PWM16_1_WritePeriod(F);
            else if (*strPtr=='g')
            PWM16_1_WritePeriod(S);
            else if (*strPtr=='h')
            PWM16_1_WritePeriod(R);
            else if (*strPtr=='j')
            PWM16_1_WritePeriod(C);
            else if (*strPtr=='k')
            PWM16_1_WritePeriod(DD);
            else if (*strPtr=='l')
            PWM16_1_WritePeriod(SILENT);  }
       UART_CmdReset();                    // Reset command buffer
      }
    }
  }
```

PCとの通信確認

- EIA-574（DSUB9）ストレート通信ケーブルをPCと3210基板に接続
- PCからTeraTermを起動,新しい接続はシリアルを指定,OKをクリック
- 設定＞シリアルポート（E）＞ボー・レートその他を下図のとおり設定,
  OKをクリック



以上で通信状態に入ります

キーボードから打ち込んだ文字を

EnterでPCのシリアルポートのTXに送信します

PCのキーボードを鍵盤に使う

- PCからTeraTermを起動し設定が完了したら,3210基板のRESETキーを押す. PCのTerminal画面に “PSoC Synthesizer V1.1” と表示されたら、正常に接続完了
- キーボードから、a<Enter>とタイプするとドの音が出ます
- s,d,f,g,h,j,kで順にレミファソラシドとlで(無音)のコントロールができます
- 各文字の送信は、<Enter>を押します
- 音楽の演奏ができましたか？

自由課題

- もしPCからコマンドを入力速度が、PSoCが処理できる速度を超えた場合には、どのような事態が発生するかを考察してみよう
- 人間の手入力ではエラーを発生させるような速度でPSoCのRXポートにコマンドを送信することは不可能と考えられるが,C++でコマンド文字コード送信プログラムを書いた場合は、高速コマンド送信が可能となる。
- それでは、現在のプログラムとハードウェアの構成では、どのくらいの速度でPSoCのRXポートに文字コードを送信したらエラーが発生するかを考察せよ
- エラーを回避しさらに高速で動作させようとした場合は、どのような改善をハードウェアとプログラムに加えたらよいかを考察せよ。

Memo

フォローアップURL

http://mikami.a.la9.jp/meiji/MEIJI.HTM

担当講師

三上廉司(みかみれんじ)

Renji_Mikami(at_mark)nifty.com (Default - Recommended)

mikami(at_mark)meiji.ac.jp (Alternative)

http://mikami.a.la9.jp/_edu.htm