ディジタルシステム設計
～尺八な音～

班：D班

氏名：洌鎌温子 (075726J)
津波古正輝 (075739A)
比嘉健人 (075753F)

提出日：7月29日 (水曜日)

ディジタルシステム設計

```
//このプログラムを動かすにあたって必要なヘッダーファイルの読み込み
#include "xparameters.h"
#include "xutil.h"
#include "xio.h"
#include "xgpio.h"
#include "xtime_l.h"
#include "xexception_l.h"
#include "xintc.h"
#include "xstatus.h"
#include "sleep.h"
#include "cdc_hw.h"
#include <string.h>
#include <stdio.h>

//装置の初期化と割り込み処理の設定を行う関数のプロトタイプ宣言
static XStatus Device_Init(void);
static XStatus Interrupt_Setup(void);
static void CDC_Handler(void);
short singen(void);

//装置操作の為の変数
XGpio      pushs;
XGpio      leds;
XIntc      intc;

//音に関係のある変数 buffer と基本となるアドレスを用意する
#define BUF_RX_BASE_ADDRESS  0x00000000
#define BUF_TX_BASE_ADDRESS  0x00001000
unsigned int *buf_rx;
unsigned int *buf_tx;
unsigned int buf_temp[8];
```

```
// サイン弦を生成
short y[3] = {0, 0x1492, 0};
short a1 = 0x6835;
short a2 = 0xc000;
short sin_temp;

//メイン関数部分
int main (void) {
  int i=0, j=0;
  XStatus sys_status;

  XExc_Init();

  //装置の初期化
  sys_status = Device_Init();
  if(sys_status != XST_SUCCESS){ return sys_status; }

  //装置の初期化
  sys_status = Interrupt_Setup();
  if(sys_status != XST_SUCCESS){ return sys_status; }

  //Initialize interface buffer (with debug value)
  //Assign Phisical Adderss to buffer
  buf_rx = (unsigned int *)BUF_RX_BASE_ADDRESS;
  buf_tx = (unsigned int *)BUF_TX_BASE_ADDRESS;

  //初期化
  for(i=0; i<8; i++){
    buf_rx[i]=0;
    buf_tx[i]=0;
    buf_temp[i]=0;
  }

  //Main Loop
```

```
  while(1){
    if (j==0) {
       REQ_RX_BUF;
       j=1;
    }

  }//End of Main Loop

   return 0;
}


static void CDC_Handler()
{
  static int l=0;
  static float x=1;//音の変化の為のカウンター変数
  static unsigned char p_in=0;

  p_in = XGpio_DiscreteRead(&pushs, 1);

  sin_temp = singen();

  XGpio_DiscreteWrite(&leds, 1, 0x0 );//4bit の LED を初期化

  //p_in のボタンが押されたら音を発生させる
  if (p_in == 1) {
     for(l=0; l<x; l++){
        buf_temp[l] = buf_rx[l];
        buf_tx[l] = buf_temp[l] + (int)(sin_temp << 3);
        //x が 8 を超えたら、1 にする。
        if(x>8){
          x=1;
        }
   }
   x=x+0.001; //カウンタ
```

```
}else if (p_in == 2) {
XGpio_DiscreteWrite(&leds, 1, 0x2 );//LEDを光らす
     for(l=0; l<2; l++){
        buf_temp[l] = buf_rx[l];
        buf_tx[l] = buf_temp[l] + (int)(sin_temp << 2);
     }

  }else if(p_in == 4){
      for(l=0; l<3; l++){
        buf_temp[l] = buf_rx[l];
        buf_tx[l] = buf_temp[l] + (int)(sin_temp << 2);
     }

  }else if(p_in == 8){
      XGpio_DiscreteWrite(&leds, 1, 0x8 );
      for(l=0; l<4; l++){
        buf_temp[l] = buf_rx[l];
        buf_tx[l] = buf_temp[l] + (int)(sin_temp << 2);
     }

  }else if(p_in == 16){
     for(l=0; l<5; l++){
        buf_temp[l] = buf_rx[l];
        buf_tx[l] = buf_temp[l] + (int)(sin_temp << 2);
     }

  }else if(p_in == 3){
     XGpio_DiscreteWrite(&leds, 1, 0x3 );
     for(l=0; l<6; l++){
        buf_temp[l] = buf_rx[l];
        buf_tx[l] = buf_temp[l] + (int)(sin_temp << 2);
     }

  }else if(p_in == 6){
```

```
      for(l=0; l<7; l++){
        buf_temp[l] = buf_rx[l];
        buf_tx[l] = buf_temp[l] + (int)(sin_temp << 2);
      }

  }else if(p_in == 12){
     XGpio_DiscreteWrite(&leds, 1, 0xc );
     for(l=0; l<8; l++){
        buf_temp[l] = buf_rx[l];
        buf_tx[l] = buf_temp[l] + (int)(sin_temp << 2);
     }

  } else {
      for(l=0; l<8; l++){
        buf_temp[l] = buf_rx[l];
        buf_tx[l] = buf_temp[l];
      }
  }

  SET_TX_BUF;
  usleep(1);
  FREE_TX_BUF;
}

//装置の初期化関数
static XStatus Device_Init()
{
  XStatus sys_status;

  sys_status = XGpio_Initialize(&leds, XPAR_LEDS_4BIT_DEVICE_ID);
  if(sys_status != XST_SUCCESS){ return sys_status; }
  XGpio_SetDataDirection(&leds, 1, 0x0);

  sys_status = XGpio_Initialize(&pushs, XPAR_PUSH_BUTTONS_POSITION_DEVICE_ID);
  if(sys_status != XST_SUCCESS){ return sys_status; }
```

```
  XGpio_SetDataDirection(&pushs, 1, 0x1);

  sys_status = XIntc_Initialize(&intc, XPAR_OPB_INTC_0_DEVICE_ID);
  if(sys_status != XST_SUCCESS){ return sys_status; }

  return XST_SUCCESS;
}


//装置のセットアップ関数
static XStatus Interrupt_Setup()
{
  XStatus sys_status;

  //Register Handlers

  //初期化
  XExc_RegisterHandler( XEXC_ID_NON_CRITICAL_INT,
                        (XExceptionHandler)XIntc_InterruptHandler,
                        &intc );

  sys_status = XIntc_Connect( &intc,
                              XPAR_OPB_INTC_0_OPB_CODEC_CNTLR_0_RX_INTR_INTR,
                              (XInterruptHandler)CDC_Handler,
                              (void *)0 );
  if(sys_status != XST_SUCCESS){ return sys_status; }

  XIntc_Enable(&intc, XPAR_OPB_INTC_0_OPB_CODEC_CNTLR_0_RX_INTR_INTR ); //[0]

  XIntc_Acknowledge(&intc, XPAR_OPB_INTC_0_OPB_CODEC_CNTLR_0_RX_INTR_INTR ); //[0]

  XExc_mEnableExceptions(XEXC_NON_CRITICAL);

  sys_status = XIntc_Start(&intc, XIN_REAL_MODE);
  if(sys_status != XST_SUCCESS){ return sys_status; }
```

```
  return XST_SUCCESS;
}


//サイン弦の生成関数
short singen () {

  y[0] = ((int)a1*y[1] + (int)a2*y[2]) >> 14;
  y[2] = y[1];
  y[1] = y[0];

  return(y[0]);

}
```

説明:

- static void CDC_ Handler
  割り込みを行う関数。p_in ボタンが押された時の外部からの入力に対しての処理を行う
- static XStatus Device_Init
  装置の初期化を行う関数
- static XStatus Interrupt_Setup
  装置のセットアップを行う関数
- short singen
  発生される音を生成する関数

p_in=1 のボタンが押された時、生成された sin 弦を l(エル) の値だけ加算し、音を生成する。sin 弦の加算回数を多くすれば多くするほど音が低くなっていく。つまり、sin 弦の加算回数を制御している l(エル) の数が多いほど音は低くなる。このことを利用し、l(エル) の回数を制限するカウンタ x を宣言し、p_in=1 ボタンでは押されるごとに音が低くなっていくようになっている。ここで注意しなければならないのが「ボタンを一回押す」という行為は CPU にとってはボタンを一回押していることにはならないことである。装置のボタンを一回押す瞬間に CPU の中ではもの凄い早さで main 関数の中を何回もループしている。なので、カウンタ変数 x はものすごい早さでインクリメントされていく。なので、x のインクリメントの値を大きくすると、確認できないほど瞬時に音が変化するので x のインクリメント値は小さくしなければならない。

p_in=2,4,8,16,3,6,12 のボタンはそれぞれ l(エル) の値を変えている。l(エル) の数字を大きくすると音が低くなる。p_in の数字が大きくなる l(エル) の値を大きくしているのでどんどん音が低くなっていく。

サイン弦 (sin_temp) は最初で宣言した short y[3]、short a1、short a2 を元に生成される。なので y[3]、a1、a2 を変更すると音が変化する。生成された sin_temp を左にシフトさせることで出力される音を大きくすることができる。p_in=1 ボタンでは、音の変化がわかりやすいように他の p_in ボタンよりも大きく音を出力したいので、3 ビットシフトさせている。

例；

```
if(p_in == 6)//p_in=6 ボタンが押された場合
        for(l=0;l<7; l++){//l の回数で音の高さが決まる
                buf_temp[l] = buf_rx[l];
                buf_tx[l] = buf_temp[l] + (int)(sin_temp << 2);//sin_temp のシフト数で出力される
                                                                //音の大きさが変わる
         }
```

p_in ボタン (2,3,8,12) を押すと LED ランプを光らすことができる関数 XGpio_DiscreteWrite() で p_in ボタンを押した時に LED が光るようになっている。第三引数のアドレス値を変えると、その数字を 2 進数であらわしている LED ランプの位置が光る。(LED ランプは右から左に向かって数字を表しているので、2 は 0010 となる。なので p_in=2 ボタンは右から 2 番目の LED ランプが光ることになる)

例；

```
XGpio_DiscreteWrite(&leds, 1, 0xc );//左から 1 番目と 2 番目の LED ランプが光る (1100)
```