ロボティクスプロジェクトの発表

1126100
前川大輝

発表の流れの説明

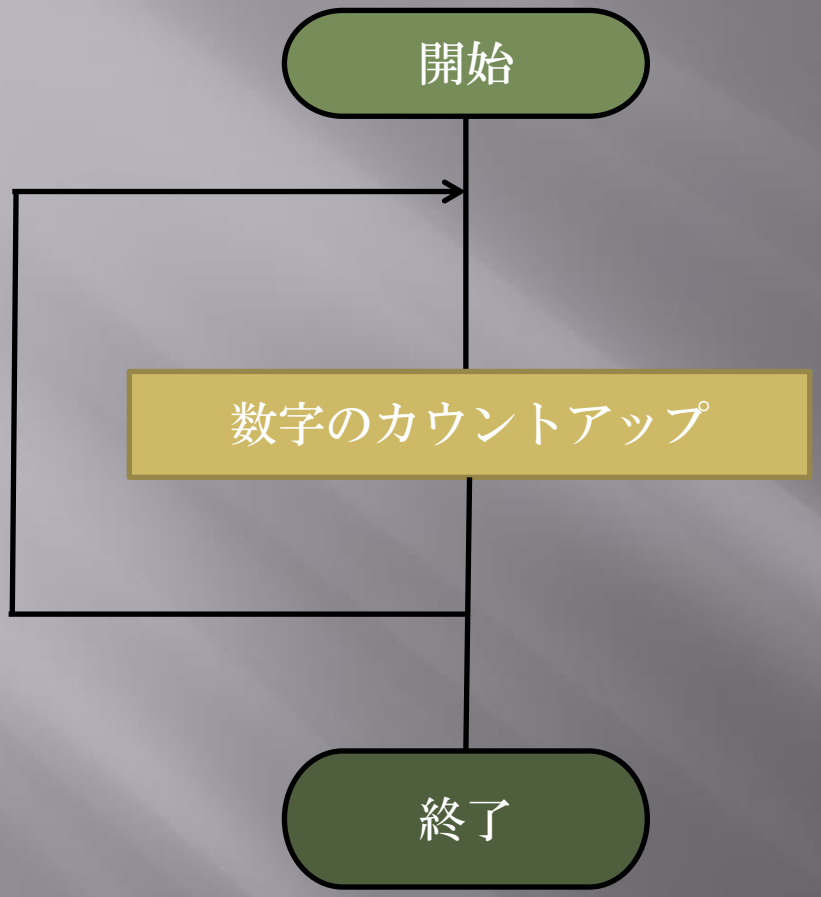
1．今回の学習目的
2．プログラム概要説明
3．実機によるデモンストレーション
4．LCDの基礎知識
5．プログラムソース解説
6．質疑応答

１．今回の学習目的

LCDに数字を表示すること

2.プログラム概要説明

１～９までの数字で一定時間ごとに表示
9の次は１に戻り無限ループ


開始
数字のカウントアップ
終了

カウントアップについて

１～9までの数字を0.1秒ごとに
カウントアップして
LCDの1マス目に表示する

それではさっそくデモンストレーションします。

LCDの基礎知識

R/S端子　0Vの時コマンドと認識

5Vの時文字データと認識

R/W端子　5Vの時リードモード

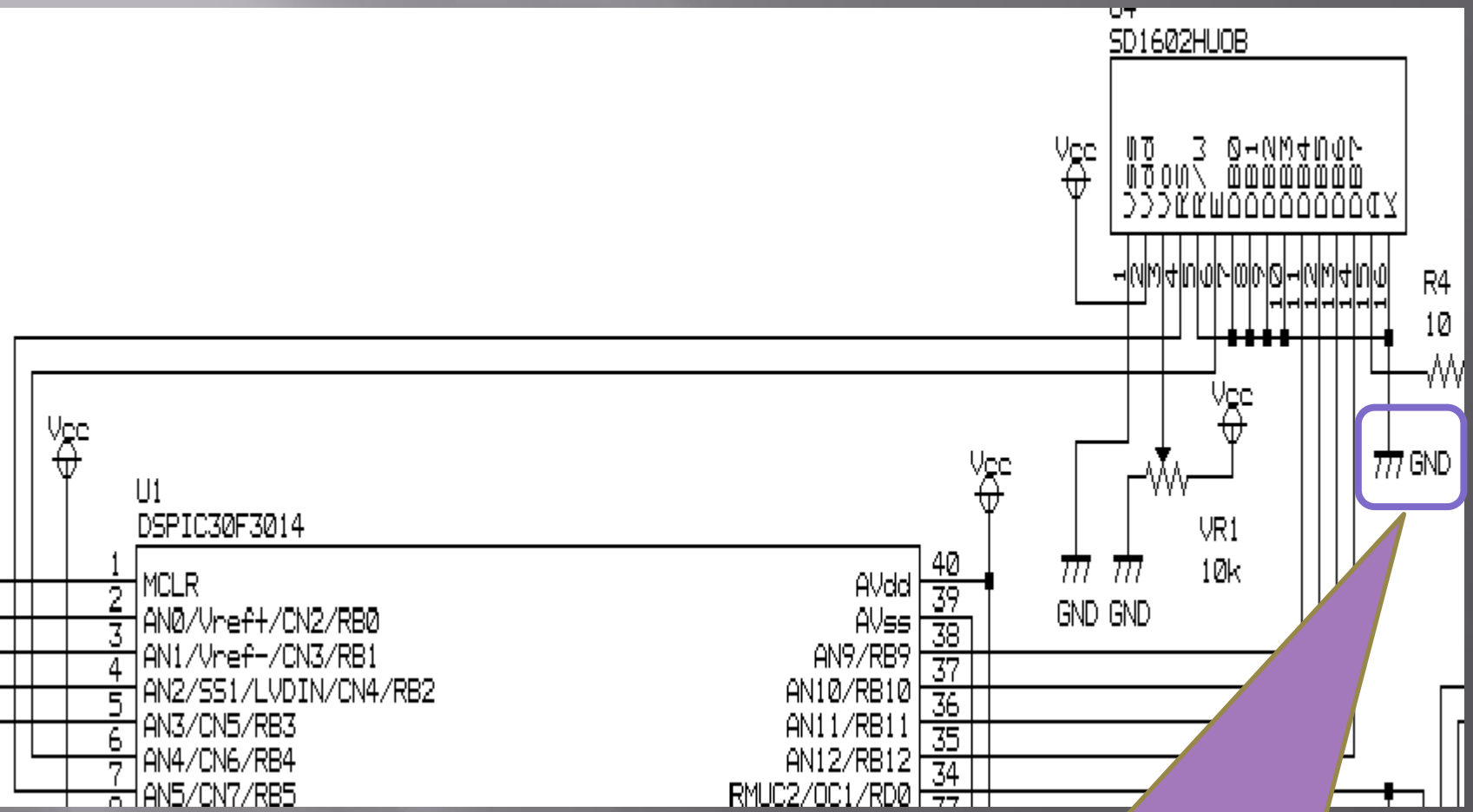
0Vの時ライトモード

E端子　0V→5Vに変化で命令を回収

今回は常にライトモード(R/WがGNDに落としてあるため考えなくてよい)

これにより意図的な命令であることと信号を送るタイミングを知らせることができる

LCDのR/W端子



R/Wは強制的に0V

LCDの基礎知識

LCDに用意されたコマンド

	↓各コマンドごとの１を立てる位置
Clear Display	0000 0001
Cursor At Home	0000 0010
Entry Mode Set	0000 0100
Display ON/OFF	0000 1000
Cursor/DisplayShift	0001 0000
Function Set	0010 0000
CGRAW AddressSet	0100 0000
DDRAW AddressSet	1000 0000

どの位置に１を立てるかでどのコマンドかが判別される

の8つです。詳しくはプログラム中で解説します

LCDの基礎知識

LCD初期化手順(仕様書に明記された方法)

電源ON
１５ms以上待つ
8ビットモードに設定
4.1ms以上待つ
8ビットモードに設定
100μs以上待つ
8ビットモードに設定
4ビットモードに設定
ファンクション設定
ディスプレイOFF
ディスプレイON
エントリーモード

プログラムソース解説

１．まず必要なヘッダーファイルを読み込みます。

```
#include ″p30F3014.h″
#include ″timer_def.h “
#include  “lcd.h”
```

lcd.hの中身

```
#define LCD_RS LATBbits.LATB5
#define LCD_E LATBbits.LATB4
#define LCD_D4 LATBbits.LATB9
#define LCD_D5 LATBbits.LATB10
#define LCD_D6 LATBbits.LATB11
#define LCD_D7 LATBbits.LATB12
#define CLOCK  80
```

lcd.hの中身(2)

関数を7個のLCD制御用関数を作成しました

8ビットモード用指令関数

```
void lcd_out8(unsigned char dat){
        //int i;

        LCD_RS = 0;
        LCD_E = 1;
        LCD_D4 = (dat & 0x01);
        dat = dat >> 1;
        LCD_D5 = (dat & 0x01);
        dat = dat >> 1;
        LCD_D6 = (dat & 0x01);
        dat = dat >> 1;
        LCD_D7 = (dat & 0x01);

        machi_usec(50);
        LCD_E = 0;
        machi_usec(50);

}
```

LCDにコマンドを送る

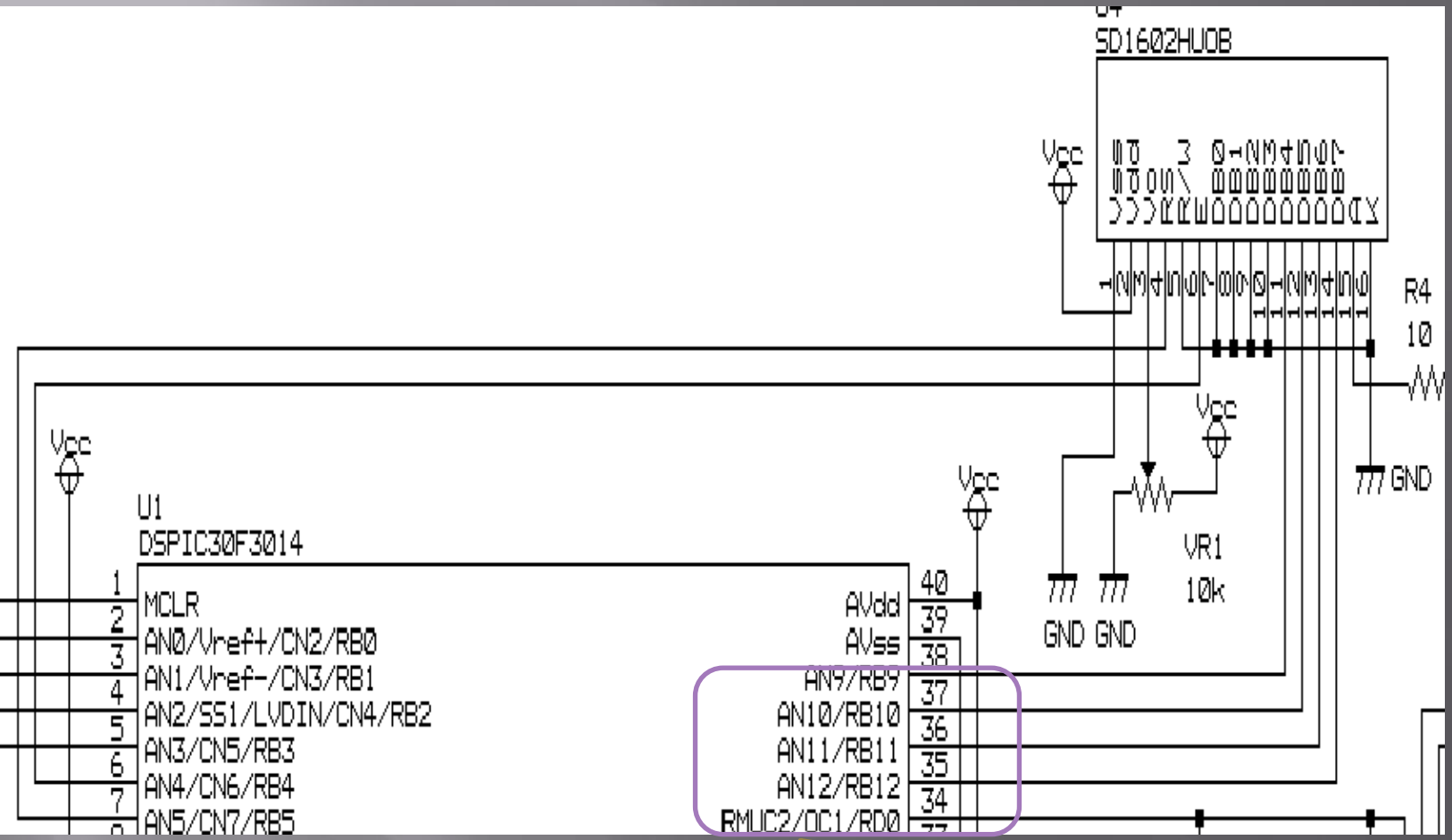
コマンドを回収させる
(450nsecかかる)

LCDの上位4本に指令を送る
なぜ一つずつ送っているのかは次のページで

上記の命令を回収

E端子を0Vに戻す

LCDの回路図



LCDの上位4ピンは
9,10,11,12
に接続されている.
char型は8ビットなので
まとめてLATBで出力にはできない

4ビットモード用指令関数

```
void lcd_out4(int rs,unsigned char dat){
        unsigned char bk = dat;

        LCD_RS = rs;
        LCD_E = 1;

        bk = bk >> 4;
        LCD_D4 = (bk & 0x01);
        bk = bk >> 1;
        LCD_D5 = (bk & 0x01);
        bk = bk >> 1;
        LCD_D6 = (bk & 0x01);
        bk = bk >> 1;
        LCD_D7 = (bk & 0x01);

        machi_usec(50);
        LCD_E = 0;
        //wait 50us
        machi_usec(50);

        LCD_E = 1;

        LCD_D4 = (dat & 0x01);
        dat = dat >> 1;
        LCD_D5 = (dat & 0x01);
        dat = dat >> 1;
        LCD_D6 = (dat & 0x01);
        dat = dat >> 1;
        LCD_D7 = (dat & 0x01);

        machi_usec(50);
        LCD_E = 0;
        machi_usec(50);
        LCD_RS = 0;
}
```

rs=0の時コマンド
rs=1の時文字データ

上位4ビット送信

下位4ビット送信

(例)
0010 1000
が送られてきたとする

順番だけ見れば
0010 0001
と認識しそうだが
LCD_D7が12ビットなの
を思い出せば
それぞれ反転すること
がわかる

↓

0010 1000

LCD初期化関数

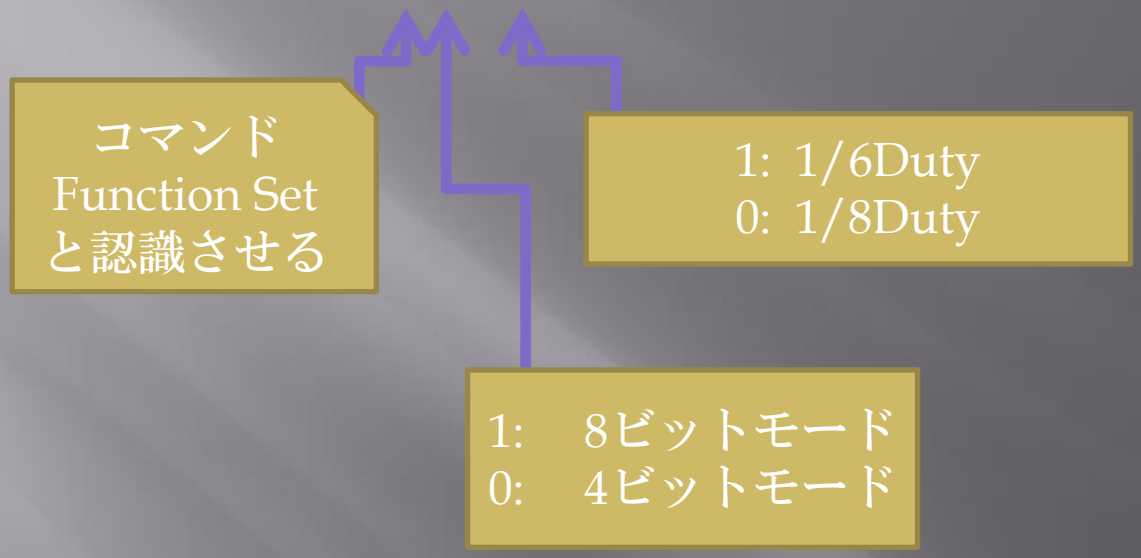
```
void lcd_format(void){

        machi_msec(20);
        lcd_out8(0x23);
        machi_msec(10);
        lcd_out8(0x23);
        machi_msec(10);
        lcd_out8(0x23);
        lcd_out8(0x22);
        lcd_out4(0,0x28);       //FunctionSet  0010 1000
        lcd_out4(0,0x0E);       //ディスプレイOFF 0000 1110
        lcd_out4(0,0x06);       //Entry ModeSet 0000 0110
        lcd_out4(0,0x02);       //カーソルをホームに移動
}
```

0010 0011 → (認識)0011 0000

0010 0010 → (認識)0010 0000

Function Setコマンド

8ビットモード　0010 0000



Display ON/OFF コマンド

0000 1000

コマンド
Display ON/OFF
と認識させる

3ビット目が
1: 液晶ON
0: 液晶OFF

2ビット目が
1: カーソルが表示
0: カーソルが非表示

1ビット目
1: カーソルが点滅
0: カーソルが点灯したまま

Entry Mode Set コマンド

0000 0100

コマンド
Entry ModeSet
と認識させる

2ビット目が
1: メモリ書き込みで　アドレスが+1
0: メモリ書き込みで　アドレスが-1

1ビット目
1: 表示もシフト
0: 表示はシフトせず

Cursor At Home コマンド

0000 0010

コマンド
Cursor At Home
と認識させる

1ビット目
0か１どちらでもよい

このコマンドは
カーソルをホームへ移動
表示内容は変化なし

1文字表示用関数

```
void lcd_data(unsigned char asci){

      lcd_out4(1,asci);
      machi_usec(50);

}
```

文字列表示用関数

```
void lcd_puts(unsigned char *str){
      while(*str != 0x00){
      lcd_out4(1,*str);  //文字列を一文字ずつ表示する
            str++;//ポインタ＋１することによって
      }
}
```

数字表示用関数

```
void lcd_convert(unsigned int number){

      lcd_data('0' + number);
}
```

画面消去関数

```
void lcd_clear(void){

      lcd_out4(0,0x01);
      machi_usec(1650);

}
```

コンフィギュレーション設定

_FOSC(CSW_FSCM_OFF & XT_PLL8);

8×10MHz(外部発振機)=
80MHz(クロック周波数)

CSW（クロック切り替え）FSCM(クロックのエラー検出)ともにOFF
（※クロック停止の際の処理→監視もしないし、内部の別クロック源に切り替えもしない）

_FWDT(WDT_OFF);

ウォッチドッグタイマ無効
(※特にシステムの監視を行わない場合は、これをOFFとする)

_FBORPOR(PBOR_ON & BORV_20 & PWRT_64 & MCLR_EN);

電源ON直後に動作リセット
電源OFF直後に動作停止
電源OFFを検知する電圧→2.0V
電源ON直後のリセットパルス幅→64msec
MCLRピン→有効

_FGS(CODE_PROT_OFF);

コードプロテクト→読み出し・書き込みともにOFF

main文

```
unsigned char msgA[] = "start";
        LATB = 0x0;
        TRISB = 0x00;

        lcd_format();              //LCD初期化

        lcd_puts(msgA);
        machi_msec(1000);

//timer1 (4/80MHz×64×31250=100msec)
ConfigIntTimer1(T1_INT_PRIOR_3 & T1_INT_ON);
OpenTimer1(T1_ON & T1_GATE_OFF & T1_PS_1_64 & T1_SYNC_EXT_OFF
                                        & T1_SOURCE_INT,31250-1);

while(1){
}
```

timer1呼び出し設定

```
//timer1 initialize function(4/80MHz×64×31250=100msec)
ConfigIntTimer1(T1_INT_PRIOR_3 & T1_INT_ON);
OpenTimer1(T1_ON & T1_GATE_OFF & T1_PS_1_64 &
T1_SYNC_EXT_OFF & T1_SOURCE_INT,31250-1);
```

優先度３
割り込み許可

ゲートタイマーモードOFF
（※外部信号の時間幅の測定に用いる）

タイマー
非同期

内部クロック使用

この式に基づきタイマー時間を算出しました

タイマー時間=(4/クロック周波数)×(プリスケーラ分周比)×(PR1設定-1)

timer1の中身

```
//timer 1 interval function 100[ms]
void _ISR _T1Interrupt(void){                    //Time1関数定義

        static unsigned int TimeCount1 = 0;
        IFS0bits.T1IF = 0;

        TimeCount1++;       //100msec間隔のインクリメント

        if(TimeCount1 <= 9){
                lcd_clear();
                lcd_convert(TimeCount1);

        }else{
                TimeCount1 = 0;
        }

}
```

割り込みサブルーチンの構文では割り込みフラグをクリアできないため

100msec毎に
1~9までの範囲でカウントアップする

5. 質疑応答

1. このプログラムのおかしいところ
2. あまりわからないところ
3. なんでもいいです。

以上にて発表を終了させていただきます。

ご清聴ありがとうございました。