2000 年 8 月

Ｈ８シングルボードコンピュータの機能評価装置の製作

東京工業高等専門学校　情報工学科

藤原　智仁

指導教官　東京工業高等専門学校　情報工学科
小坂　敏文

【1】　目的

本製作の目的はマザーボード付き H8 シングルボードコンピュータ（AKI-H8）の機能評価装置を製作することであり、次の４つの部分よりなる。

第１に H8 シングルボードコンピュータ（AKI-H8）の出荷時構成の入出力の動作確認を行う。

（1） Ｈ８シングルボードとパソコンをつないだシリアル通信のテスト

（2） キーボードから与えられた値をつなげ、整数として返す

（3） Ｈ８シングルボード付属の押しボタン SW、８ビット SW による LED のテスト

（4） LED の ON-OFF を利用した割り込みルーチンのテスト

第２に汎用入出力部を製作し動作確認を行う。

（1） DAC 端子の引き出しとテスト

（2） ADC 端子の引き出しとテスト

（3） ビット出力端子の引き出しと１２V 出力コントロールのテスト

（4） ビット入力端子の引き出しと SW のテスト

第３に PWM と位相カウント機能をもちいたモータコントロール部を製作し動作確認を行う

（1） PWM 端子の引き出しとモータのテスト

（2） 位相カウント端子の引き出しとモータに取り付けたロータリーエンコーダのテスト

第４にモータコントロールプログラムを製作し動作確認を行う

（11）モータをモータに取り付けた SW の信号をもとに制御する

（12）モータをロータリーエンコーダの信号をもとに制御する

【2】ハードウェア作成

1．　基本構想

目的を実現するため、図２－１に示す構成のハードウェアを製作する。

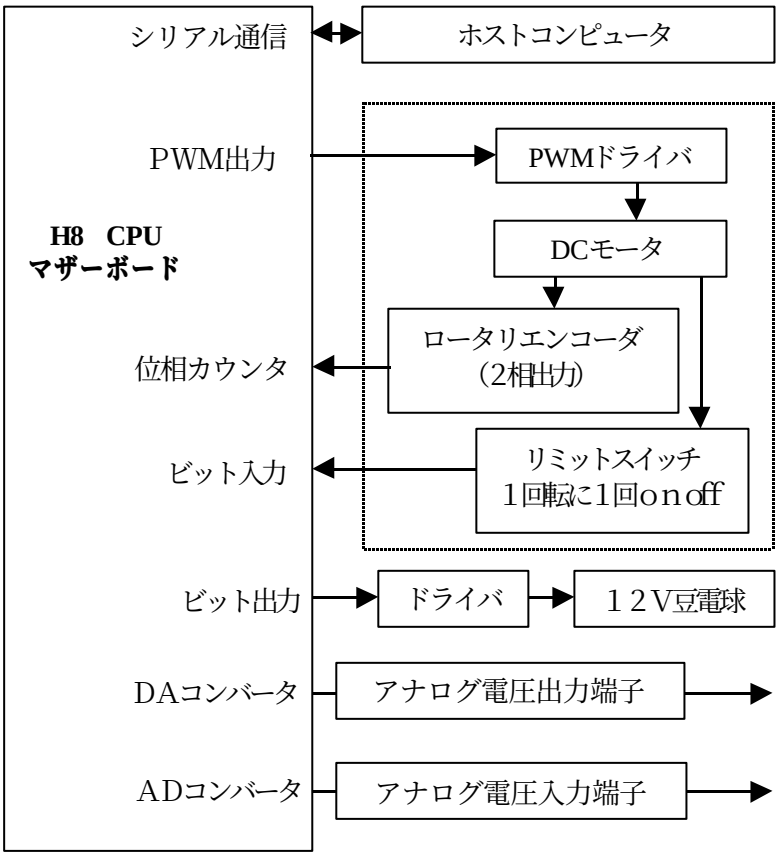


図２－１　Ｈ８シングルボードコンピュータの機能評価装置の基本構造

２．　各仕様

２．１　マイコンの仕様

機能評価装置の制御部分として日立製１６ビットＣＰＵ　Ｈ8/３０４８Ｆを使用した。表２－２－１に主な仕様を示す。図２－２－１は実際の CPU である.

表２－２－１　Ｈ8/３０４８Ｆの主な仕様

メモリ	ROM	１２８Ｋバイト	外部拡張可能
	RAM	４Ｋバイト	外部拡張可能
周辺機器	ITU	１６ビットタイマ×５ＣＨ	
	TPC	４ＣＨパルス出力	
	WDT	ウオッチドックタイマ	インターバルタイマとして使用可能
	SCI	独立２ＣＨ	
	A/D	１０ビット分解能×８ＣＨ	サンプルボード内臓
	D/A	８ビット分解能×２ＣＨ	
	I/O ポート	入出力端子７８本(最大)	

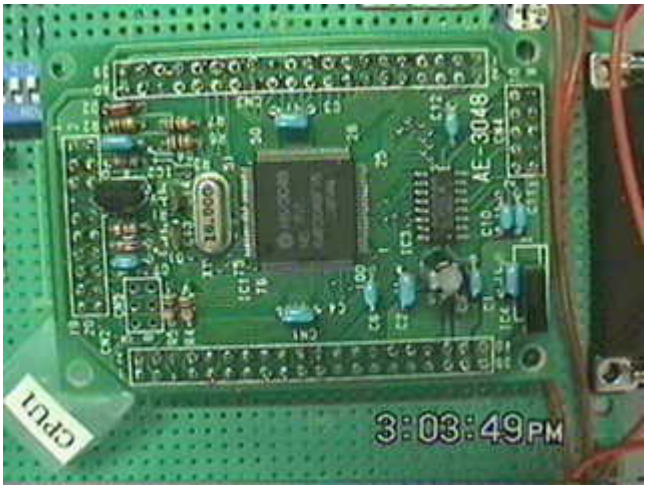


図２－２－１　CPU 本体

２．２　PWM 方式モータアンプ（PWMドライバ）

マイクロコンピュータからの信号線電流ではモータを駆動させるための電流が不充分であり、モータ駆動時に大きなノイズが発生するため、モータドライバを使用した。PWM 方式を用いるとアナログ信号に変換する必要がないため制御が容易となり、また放熱が少なく効率的である。さらに光アイソレータを用いることによってモータの発生するノイズをＣＰＵ側に与えないようにできる。よって、パワー MOS FET を用いた PWM 方式モータアンプを用いる。

モータアンプの役割、特徴は以下の通りである。
パワー MOS FET４個で H 型に構成され、単一電源でモータの正転逆転方向に電流を供給することができる。
マイクロコンピュータから入力した PWM 信号（PWM）と正転逆転信号（DIR）の２つの信号からモータに電流を供給し、モータを駆動させる。またブレーキ信号（BRK）によってモータを止めることもできるようになっている。
外部電源は DC１２V 使用している。

H 型電源供給を用いてのモータへの電流供給方法を、図２－２－２を用いて説明する。
PWM 信号がハイ状態の時は正転逆転信号に従って信号生成部内の両 1 番ピンか、両 2 番ピンの出力をする。PWM 信号がロー状態の時は両 1 番ピン、両 2 番ピンどちらも正転逆転信号に関係なく出力はしない。そして PWM 信号がハイ状態の時、なおかつ正転逆転信号がハイ状態の時信号生成部内の両 1 番ピンがアクティブ状態になり、両 2 番ピンはノンアクティブ状態になる。そして両 A 番の FET に電流が流れるこ

とにより「MOTOR ＋」に正電圧が掛かる。PWM 信号がハイ状態の時、なおかつ正転逆転信号がロー状態の時は両 1 番ピンがノンアクティブ状態になり両 2 番ピンがアクティブ状態になる。そして両 B 番の FET に電流が流れることにより「MOTOR －」に正電圧が掛かる。その結果単一電源でモータを正転方向、逆転方向に回転させることができる。

図２－２－３、図２－２－４に PWM 方式モータアンプの回路図と実装図を示す。
なおこの PWM 方式モータアンプは東京高専機械工学科の松林勝志氏の設計によるものである。
図２－２－５は実際の PWM ドライバである.

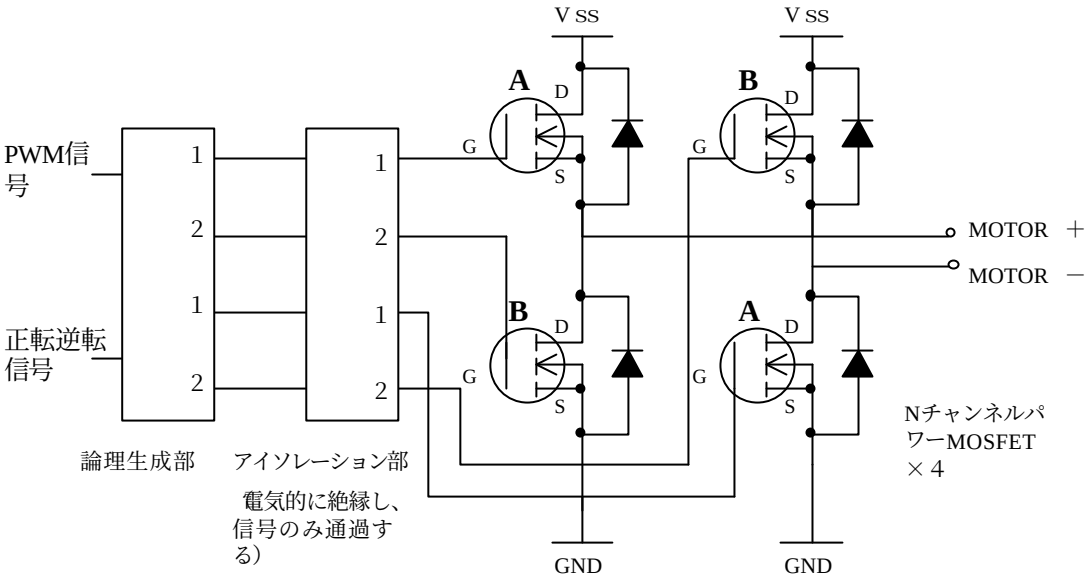


図２－２－2　PWM 方式モータアンプの H 型電源供給構成図

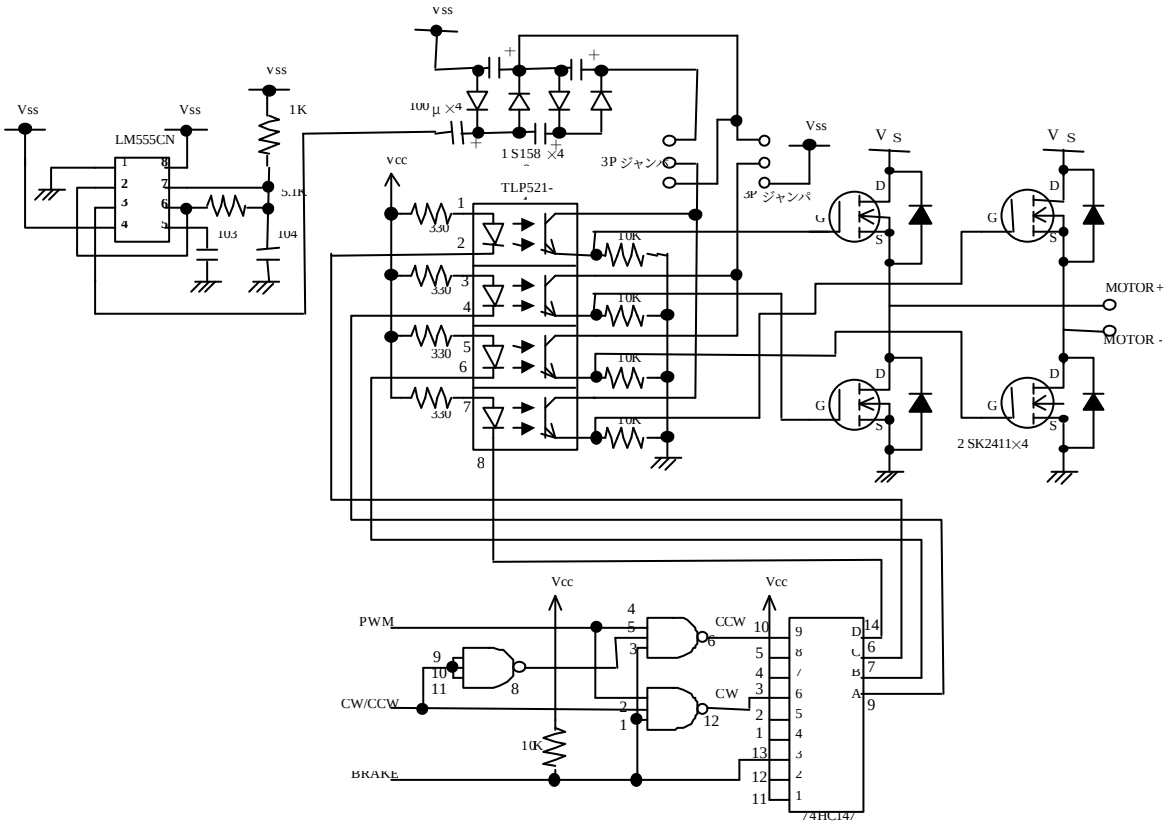


図２－２－3　モータアンプ回路図

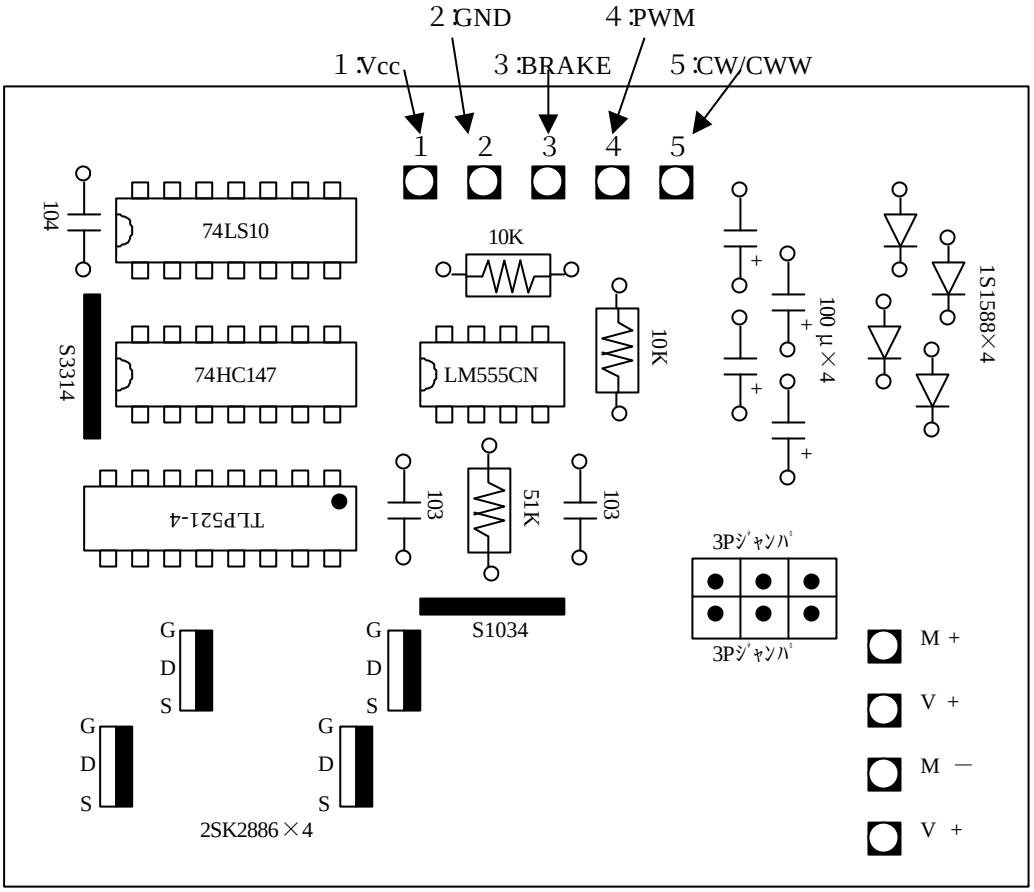


図２－２－４　基板配置図

図２－２－５　PWM ドライバ（図２－２－４を左側から見たところ）

２．３　ギアドＤＣモータの仕様

ツカサ電工社製 TG－23A－SG－60－KA を使用した。表２－２－２に仕様を示す。
実際のモータを図２－２－６に示す。

表２－２－２　モータの仕様

定格電圧	12[V]
定格トルク	40[g・cm]
ギヤトルク	147[g・cm]
定格回転数	5700[rpm]
ギヤ回転数	92.2[rpm]
定格電流	280[mA]
重量	75[g]

図２－２－６　DC モータ

２．４　ロータリーエンコーダの仕様

オムロン社製ロータリーエンコーダＥ６Ｃ－ＣＷ１００からのＡ相信号Ｂ相信号を、Ｈ８ＣＰＵの位相カウンタの引き出し端子に与えた。表２－２－３に仕様を示す。
実際のロータリーエンコーダを図２－２－７に示す。

表２－２－３　ロータリーエンコーダ仕様

電源電圧	4.5～13[V]
出力パルス数	100[パルス/回転]
応答パルス数	5000[パルス/秒]
消費電流	20[mA]
出力相数	2相

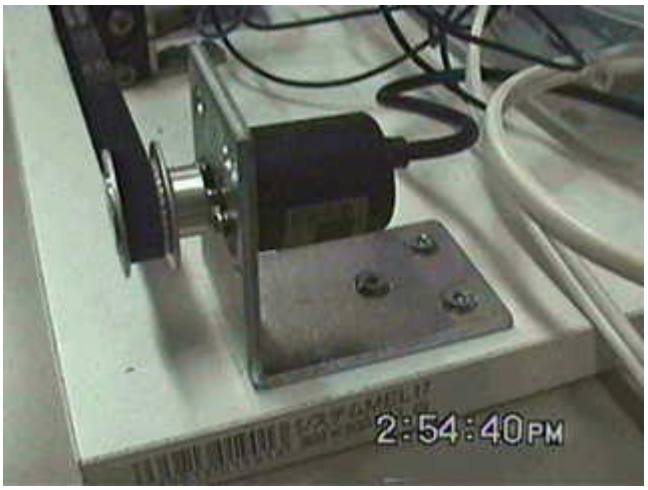


図２－２－７　ロータリーエンコーダ

２．５　マイコン、パソコン間の通信について

マイコンにプログラムの送信、マイコンからの出力の表示をマイコンとパソコンを RS-232C を介して結び、通信することにより行う。これにより、パソコン上の通信プログラムによってマイコンからのデータを読み込み記憶することができ、マイコンからの出力をパソコン上の画面に表示することができる。

３．　CPU まわりハードウェア作成

３．１　出荷時入出力部

出荷時のチャンネル、ポートの割り当てを表２－３－１に示す。汎用入出力のポートの割り当てを表２－３－２に示す。
電力増幅してからビット出力を行う回路を図２－３－１に示す。
ビット入力時のプルアップはソフトウェア内で

P4.PCR.BYTE |= 0x04; /*P4-2 はプルアップ */

として行う。図２－３－２はビット入力部を示し、図２－３－３はビット入力部につなげるリミットスイッチを示す。図２－３－４は実際のリミットスイッチである.

表２－３－１　出荷時の信号の割り当て

入出力	信号名	CPU PIN番号	マザーボードコネクタ
SCI 1-TXD			CN 4－4
SCI1-RXD			CN 4－ 6
押しボタンSW#0	P 4－ 4	23	CN 3－ 3
押しボタンSW#1	P 4－ 5	24	CN 3－ 4
押しボタンSW#2	P 4－ 6	25	CN 3－ 5
押しボタンSW#3	P 4－ 7	26	CN 3－ 6
8ビットSWb0	P 2－ 0	45	CN 3－ 23
8ビットSWb1	P 2－ 1	46	CN 3－ 24
8ビットSWb2	P 2－ 2	47	CN 3－ 25
8ビットSWb3	P 2－ 3	48	CN 3－ 26
8ビットSWb4	P 2－ 4	49	CN 3－ 27
8ビットSWb5	P 2－ 5	50	CN 3－ 28
8ビットSWb6	P 2－ 6	51	CN 3－ 29
8ビットSWb7	P 2－ 7	52	CN 3－ 30
LED＃０	P 5－ 0	53	CN 3－ 31
LED＃１	P 5－ 1	54	CN 3－ 32

- SCI は１チャンネルのみ使用

表２－３－２　汎用入出力の信号の割り当て

入出力	信号名	CPU PIN番号	マザーボードコネクタ
ADC	AN－０	78	CN 2－ 12
DAC	DA－０	84	CN 2－ 18
ビット出力*1	P 5－ 2	55	CN 3－ 33
ビット入力*2	P 4－ 2	20	CN 1－ 33

*1　パワー MOS－FET によって１２V 出力（１A　MAX）になる
*2　GND との ON　OFF 入力　プルアップはソフトウェアによって行う

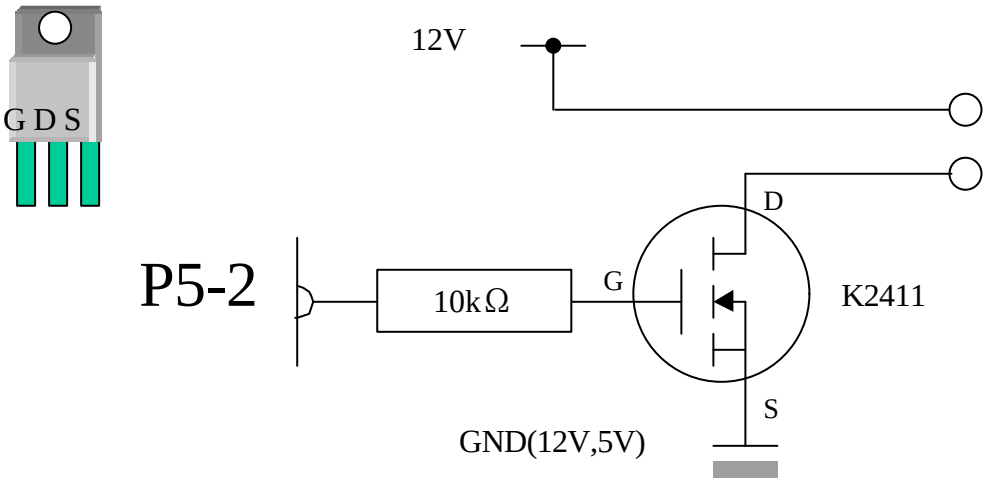


図２－３－１　ビット出力部

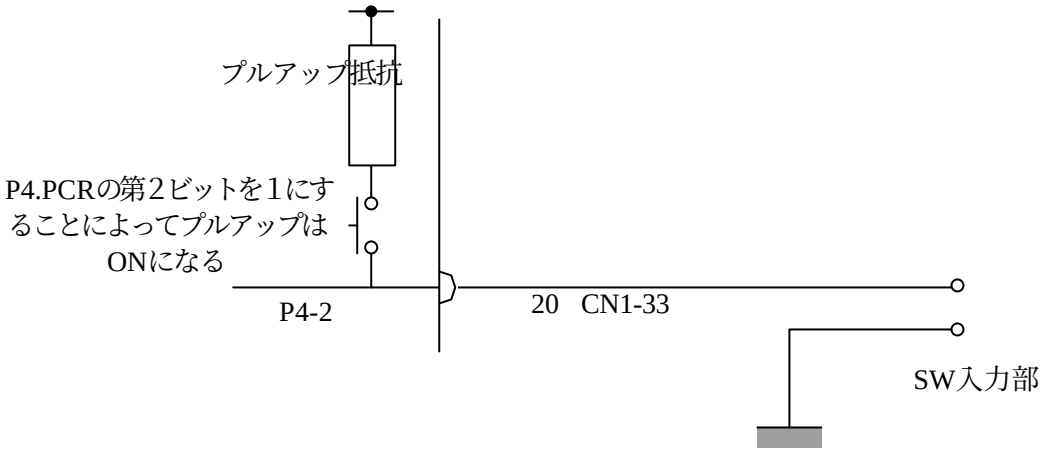


図２－３－２　ビット入力部

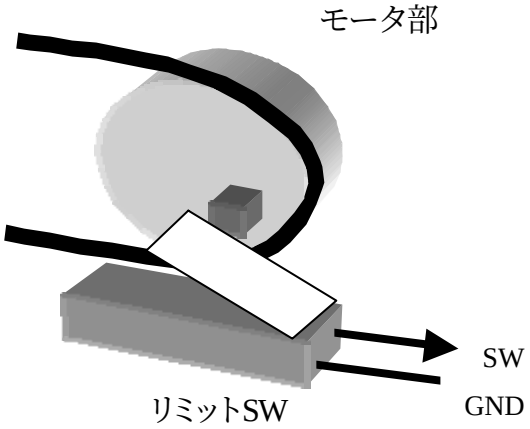


図２－３－３　モータ部に取り付けられたリミットスイッチ

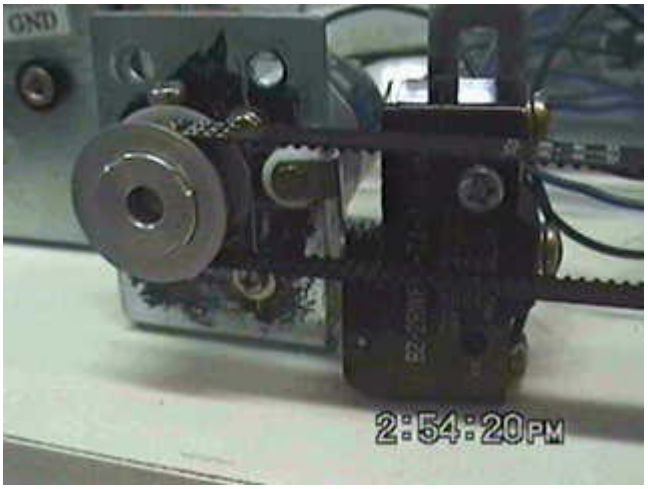


図２－３－４　リミットスイッチ

３．２　ディジタルモータコントロール

PWM **と位相カウント機能をもちいたモータコントロール部のチャンネル、ポートの割り当てを表２－３－３に示す。**GND **と＋５**V **電源の割り当てを表２－３－４に示す。**ITU **のチャンネルの割り当てを表２－３－５に示す。**

表２－３－３　モータコントロール部の信号の割り当て

入出力	信号名	CPU PIN番号	マザーボードコネクタ
PWM	TIOCA３	2	CN１－１６
DIR	Ｐ４－０	18	CN１－３１
BRK	Ｐ４－１	19	CN１－３２
ロータリーエンコーダA相	TCLKA	93	CN１－８
ロータリーエンコーダB相	TCLKB	94	CN１－９

- **PWM 回路に入力される信号**
  PWM（モータに送るデューティ比を調整する信号）
  DIR（モータの回転方向を設定する信号）
  BRK（モータにブレーキをかけるための信号）

表２－３－４　GND と＋５V 電源の信号の割り当て

入出力	信号名	CPU　PIN番号	マザーボードコネクタ
GND			CN４－２
5V			CN４－３

表２－３－５　ITU のチャンネルの割り当て

入出力	ITUチャンネル番号
タイマ割り込み＃０	0
タイマ割り込み＃１	1
ロータリーエンコーダ	2
PWM	3

４．　配線
制御系全体図を図 2-４-0 に示す。

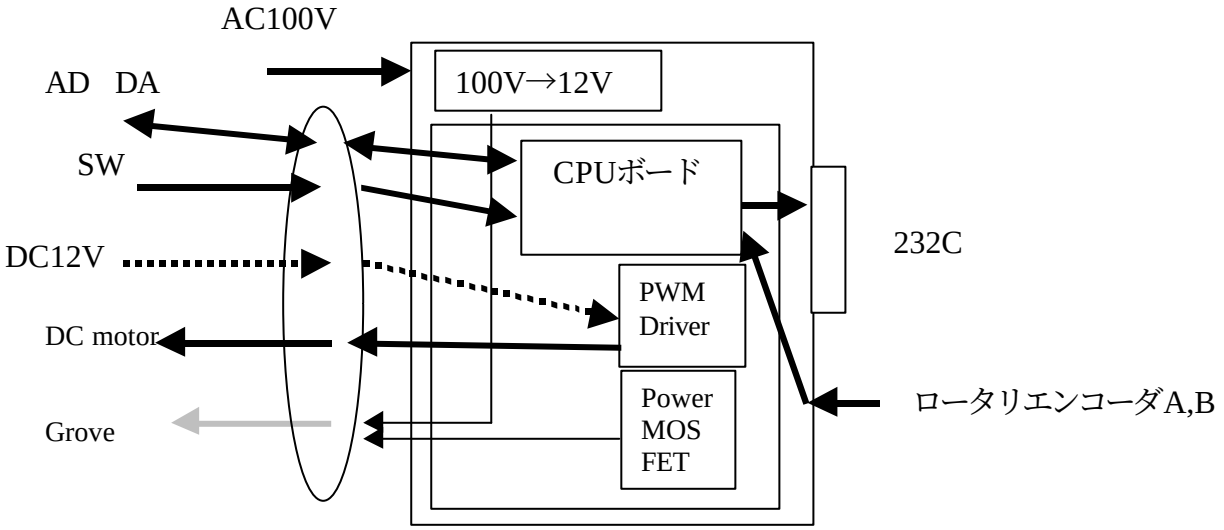


図 2-４-0　制御系全体図

４．１ロータリーエンコーダの配線
ロータリーエンコーダコネクタピン番号について図２－４－１に示し、それに対応する信号を表２－４－１に示す。プルアップは図２－４－２のように行う。

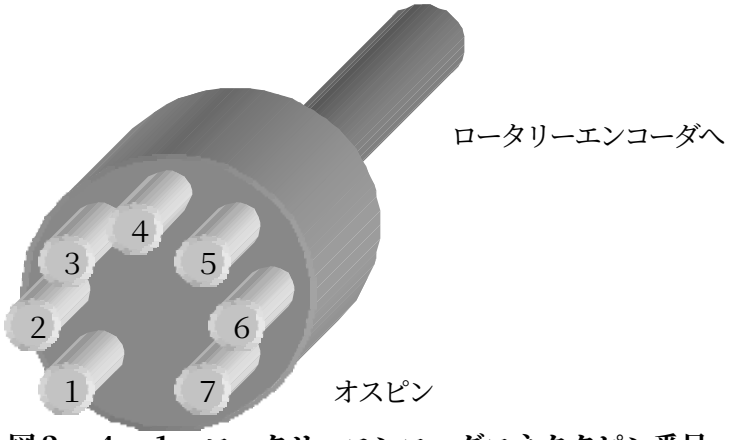


図２－４－１　ロータリーエンコーダコネクタピン番号

表２－４－１　ロータリーエンコーダコネクタピン番号と信号の対応

ピン番号	信号	色
1	A相	白
2	B相	緑
3		
4	GND	黒
5	5V	赤
6		
7		

＊メスピンのピン番号７番は GND がつながっている.

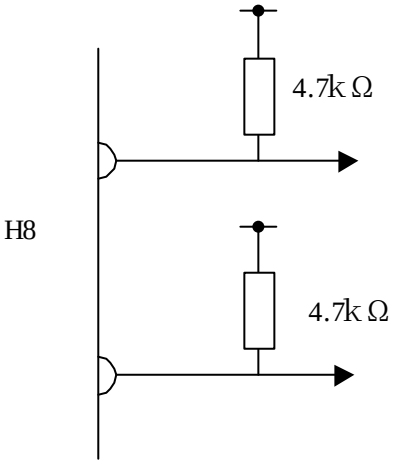


図２－４－２　ロータリーエンコーダの外部抵抗プルアップ

４．２　モータとロータリーエンコーダの位置関係

モータとロータリーエンコーダをベルトでつなぎ固定したものを図２－４－３に示す。

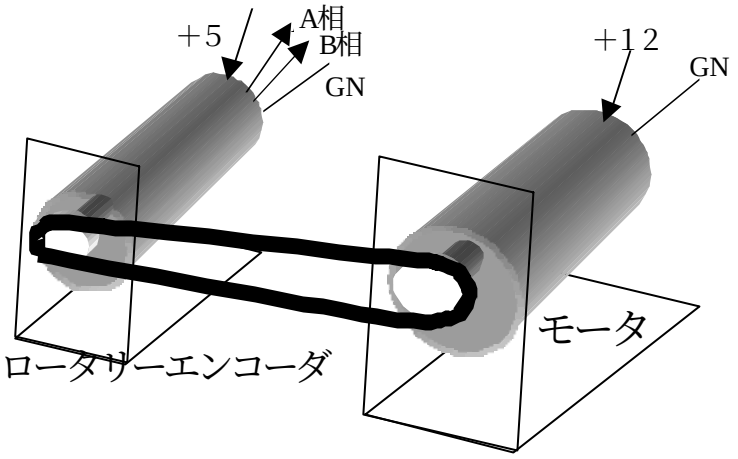


図２－４－３　モータとロータリーエンコーダの位置関係

４．３　供給電源

ハードウェアの電源部を含めた構成を図２－４－４に示す。図２－４－５は実際のシリーズ電源である。

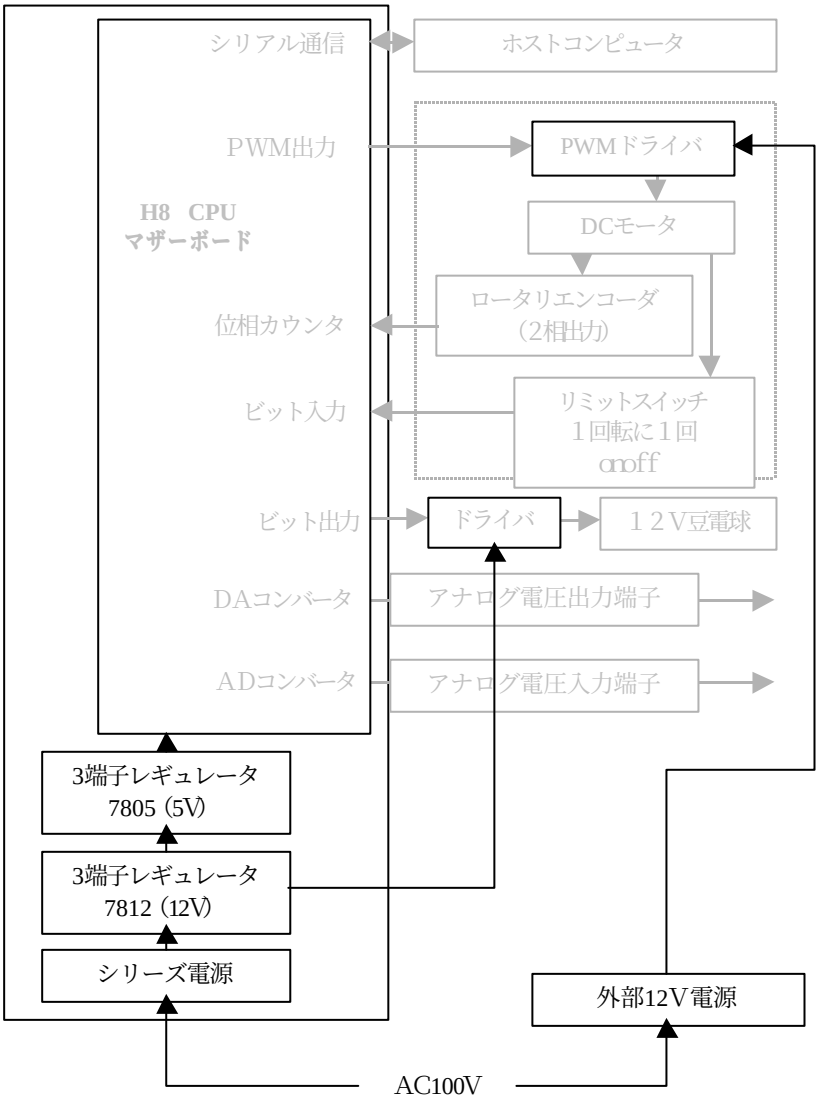


図２－４－４　ハードウェアへの供給電源

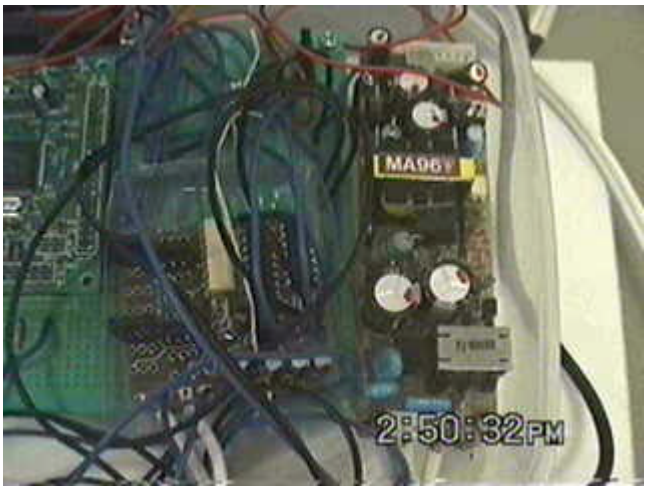


図２－４－５　シリーズ電源

４．４　実際の全体図

実際の配線を終えた時の全体図を図２－４－６に示す。図２－４－７はマザーボードの入ったボックスを上から見たもので、図２－４－８は実際のマザーボードである.

図２－４－６　完成した装置の全体図

図２－４－７　ボックスを上から見た図

図２－４－８　マザーボード

【3】ソフトウェア作成

プログラムの作成にあたり、初期化などの直接 CPU 側にアクセスするものはすべてモジュール関数の中でそれを行う関数を作成し、メイン関数のは CPU 内部を気にしなくてもよくできるようにする.

１.　作成したメインプログラム

（１）H８シングルボードとパソコンをつないだシリアル通信のテスト

ホスト側から何も入力されていないなら@を表示させエラーが起きたら*を表示し、ホスト側から何か入力されたらそれを表示させ、1 バイト分　16 進の１２を表示、１６進数表示、符号付の１６進数の表示、１０進数を表示、符号付の１０進数の表示をしてから、ホスト側から入力された文字のアスキーコードに１を足したものの表示を繰り返すプログラム。

このプログラムは前年度の人が作成したものである。

```
#include "h8-00.h"

main()
{
	int c=0;
	sci1_init();
	do {
		c=sci1_chkandget();
		if (c==-1) sci1_putbyte('@');
		else if (c==-2) sci1_putbyte('*');
		else sci1_putbyte(c);
	} while(c==-1 | | c==-2);
	sci1_rtlf();
	sci1_putString("sci1_putByte(0x12)¥n");
	sci1_putByte(0x12);
	sci1_rtlf();
	sci1_putString("sci1_putWord(0x1234)¥n");
	sci1_putWord(0x1234);
	sci1_rtlf();
	sci1_putString("sci1_putInt(-0x2345)¥n");
	sci1_putInt(-0x2345);
	sci1_rtlf();
	sci1_putString("sci1_putDec(9876)¥n");
	sci1_putDec(9876);
	sci1_rtlf();
	sci1_putString("sci1_putDecInt(19876)¥n");
	sci1_putDecInt(19876);
	sci1_rtlf();
```

```
        sci1_putString("sci1_putDecInt(-19876)¥n");
        sci1_putDecInt(-19876);
        sci1_rtlf();
        while(1) {
                c=sci1_getbyte();
                sci1_putbyte(++c);
        }
}
```

（２）キーボードから与えられた数字をつなげ、整数として返す

ホスト側から文字列を受け取る。「-（マイナス）」と数字だけでできている文字列を受け取ったら１０進数として解釈し、「0ｘ」で始まるか「-0ｘ」で始まる数字と「ABCDEF」からできている文字列を受け取ったら１６進数として解釈し、値を受け取り、それを１６進数と１０進数でホスト側に返すプログラム

```
#include "h8-00.h"

main()
{
        int x;
        sci1_init();
        while(1) {
                x=getint("x=");
                sci1_putInt(x);
                sci1_putbyte(' ');
                sci1_putDecInt(x);
                sci1_rtlf();
        }
}
```

（３）８シングルボード付属の押しボタン SW、８ビット SW による LED のテスト

起動時のマザーボード上の８ビット SW の状態により異なる動作をして LED の ON-OFF テストを行うプログラム。

（ａ）８ビット SW の第０ビット（スイッチには１と表示されている）のみが ON なら両 LED ON にする。

（ｂ）８ビット SW の第１ビット（スイッチには２と表示されている）のみが ON なら両 LED OFF にする。

（ｃ）８ビット SW の第２ビット（スイッチには３と表示されている）のみが ON なら押しボタン SW（マザーボード上の黒い押しボタンスイッチ）により LED の ON OFF をおこなう。

（ｄ）８ビット SW の第３ビット（スイッチには４と表示されている）のみが ON なら、１秒程度で ON OFF のきりかえを継続する。

それ以外は何も起こらないプログラム

```
#include "h8-00.h"

main()
{
        ddr_init();
        led_init();
        pushsw_init();
        eightbitsw_init();
        if (geteightbitsw(0)) {
                led_on(0);
                led_on(1);
                while(1);
```

```
        } else if (geteightbitsw(1)) {
                led_off(0);
                led_off(1);
                while(1);
        } else if (geteightbitsw(2)) {
                while(1) {
                        if (getpushsw(0)) led_on(0);
                        else led_off(0);
                        if (getpushsw(1)) led_on(1);
                        else led_off(1);
                }
        } else if (geteightbitsw(3)) {
                while(1) {
                        int i,j;
                        led_on(0);
                        led_off(1);
                        for(i=0;i<30000;i++)for(j=0;j<30;j++);
                        led_on(1);
                        led_off(0);
                        for(i=0;i<30000;i++)for(j=0;j<30;j++);
                }
        } else {
                while(1);
        }
}
```

（4）LED の ON-OFF を利用した割り込みルーチンのテスト
0.5 秒毎に LED の ON　OFF が切り替わる割り込みの発生するプログラム
図３－１は実行した結果の静止画である。

```
#include "h8-00.h"

extern void E_INT();
int a;

main()
{
        ddr_init();
        led_init();
        timer_init(500);/*0.5sec interrapt*/
        a=0;
        E_INT();
        timer_start();
        while(1);
}

void interrupt_cfunc()
{
        a=1-a;
        if(a) {
                led_on(0);
                led_off(1);
        } else {
                led_on(1);
                led_off(0);
        }
}
```

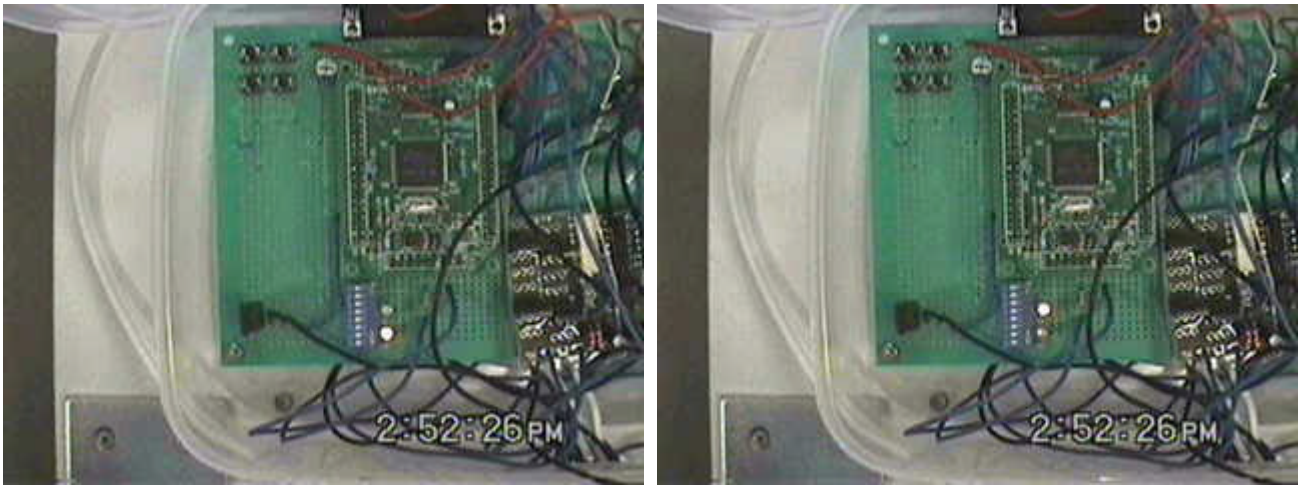


図３－１　LED の ON－OFF 切り替え

（５）DAC 端子の引き出しとテスト

ホスト側から０～２５５までの任意の数を受け取り、その値を DA 変換したものを出力するプログラム

```
#include "h8-00.h"

main()
{
        int x;
        sci1_init();
        da_init();
        while(1) {
                x=getint("DA value(0..255) =");
                da(0,x&0xff);
        }
}

/*
結果
入力値　出力値
    0       0.08V
   64       1.27V
  128       2.53V
  192       3.78V
  255       5.02V
*/
```

（６）ADC 端子の引き出しとテスト

ホスト側から０～２５５までの任意の数を受け取り、その値を DA 変換したものを出力し、その値を ADC に入力しその変換値をホスト側に送り表示させるプログラム

```
#include "h8-00.h"

main()
{
        int x;
        unsigned int y;
        sci1_init();
        da_init();
```

```
        while(1) {
                x=getint("DA value(0..255) =");
                da(0,x&0xff);
                y=ad(0);
                sci1_rtlf();
                sci1_putDecInt((int)y);
                sci1_rtlf();
        }
}

/*
結果
入力値                    出力値
   0      0.08V    →        4
  64      1.27V    →      256
 128      2.53V    →      508
 192      3.78V    →      765
 255      5.02V    →     1017
*/
```

（７）ビット出力端子の引き出しと１２V 出力コントロールのテスト
**　１秒程度毎に１２V を出力させるプログラム**
図３－２は実行した結果の静止画である

```
#include "h8-00.h"
#include "led12_p5b2.h"

main()
{
        int x;
        ddr_init();
        led12_init();
        while(1) {
                int i,j;
                led12_on();
                for(i=0;i<30000;i++)for(j=0;j<30;j++);
                led12_off();
                for(i=0;i<30000;i++)for(j=0;j<30;j++);
        }
}
```

図３－２　１２V 出力の ON－OFF 切り替え

（8）ビット入力端子の引き出しと SW のテスト

ビット入力端子についている SW の状態を読み出し、押されているなら１、そうでないなら０をホスト側に送り表示させるプログラム

```
#include "h8-00.h"
#include "rsw_p4b2.h"

main()
{
        int x;
        ddr_init();
        sci1_init();            /*シリアルポート初期化*/
        rsw_init();
        while(1) {
                x=rsw();
                sci1_putDecInt(x);
                sci1_rtlf();
        }
}
```

（9）PWM 端子の引き出しとモータのテスト

ホスト側から－４０９６から４０９６までの値を受け取り、その値に応じたスピードでモータを回し、それ以外の数字であるならブレーキをかけるプログラム、なお符号はモータの回転方向を決めている。
正の値を PWM 値として出力すると、モータは正面から見て反時計回りに回る。
負の値を PWM 値として出力すると、モータは正面から見て時計回りに回る。

```
#include "h8-00.h"
#include "pwm3cntl.h"

main()
{
        int x;
        ddr_init();
        sci1_init();            /*シリアルポート初期化*/
        initPWM3(4096);         /*PWM 出力初期化（ITUchanel3 使用）*/
        while(1) {
                x=getint("PWM value(-4096..4096 otherwize break:on) =");
                sci1_putString("¥n");
                sci1_putDecInt(x);
                sci1_rtlf();
                if (-4096<=x&&x<=4096) outPWM3(x);
                else outPWM3_break(1);
        }
}
```

（10）位相カウント端子の引き出しとモータに取り付けたロータリーエンコーダのテスト

ロータリーエンコーダの値を読み出してホスト側に送り表示させ、ホスト側から何か入力されたらカウンタの値をリセットさせるプログラム
なおロータリーエンコーダは１周に付き４００パルスをカウントしている。（ロータリーエンコーダは１周につき１００パルスを発生しているが４逓倍されて１周あたり４００パルスにカウントされる。）

```
#include "h8-00.h"

main()
{
```

```
        int x;
        sci1_init();            /*シリアルポート初期化*/
        h8counter_init();       /*H8 位相内部カウンタ初期化（ITUchanel2 使用）*/
        while(1) {
                x=geth8counter();     /*位相カウンタの読み出し*/
                sci1_putDecInt(x);    /*シリアルポートへ送信*/
                sci1_rtlf();          /*シリアルポートへ改行送信*/
                if (0<sci1_chkandget()) h8counter_init();
                      /*シリアルポートから入力があったら、位相カウンタクリア*/
        }
}
```

（11）モータをモータに取り付けた SW の信号をもとに制御する

モータを PWM 出力値=2000 で回転させ、モータの回転軸に取り付けたねじが SW にあたり、SW が ON になったら１秒程度止まる。ここを初期位置とする。そこから PWM 出力値=2000 で回転させ、３回 SW が ON になるまで進み、また１秒程度とまり、今度は逆回転（PWM 出力値=-2000）して SW が３回 ON になるまで回転し、また１秒程度止まる。「初期位置から３回転進み３回逆転し初期位置に戻る」を繰り返すプログラム

```
#include "h8-00.h"
#include "pwm3cntl.h"
#include "rsw_p4b2.h"

int average_rsw(void)
/*1:安定した ON 0:安定した OFF -1:不安定*/
{
        int stat,i;
        stat=rsw();
        for (i=0;i<100;i++) {
                if (stat!=rsw()) return -1;
        }
        return stat;
}

void go(int x)
{
        int i,j;
        outPWM3(x);
        while(average_rsw()!=0);
        while(average_rsw()!=1);
        while(average_rsw()!=0);
        while(average_rsw()!=1);
        while(average_rsw()!=0);
        while(average_rsw()!=1);
        outPWM3_break(1);
        for(i=0;i<30000;i++) for(j=0;j<30;j++);
}

main()
{
        int x,s,a=0,i,j;
        ddr_init();
        sci1_init();            /*シリアルポート初期化*/
        rsw_init();
        initPWM3(4096);         /*PWM 出力初期化（ITUchanel3 使用）*/
```

```
        x=2000;
        outPWM3(x);
        s=rsw();
        while(s!=1){
                s=rsw();
        }
        outPWM3_break(1);
        for(i=0;i<30000;i++) for(j=0;j<50;j++);
        while(1){
                go(x);
                go(-x);
        }
}
```

（12）**モータをロータリーエンコーダの信号をもとに制御する**

モータを回転させ、ベルトでつながれたロータリーエンコーダの値が１０００調度の位置でとまらせ、１秒程度とまり、今度は逆回転をし、ロータリーエンコーダの値が０の位置で止まらせる。これを繰り返すプログラム。

本プログラムでは、モータを PWM **出力値**=4095 **で回し、ロータリーエンコーダの値が８００になったら速度を１/８にし、９８**0 **を越えるとブレーキして１０００調度で止まるようにし、逆回転で同じような動作を繰り返しこんどは０でとまるようにする。一連のどうさは、０．０１秒毎の割り込みによって行われる。**

図３－３は実行結果をグラフ化したものである。

```
#include "h8-00.h"
#include "pwm3cntl.h"
#include "rsw_p4b2.h"

extern void E_INT();

volatile int x,moku,g;
volatile unsigned long int count,count0;
volatile int status; /* 0:stop,1:forward -1:reverse */
                     /* 2:stop & next status=1 */
                     /* -2:stop & next status=-1 */

main()
{
        ddr_init();
        sci1_init();            /*シリアルポート初期化*/
        h8counter_init();     /*H8 位相内部カウンタ初期化（ITUchanel2 使用）*/
        initPWM3(4096);         /*PWM 出力初期化（ITUchanel3 使用）*/
        timer_init(10);/*0.01sec interrapt*/

        count=0;
        g=0;
        x=4095;
        moku=1000;
        status=1;
        E_INT();
        timer_start();
        while(1);
}

void interrupt_cfunc()
{
        int c,x1=0;
```

```
	c=geth8counter();

	if(status==1){
		if(c<moku/10*8){
			x1=x;
			outPWM3(x1);
		}
		else if(moku/10*8<=c&&c<moku-20){
			x1=x/8;
			outPWM3(x1);
		}
		else{
			status=-2;
			outPWM3_break(1);
			count0=count;
		}
	}
	else if(status==-1){
		if(moku/10*2<c){
			x1=-x;
			outPWM3(x1);
		}
		else if(0+20<c&&c<=moku/10*2){
			x1=-x/8;
			outPWM3(x1);
		}
		else{
			status=2;
			outPWM3_break(1);
			count0=count;
		}

	}
	else {
		if(count==count0+500) status=status/2;
	}

	sci1_putDecInt((int)count);
	  /*タイムカウンタ値をシリアルポートへ送信*/
	sci1_putbyte(' ');
	sci1_putDecInt(c);   /*角度値をシリアルポートへ送信*/
	sci1_putbyte(' ');
	sci1_putDecInt(x1);   /*モータ出力値をシリアルポートへ送信*/
	sci1_rtlf();            /*シリアルポートへ改行送信*/
	count++;
}
```

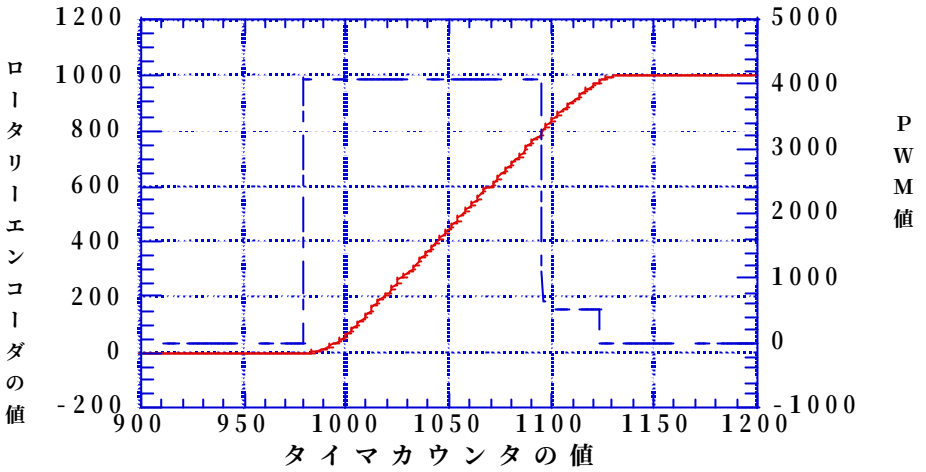


図３－３　時間による PWM 値と
ロータリーエンコーダの値の変化

２．モジュール関数

H8CPU では IO はメモリマップド空間で行われている。具体的な IO アドレスはインクルードファイル 3048f.h によってユーザから隠され、アドレスに対応する IO 名で操作できるようになっている。

例えば

IO 名 P5 は

```
#define P5      (*(volatile struct st_p5    *)0xFFFFC8) /* P5    Address*/
```

のように定義されており、P5 のアドレスは 0xFFFFC8 になっている。（実は P5.DDR のアドレス）

次にユーザプログラムでは毎回 IO 名でプログラミングしたり、ある動作をさせるのにある IO 名のアドレスにどのような値を設定しなければならないかを調べる必要がないように、モジュール関数を作成し、細部を隠している。

以下にモジュール関数定義ファイルを示す。関数定義ファイルはインクルードファイルとなっており、現在のところ毎回コンパイルされる。

ここに示すファイルは

（１）h8-00.h　　シリアル通信、8 ビットスイッチ、押しボタンスイッチ、時間割り込み、位相計測（ロータリエンコーダ信号計測）

（２）pwm3cntl.h　　ITU3 を用いた PWM 信号発生

（３）rsw_p4b2.h　　外部スイッチ

（４）led12_p5b2.h　　外部出力

である。

（１）h8-00.h

```
#include<3048f.h>

int P1DDR,P2DDR,P3DDR,P4DDR,P5DDR,P6DDR,P7DDR,P8DDR,P9DDR,PADDR,PBDDR;

unsigned char dummydummy;

extern void E_INT();

void ddr_init(void)
/*どんなプログラムも呼び出さなければならない*/
{
        P1DDR=P2DDR=P3DDR=P4DDR=P5DDR=P6DDR=P7DDR=P8DDR=P9DDR=PADDR=PB
DDR=0;
}

/* ------------------------------------------- */
```

```
/* SCI1 INITIALIZATION fixed baud at 38400          */
/* ------------------------------------------------ */
void sci1_init(){
        int i;
        unsigned char dummy;
        SCI1.SCR.BYTE = 0;      /* clear all flags */
                               /* 2400-38400baud are available at n=0(cks1=0,cks2=0) */
        SCI1.SMR.BYTE = 0;      /* Ascnc, 8bit , NoParity, stop1, 1/1 */
        /* SCI1.BRR = 15;          31250baud (CPU=16MHz) */
        /* SCI1.BRR = 25;          19200baud (CPU=16MHz) */
        /* SCI1.BRR = 51;          9600baud  (CPU=16MHz) */
        SCI1.BRR = 12;          /* 38400baud (CPU=16MHz) */
        for(i=0;i<30000;i++);   /* wait more than 1bit time */
        SCI1.SCR.BYTE = 0x30;   /* scr = 0011 0000 (TE=1,RE=1) */
        /*dummy = SCI1.SSR.BYTE;*/  /* read dummy              ????*/
        /*SCI1.SSR.BYTE = 0x80;*/     /* clear error flag (TDRE = 1)     ????*/
        return;
}

/* ------------------------------------------------ */
/* GET BYTE FROM SCI1 */
/* ------------------------------------------------ */
int sci1_getbyte()
/* return value 0x00-0xFF:received data */
/*              -2(0xFFFE):error  */
{
        int flags,recdata;
        do {
                flags = SCI1.SSR.BYTE;
                if (flags&0x38) {/* error */
                        SCI1.SSR.BIT.RDRF = 0;
                        SCI1.SSR.BIT.ORER = 0;
                        SCI1.SSR.BIT.FER = 0;
                        SCI1.SSR.BIT.PER = 0;
                        return -2;
                }
                if (flags&0x40) {/* normally received one data */
                        SCI1.SSR.BIT.RDRF = 0;
                        recdata=SCI1.RDR;
                        return recdata;
                }
        } while (1);
}

/* ------------------------------------------------ */
/* CHECK SCI BUFFER AND GET DATA */
/* ------------------------------------------------ */
int sci1_chkandget()
/* return value -1(0xFFFF):no received data  */
/*              0x00-0xFF:received data  */
/*              -2(0xFFFE):error  */
{
        int flags,recdata;
        flags = SCI1.SSR.BYTE;
        SCI1.SSR.BIT.RDRF = 0;
        if (flags&0x38) {/* error */
                SCI1.SSR.BIT.ORER = 0;
                SCI1.SSR.BIT.FER = 0;
```

```
                SCI1.SSR.BIT.PER = 0;
                return -2;
        }
        if (flags&0x40) {/* normally received one data */
                recdata=SCI1.RDR;
                return recdata;
        } else {
                return -1;
        }
}

/* ------------------------------------------------ */
/* GET INTEGER FROM SCI1 */
/* ------------------------------------------------ */
int getint(char prompt[])
/*getting integer from serial port*/
/* format 123[ret] */
/*          -123[ret]  */
/*          0x1a[ret]  */
/*          -0x100[ret]  */
{
        int x=0,y,m=0,n=0,v=0;
        sci1_putString(prompt);
        y=sci1_getbyte();
        while(y!='¥r'){
                if(y=='-') m=1;
                if('a'<=y&&y<='z') y=y-'a'+'A';
                if(y=='0') n=1;

                if(v==1){
                        if('0'<=y&&y<='9'){
                                y=y-'0';
                        }
                        else if('A'<=y&&y<='F'){
                                y=y-'A'+10;
                        }
                        x=16*x+y;
                }

                if(n==1&&y=='X'){
                        v=1;
                }

                if(v==0&&'0'<=y&&y<='9'){
                        y=y-'0';
                        x=10*x+y;
                }

                y=sci1_getbyte();
        }
        if(m==1) x=-x;
        return x;
}

/* ------------------------------------------------ */
/* SEND BYTE TO SCI1 */
/* ------------------------------------------------ */
void sci1_putbyte(char c)
```

```
{
	unsigned char tmp;
	do{
		tmp = SCI1.SSR.BYTE;
	} while((tmp & 0x80)==0);
	SCI1.TDR = c;
	SCI1.SSR.BIT.TDRE = 0;
	return;
}

/* ------------------------------------------------ */
/* SCI1 CARRIGE RETURN LINE FEED*/
/* ------------------------------------------------ */
void sci1_rtlf()
{
	sci1_putbyte(0x0d);
	sci1_putbyte(0x0a);
	return;
}

/* ------------------------------------------------ */
/* SCI1 CARRIGE RETURN */
/* ------------------------------------------------ */
void sci1_rt()
{
	sci1_putbyte(0x0d);
	return;
}

static unsigned char hex[] = "0123456789ABCDEF";
/* ------------------------------------------------ */
/* SEND 1 BYTE TO SCI1 */
/* ------------------------------------------------ */
void sci1_putByte(unsigned char c)
{
	sci1_putbyte(hex[(c>>4)&0x0f]);
	sci1_putbyte(hex[c&0x0f]);
}

/* ------------------------------------------------ */
/* SEND WORD(2 BYTE) TO SCI1 */
/* ------------------------------------------------ */
void sci1_putWord(unsigned short int c)
{
	sci1_putByte(c>>8);
	sci1_putByte(c);
}

/* ------------------------------------------------ */
/* SEND INTEGER  TO SCI1 */
/* ------------------------------------------------ */
void sci1_putInt(int i){
	if(i<0){
		sci1_putbyte('-');
		i = -i;
	}
	else{
		sci1_putbyte(' ');
```

```
	}
	sci1_putWord((unsigned short int)i);
}

/* ---------------------------------------------- */
/* SEND DECIMAL INTEGER TO SCI1 */
/* ---------------------------------------------- */
void sci1_putDec(unsigned int i){
	int d;
	for(d=10000;d!=0;d/=10){
		sci1_putbyte((i/d)+'0');
		i=i%d;
	}
}

/* ---------------------------------------------- */
/* SEND DECIMAL INTEGER TO SCI1 */
/* ---------------------------------------------- */
void sci1_putDecInt(int i){
	if(i<0){
		sci1_putbyte('-');
		i = -i;
	}
	else{
		sci1_putbyte(' ');
	}
	sci1_putDec((unsigned short int)i);
}

/* ---------------------------------------------- */
/* SEND STRING TO SCI1 */
/* ---------------------------------------------- */
void sci1_putString(char *str)
{
	while(*str){
		sci1_putbyte(*str);
		str++;
	}
}

/* ---------------------------------------------- */
/* DA INITIALIZATION */
/* ---------------------------------------------- */
void da_init(){
	DA.CR.BYTE |= 0x40;
}

/* ---------------------------------------------- */
/* DA CONVERT */
/* ---------------------------------------------- */
void da(unsigned char ch, unsigned char data){
	if(ch==0)
		DA.DR0 = data;
	else if(ch==1)
		DA.DR1 = data;
}

/* ---------------------------------------------- */
```

```
/* AD CONVERT */
/* ---------------------------------------------- */
unsigned int ad(unsigned char ch){
        unsigned int tmp;

        AD.CSR.BYTE = ch;       /*  ＡＤ設定単一モード、ＡＮ(ch)のみ使用*/
        AD.CSR.BIT.ADST = 1;  /* start */
        while(AD.CSR.BIT.ADF == 0); /* AD 変換終了待機  */
        AD.CSR.BIT.ADF = 0;

        tmp = (int)(*((volatile unsigned char *)(&AD.DRA)+ch*2  )<<2)
            + (int)(*((volatile unsigned char *)(&AD.DRA)+ch*2+1)>>6);
        return tmp;
}

/* ---------------------------------------------- */
/* LED INITIALIZATION */
/* ---------------------------------------------- */
/*****************************************************************
LED 0:P5-0
LED 1:P5-1
*****************************************************************/
void led_init(){
        P5DDR  |=   0x3;
        P5.DDR  |=   P5DDR;
        /*P5.DDR  |=   0x3;*/
}

/* ---------------------------------------------- */
/* LET LED ON */
/* ---------------------------------------------- */
void led_on(int rl){
        if(rl==0){
                P5.DR.BYTE  |= 1; /* right on */
        }
        if(rl==1){
                P5.DR.BYTE  |= 2; /* left   on */
        }
}

/* ---------------------------------------------- */
/* LET LED OFF */
/* ---------------------------------------------- */
void led_off(int rl){
        if(rl==0){
                P5.DR.BYTE &= 0xfe; /* right off */
        }
        if(rl==1){
                P5.DR.BYTE &= 0xfd; /* left   off */
        }
}

/* ---------------------------------------------- */
/* PUSH SW INITIALIZATION */
/* ---------------------------------------------- */
/*****************************************************************
```
押しボタンスイッチ S0：P4-4
押しボタンスイッチ S1：P4-5

```
押しボタンスイッチ S2：P4-6
押しボタンスイッチ S3：P4-7
************************************************************/
void pushsw_init(void)
{
        P4DDR &= 0xf; /*P4-4,5,6,7 は入力*/
        P4.DDR=P4DDR;
        P4.PCR.BIT.B4 = 1; /*P4-4 はプルアップ  */
        P4.PCR.BIT.B5 = 1; /*P4-5 はプルアップ  */
        P4.PCR.BIT.B6 = 1; /*P4-6 はプルアップ  */
        P4.PCR.BIT.B7 = 1; /*P4-7 はプルアップ  */
}

/* ----------------------------------------------- */
/* GET PUSH SW  */
/* ----------------------------------------------- */
int getpushsw(int number)
/*push sw 0,1,2,3 の状態を調べる    number:0,1,2,or 3*/
/*押されていたら 1、そうでなかったら 0 を返す*/
/*number が 0,1,2,3 以外だったら-1 を返す*/
{
        int tmp,ret=1;
        tmp=P4.DR.BYTE;
        switch (number) {
                case 0:
                        if (tmp&0x10) ret=0;
                        break;
                case 1:
                        if (tmp&0x20) ret=0;
                        break;
                case 2:
                        if (tmp&0x40) ret=0;
                        break;
                case 3:
                        if (tmp&0x80) ret=0;
                        break;
                default:
                        ret=-1;
                        break;
        }
        return ret;
}

/* ----------------------------------------------- */
/* PUSH 8 BIT SW INITIALIZATION */
/* ----------------------------------------------- */
void eightbitsw_init(void)
{
        P2.DDR = 0x00;/*8bitSW のポートを入力に設定*/
        P2.PCR.BYTE = 0xff;/*8bitSW のプルアップ設定*/
}

int geteightbitsw0(void)
/*8bitsw の状態をそのまま返す*/
{
        return P2.DR.BYTE;
}
```

```
int geteightbitsw(int number)
/*8bitsw 0,1,2,3,4,5,6,7 の状態を調べる　number:0,1,2,3,4,5,6,or 7*/
/*ON なら 1、そうでなかったら 0 を返す*/
/*number が 0,1,2,3,4,5,6,7 以外だったら-1 を返す*/
{
        int tmp,ret=1;
        tmp=P2.DR.BYTE;
        switch (number) {
                case 0:
                        if (tmp&0x1) ret=0;
                        break;
                case 1:
                        if (tmp&0x2) ret=0;
                        break;
                case 2:
                        if (tmp&0x4) ret=0;
                        break;
                case 3:
                        if (tmp&0x8) ret=0;
                        break;
                case 4:
                        if (tmp&0x10) ret=0;
                        break;
                case 5:
                        if (tmp&0x20) ret=0;
                        break;
                case 6:
                        if (tmp&0x40) ret=0;
                        break;
                case 7:
                        if (tmp&0x80) ret=0;
                        break;
                default:
                        ret=-1;
                        break;
        }
        return ret;
}

/* ------------------------------------------------ */
/* TIMER INITIALIZATION */
/* ------------------------------------------------ */
void timer_init(unsigned int period)/*period [msec]*/
{
        /*ITU.TMDR.BYTE = 0;*/       /* タイマー初期設定 通常動作 */ /*他との競合のため削除*/
        ITU0.TCR.BYTE = 0xc3;   /* タイマー　CH0 GRB でクリア、内部 1/8 ｸﾛｯｸ */
        ITU0.TIOR.BYTE = 0xb8;  /* タイマー　CH0 GRB のコンペアマッチでトグル出力 */
        ITU0.GRB = 2000-1;          /* タイマー　CH0 ((16/8)MHz)/1000Hz=2000 */

        ITU1.TCR.BYTE = 0xb7;   /* タイマー　CH1 GRA でクリア、TCLKD をカウントする */
        ITU1.TIOR.BYTE = 0x8a;  /* タイマー　CH1 GRA のコンペアマッチ出力 */
        ITU1.GRA = period-1;       /* タイマー　CH1 1000 = 1SEC */
        ITU1.TIER.BYTE = 0xf9;  /* タイマー　CH1 GRA で割り込みする */
}

void timer_init3(unsigned int period)/*period [   sec]*/
```

```
{
        /*ITU.TMDR.BYTE = 0;*/       /* タイマー初期設定 通常動作 */ /*他との競合のため削除*/
        ITU0.TCR.BYTE = 0xc3;   /* タイマー   CH0 GRB でクリア、内部 1/8 ｸﾛｯｸ */
        ITU0.TIOR.BYTE = 0xb8;  /* タイマー   CH0 GRB のコンペアマッチでトグル出力 */
        ITU0.GRB = 2-1;          /* タイマー   CH0 ((16/8)MHz)/1000kHz=2 */

        ITU1.TCR.BYTE = 0xb7;   /* タイマー   CH1 GRA でクリア、TCLKD をカウントする */
        ITU1.TIOR.BYTE = 0x8a;  /* タイマー   CH1 GRA のコンペアマッチ出力 */
        ITU1.GRA = period-1;       /* タイマー   CH1 1000 = 1mSEC */
        ITU1.TIER.BYTE = 0xf9;  /* タイマー   CH1 GRA で割り込みする */
}

/* ---------------------------------------------- */
/* PORT 1 INITIALIZATION */
/* ---------------------------------------------- */
void p1_init(){
        P1.DDR |=  0xff;
}

/* ---------------------------------------------- */
/* TIMER START */
/* ---------------------------------------------- */
void timer_start(){
        ITU.TSTR.BYTE |= 0x03;          /* チャンネル 0 チャンネル 1 同時スタート */
}

/* ---------------------------------------------- */
/* H8 INNER COUNTER INITIALIZE AND CLEAR*/
/* ---------------------------------------------- */
void h8counter_init(void)
/*ITU chanel2 を使用*/
{
        ITU.TMDR.BYTE |= 0x40; /*MDF->1 ITU2:counter mode*/
        ITU.TSTR.BIT.STR2=1;
        ITU2.TCNT=0;
}

/* ---------------------------------------------- */
/* READ H8 INNER COUNTER*/
/* ---------------------------------------------- */
int geth8counter(void)
{
        return ITU2.TCNT;
}
```

（2）pwm3cntl.h

```
/*
ITU Ch3 を用いた PWM 信号発生
TIOCA3(PWM 波形出力信号) は PB-0(PortB-bit0) Conecter CH1-16
P4-bit0 正転逆転信号 モータ側に進行は逆転方向
P4-bit1 ブレーキ
*/

void initPWM3(int gra_value)
/*ITU3 channel*/
```

```
{
        P4DDR |= 0x03;          /*P4 の第 0bit,第 1bit を出力に設定*/
        P4.DDR = P4DDR;

        ITU3.TCR.BYTE = 0xA3;
        /*タイマーコントロールレジスタの設定を 10100011 にすることにより内部カウンタφ/8、
        立ち上がりエッヂでカウント。GRA のコンペアマッチでタイマカウンタをクリアする。*/

        ITU3.GRA = gra_value;   /*GRA の設定*/
        ITU3.GRB = 0;   /*GRB の設定*/
        /*GRB/GRA がデューティー比*/
        /*φ＝16MHz(CPU clock)   φ/8=2MHz*/
        /*GRA（0x1000=4096）のときφ1＝2MHz なので一周期は 1/488.28125 秒
                                                （パルス周波数 488.28125Hz）*/
        /*GRA（0x800=2048）のときφ1＝2MHz なので一周期は 1/976.5625 秒
                                                （パルス周波数 976.5625Hz）*/

        ITU.TMDR.BIT.PWM3 = 1;
        /*1 にすることによりチャンネル 3 は PWM モードとなり,
        TIOCA3 端子(ポート B の bit0)は PWM 出力になる*/

        ITU.TSTR.BIT.STR3 = 1;
        /*タイマスタートレジスタのチャンネル 3 のタイマカウンタを
        アップカウンタでスタートさせる*/
}

void outPWM3(int value)
/* value : -gra_value...gra_value */
/*ITU3 channel*/
{
        P4.DR.BIT.B1 =1; /*ブレーキ off*/

        if (0<value) {
                P4.DR.BIT.B0 = 1;
                ITU3.GRB = value;
        } else {
                P4.DR.BIT.B0 = 0;
                ITU3.GRB = -value;
        }
}

void outPWM3_break(int onoff)
/*1:ON   0:OFF*/
/*ブレーキ信号は負論理*/
{
        if (onoff) {
                P4.DR.BIT.B1 =0; /*ブレーキ on*/
                ITU3.GRB = 0;
        } else {
                P4.DR.BIT.B1 =1; /*ブレーキ off*/
        }
}
```

（3）rsw_p4b2.h

```
/*#include<3048f.h>*/

void rsw_init(void)
{
        P4DDR &= 0xfb; /*P4-2 は入力*/
        P4.DDR = P4DDR;
        P4.PCR.BYTE |=  0x04; /*P4-2 はプルアップ */
}

int rsw(void)
/*リミット sw の状態を調べる*/
/*押されていたら 1、そうでなかったら 0 を返す*/
{
        int tmp,ret=0;
        tmp=P4.DR.BYTE;
        if (tmp&0x04) ret=1;
        return ret;
}
```

（4）led12_p5b2.h

```
/************************************************************
12V_LED :P5-2
************************************************************/
void led12_init()
{
        P5DDR |=  0x4;
        P5.DDR=P5DDR;
;
}

/* ------------------------------------------------ */
/* LET LED ON */
/* ------------------------------------------------ */
void led12_on(void)
{
        P5.DR.BYTE |= 4; /* on */
}

/* ------------------------------------------------ */
/* LET LED OFF */
/* ------------------------------------------------ */
void led12_off(void)
{
        P5.DR.BYTE &= 0xfb; /* off */
}
```

ファイル H8-00.h **は笠井健太氏、長島隆氏、谷山暁氏（以上９８年度小坂研究室研究生）、**高沢直也**氏、山田雅也氏（以上９９年度小坂吉本研究室研究生）が制作し、更新してきたモジュール群であり、本実習においても更新している。**

【4】結論

以下のプログラムを作成、実行することによって本実習で作られ H8 シングルボードコンピュータ（AKI-H8）の機能評価をすることができた。

（1） H８シングルボードとパソコンをつないだシリアル通信のテスト
（2） キーボードから与えられた値をつなげ、整数として返す
（3） H８シングルボード付属の押しボタン SW、８ビット SW による LED のテスト
（4） LED の ON-OFF を利用した割り込みルーチンのテスト
（5） AC 端子の引き出しとテスト
（6） ADC 端子の引き出しとテスト
（7） ビット出力端子の引き出しと１２V 出力コントロールのテスト
（8） ビット入力端子の引き出しと SW のテスト
（9） PWM 端子の引き出しとモータのテスト
（11）位相カウント端子の引き出しとモータに取り付けたロータリーエンコーダのテスト
（12）モータをモータに取り付けた SW により制御する
（13）モータをロータリーエンコーダにより制御する

【5】感想

ハードウェアの作成にあたり、ＣＰＵを使ってモータなどのハードを動かしたり、ＳＷなどの入力を得たりする仕組みが理解できた。また、ソフトウェアの作成により、ソフトとハードの関連性が理解でき、とても勉強になった。

付録：簡単なプログラムの作成から実行まで
例　LED を点灯させる。

シングルボードマイコンを動作させるまでの流れ

シングルボードマイコン内で動作するプログラムはパソコン上で開発され、シングルボードマイコンにダウンロードされる。ソースプログラムは C 言語で記述され、クロスコンパイラにより、オブジェクトプログラムに変換され、最後はダウンロード形式（XXXX.mot）になる。シングルボードマイコンを動作させるまでの大きな流れは以下のようになる。

（1）準備（作業用のフォルダの用意）
（2）ソースプログラムの作成
（3） ダウンロード形式ファイルの作成（クロスコンパイル、コード変換）
（4）ROM ライターへの書き込み操作

これら三つの作業によりシングルボードマイコン内で動作させることが出来る。

1．準備

作業用のフォルダを用意する。その中には以下のファイルを用意しておく。

（1）flash.exe
（2）3048.inf
（3）addpath.bat
（4）h8cc.bat

これ以降の作業はすべてこのフォルダ内で行われる。

2．プログラムの作成

専用の C コンパイラを用いることで C 言語によるプログラミングが可能である。エディタ上で、C 言語で記述されているソースプログラムを作成する。

ＬＥＤの点灯プログラム　LED.c

```
#include <3048f.h>

main()
{
        P5.DDR =  0x3; /*LED INITIALIZATION*/
        P5.DR.BYTE |= 1; /* right on */
        P5.DR.BYTE |= 2; /* left  on */
}
```

3．ダウンロード形式ファイルの作成

C ソースファイルが出来ている前提のもとで、ダウンロード形式ファイル(xxxx.mot)の作成方法について以下に述べる。ダウンロード形式ファイルを作るためには、ソースファイル(XXXX.c)、サブファイル(XXXX.sub)が必要である。

（1）サブファイルの作成

サブファイルはリンク時に必要なもので、次のような書式になる。

```
------------------------------------------------- head of list ----------------------------------------------------
  OUTPUT パス¥ソースファイル名（拡張子なし）
  PRINT パス¥ソースファイル名（拡張子なし）
  INPUT パス¥myresetv,パス¥ソースファイル名（拡張子なし）
  LIB パス¥c38hab
  START P,C,D(200),B(0FEF10)
  EXIT
------------------------------------------------- end of list -----------------------------------------------------
```

C ソースファイルが LED.c の場合

```
------------------------------------------------- head of list ----------------------------------------------------
  OUTPUT LED
  PRINT LED
  INPUT c:¥Progra~1¥h8¥myresetv, LED
```

```
LIB c: ¥Progra~1¥h8¥c¥c38hab
START P,C,D(200),B(0FEF10)
EXIT
```
---------- end of list ----------

のように書く。ファイル名は「LED.sub」になる。
　サブファイルがないとコンパイルが出来ないので、一つのソースプログラムを作るたびにそれに対応するサブファイルを作っておく。

（2）ソースプログラムとサブファイルを同じフォルダ内に移動させておく。
（3）DOS プロンプトを立ち上げ、DOS Window を表示する
（4）DOS Window 内部でカレントディレクトリを作業フォルダに変更する。
　　>cd 作業ディレクトリ

（5）「addh8path.bat」というファイル名のバッチファイルを DOS プロンプト上で実行する。そうすると検索パスが自動的に指定される。

　　>addh8path

「addh8path.bat」は以下のような書式である。

---------- head of list ----------
```
path=%path%;c:¥Progra~1¥h8¥c
```
---------- end of list ----------

ここまでがコンパイルの準備である。

（6）「h8cc.bat」というファイル名のバッチファイルを実行してコンパイルする。そうするとダウンロード形式ファイル（XXXX.mot）を作成することが出来る。
ソースの誤りは検出しエラー表示を行うが、リンク時の誤りは検出してくれないので表示を注意して見なければならない。

　　>h8cc ソースファイル名（拡張子なし）

LED の場合　　>h8cc LED
　　これにより LED.mot が作成される

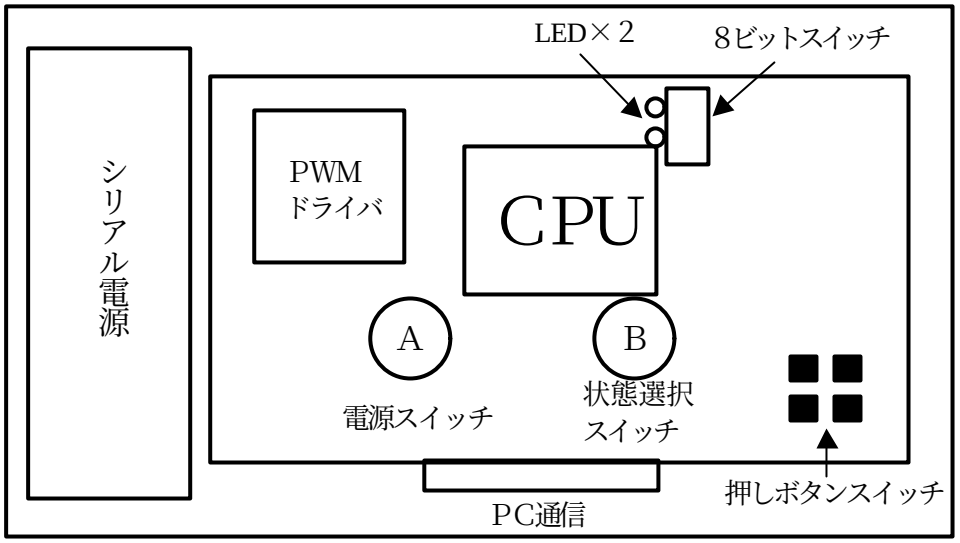
以下に H8CC.BAT の書式を示す。

---------- head of list ----------
```
rem cc38h のディレクトリに path を切っておくこと
echo off
if exist %1.obj del %1.obj
cc38h -include=c:¥Progra~1¥h8¥c %1
if not exist %1.obj goto ATERM

if exist %1.abs del %1.abs
if exist %1.obj l38h -subcommand=%1.sub
if not exist %1.abs goto ATERM

if exist %1.abs c38h %1.abs
goto TERM
:ATERM
echo ******* エラーがありました
:TERM
echo on
```
---------- end of list ----------

4．ROM ライター書き込み操作



付録図 4-1　マザーボードボックスを上から見た場合のスイッチ

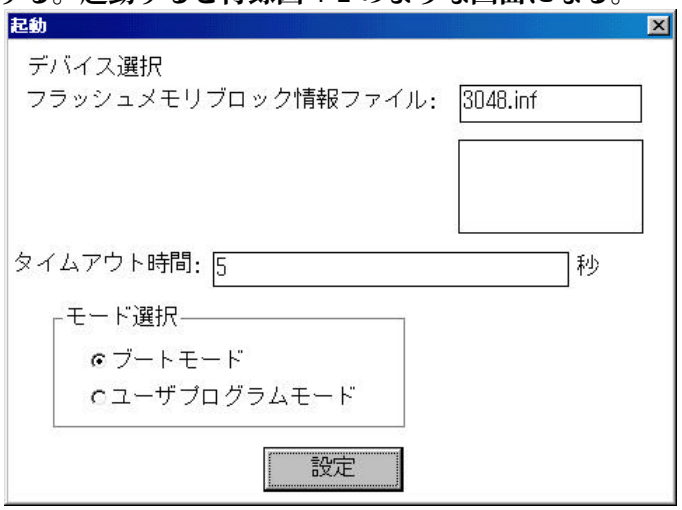
ダウンロード形式ファイル（XXXX.mot）を作成したら、シングルボードマイクロコンピュータのフラッシュ RAM（電源が切れても覚えている RAM）にそのファイルを書き込む作業が必要である。そのための手順を以下に示す。

（1）12V 入力（整流回路出力）用端子に整流回路電源をつなげる。

（2）電源スイッチ（付録図 4-1 の A、以下の説明で単に「電源スイッチ」）を下に倒し OFF にする。そしてを状態選択スイッチ（付録図 4-1 の B、以下の説明で単に「状態選択スイッチ」）を上に倒しライターモードにする.

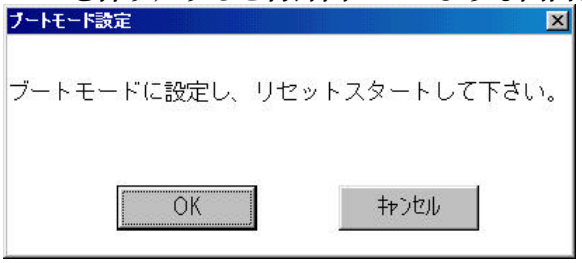
（3）電源スイッチを上に倒し ON にする。

（4）flash.exe を起動する。起動すると付録図 4-2 のような画面になる。



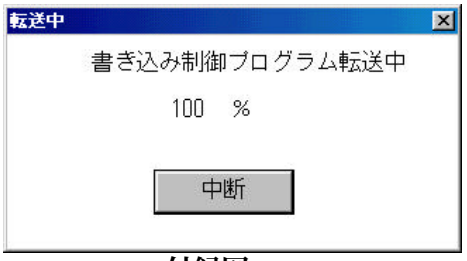
付録図 4-2

（5）設定をクリック（ENTER を押す）すると付録図 4-3 のような画面になる。



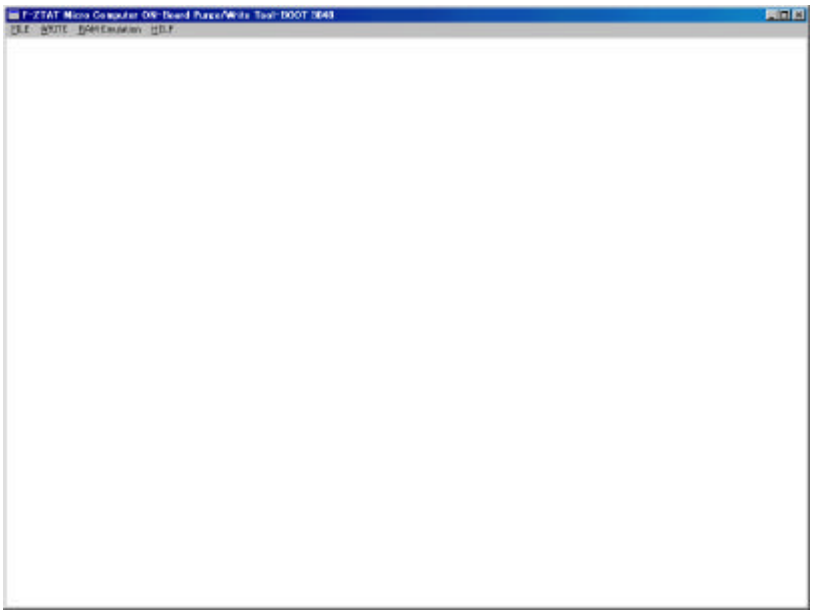
付録図 4-3

（6）OK をクリックすると付録図 4-4 のような画面になる。



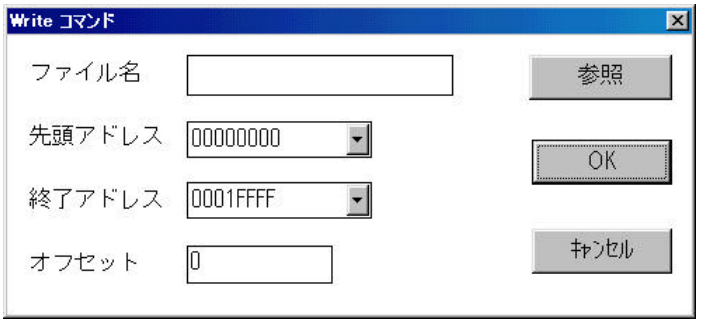
付録図 4-4

（8）転送が終了すると付録図 4-5 のような画面になる。



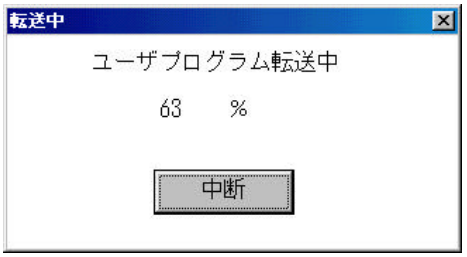
付録図 4-5

（9）メニューの Write を選択する。



付録図 4-6

（１０）参照を選択または、ファイル名のところに転送したい MOT ファイル名（例 c:¥xx…¥sample.mot）をフルパスで書きこみ、OK をクリックする。



付録 4-7

（１１）これで転送終了が終了したので、電源スイッチを下に倒し OFF にする。

（１３）状態選択スイッチを下に倒し通常モードにする。そして電源スイッチを上に倒し電源を ON にすると転送したプログラムが作動する。