講義「人を測る」

作成者：松下光次郎

1　講義「人を測る」の概要

趣旨

１．人から発せられる生体信号を用いてロボットを制御するプログラムの作成 マイクロチップ制御器における RC サーボ・LED の使い方を理解する（C 言語）．また，生体信号がどのようなものかを実測を通じて理解する．
２．ロボットの機構を製作 ペットボトルなどを用いて簡単な移動ロボットを製作し，ロボットを設計する上で，如何に身体性が重要かを理解する．

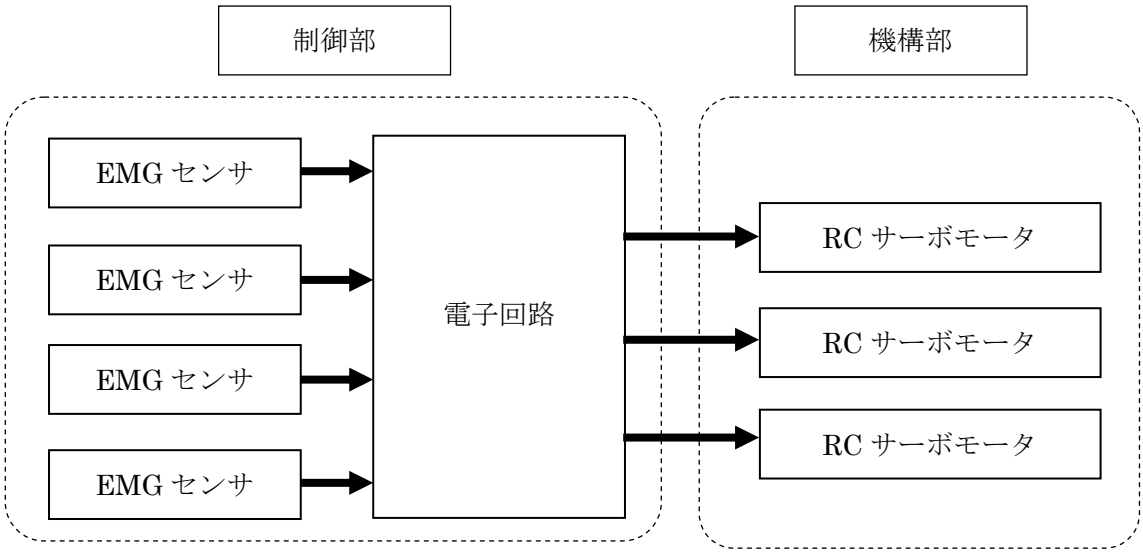
ロボットのシステム構成

ロボットのシステム構成は以下のように制御部と機構部からなっている．講義の受講者は，

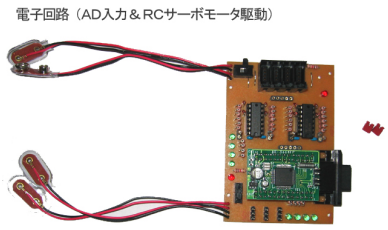
1)　複数の EMG センサから計測される**生体信号を処理しモータを制御するプログラムを作成**．

2)　複数の RC サーボモータを用いた**ロボットの機構を設計・製作**

することで，**「生体信号を用いて制御するロボット」**を実際に製作する．



EMG センサ



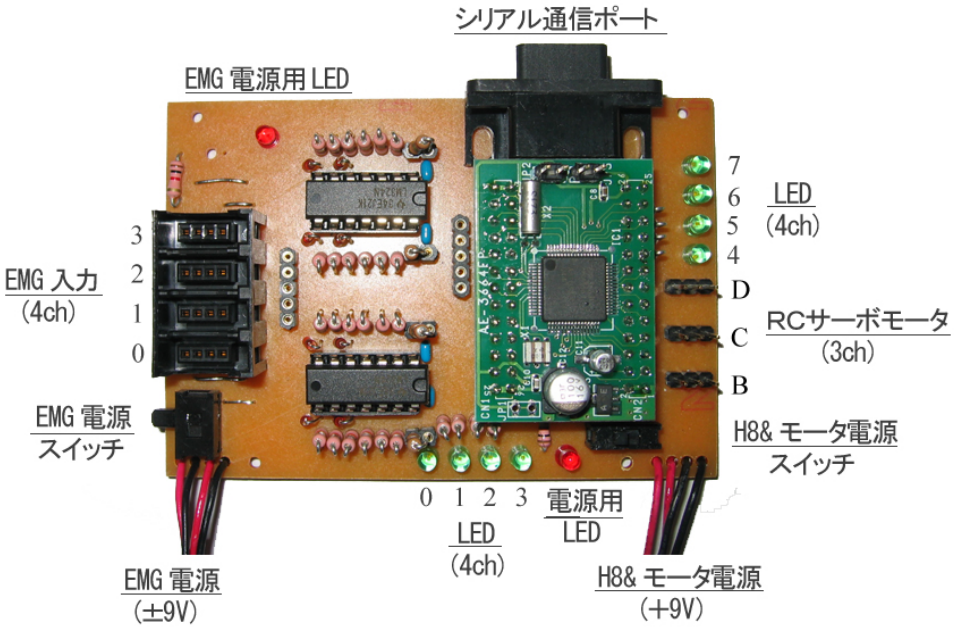
電子回路

機構部
(例:ロボットアーム)

2　システム概要

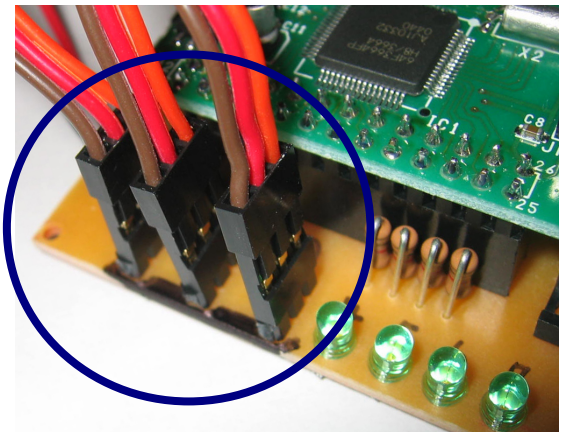
2.1　電子回路

配布された電子回路は，RC サーボモータ 3 個を制御，EMG センサ 4 個の状態を監視，さらに LED8 個の点灯・消灯を制御可能な制御器である．マイクロチップには秋葉原にある秋月電商にて購入した H8/3664 を使用している．このマイコンは，シリアル通信によりプログラムを書き込むシステムである．



電子回路ポート	内容
PWM 制御信号 3ch	RC サーボモータ制御
A/D 変換入力 4ch	EMG センサの強弱監視
デジタル出力 8ch	LED 点灯・消灯

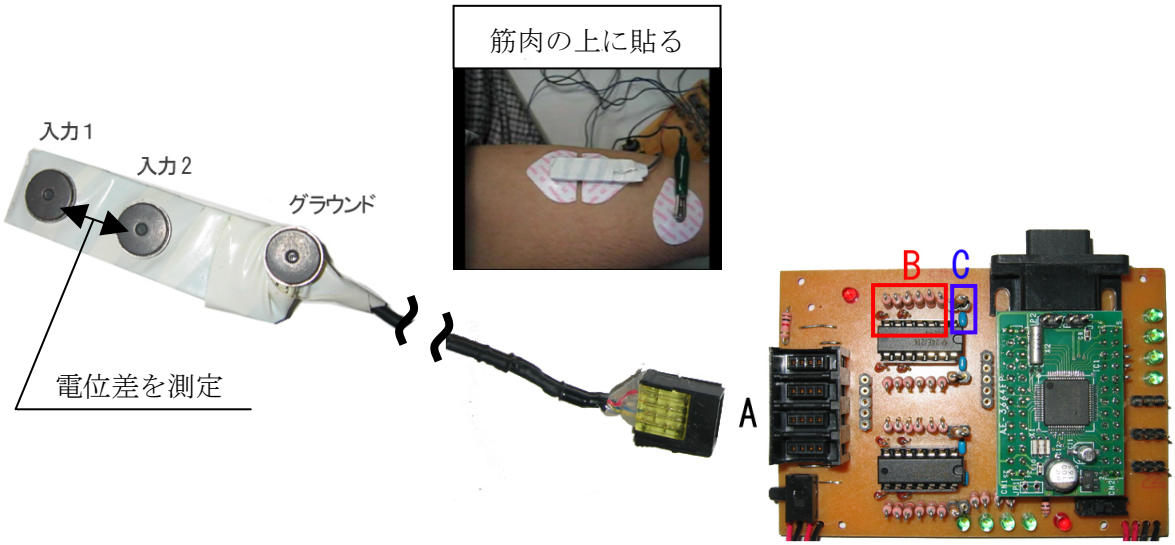
*接続時に，サーボモータのコネクタの向きには気をつける必要があります．

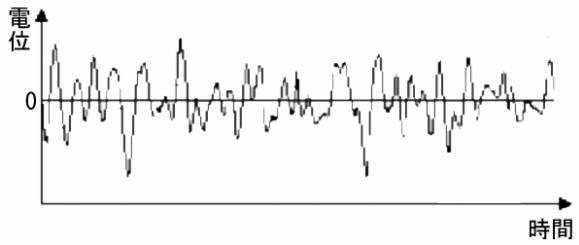
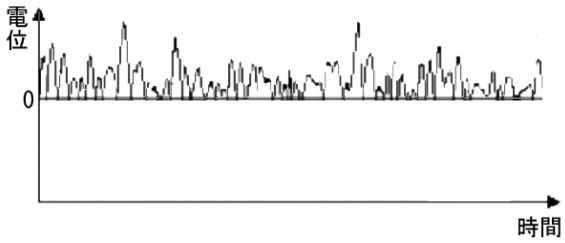
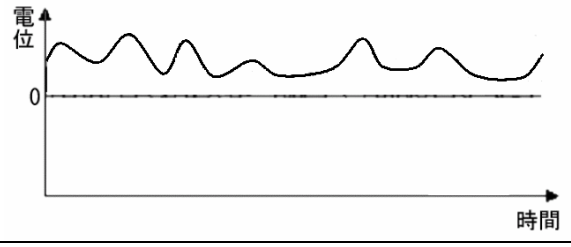


RC サーボを接続するときはコネクタの向きに気をつけること！
茶色の Ground 線が電子回路の外側に！

2.2　表面筋電位(EMG)センサ ＆ 信号処理電子回路（全波整流，ローパスフィルタ）

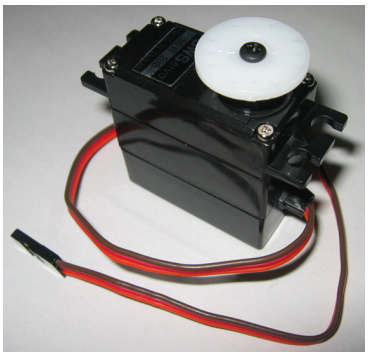
EMG センサは，人間の腕や脚などを動かす際に生じる筋肉の微弱な電位(筋電位)を皮膚表面で検出可能なセンサである．このセンサは，図に示す入力 1 と入力 2 の 2 電極間の電位差を増幅する回路となっており数十μV である筋電位信号を±9V の範囲まで増幅し出力する．また，この授業においては，前処理として電子回路上で信号の全波整流（信号の絶対値変換）及びローパス（信号の平滑化）を施すことで，**信号の振幅の大きさ**をそのままマイコンに取り込める形となっている．



信号処理順番	生体信号（EMG 信号）
A．EMG センサ 表面筋電位 1 チャンネルは EMG センサにより-9V～+9V の信号に増幅される．	
B．全波整流（電子回路） ±9V の電位で出力されている 信号の絶対値変換（0～9V）	
C．RC ローパスフィルタ（電子回路） 抵抗値：2MΩ，コンデンサ値：0.1uF カットオフ周波数 fc=1/2RC ローパスフィルタ後，信号は鈍るため おおよそ 0～+5V の信号となる．	

2.3　RC サーボモータ

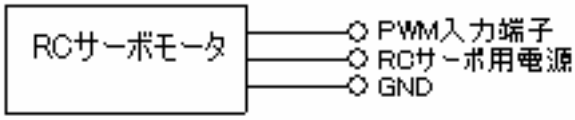
ラジコン用サーボモータ（RC サーボ）は，DC モータ ，減速機構（ギヤボックス），サーボアンプ（モータを駆動するための回路）が一体になっていて簡単に扱うことのできるアクチュエータといえます．RC サーボには，アナログサーボ（プラスチックギア・メタルギア）・デジタルサーボなどがあります．デジタルサーボはアナログサーボに比べて出力トルクも高く．ラジコンショップや模型店での入手が可能であり，価格も安く，制御が簡単でとても使いやすいため，様々なロボットで RC サーボが使用されています．



標準型 RC サーボ　GWS S03T/2B			
性能		トルク	速度
	4.8V	7.2kg-cm	0.33sec/60°
	6.0V	8.0kg-cm	0.27sec/60°
重量	46 g		
寸法	39.5 × 20.0 × 39.6mm		

図　RC サーボモータ

RC サーボは，図のようにケーブル 3 本とコネクタがついています．このケーブル 3 本を制御信号・電源電圧・グラウンド接続して初めて動かすことができます．制御信号とは幅の違う矩形波（PWM）のことであり，幅によりモータの位置を制御しています．PWM については，次の節で説明します．



橙色：　PWM 入力端子
赤色：　RC サーボ用電源
茶色：　グラウンド

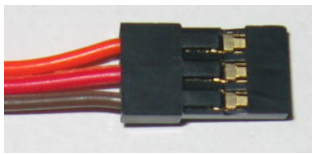
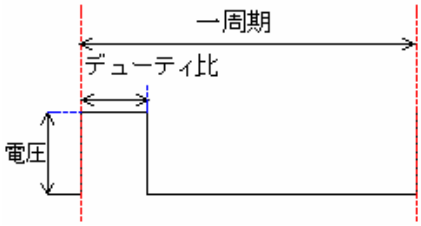


図　RC サーボのコネクタ

2.4　RC サーボモータのモータの制御方法 - PWM（パルス幅変調）

パルス変調幅(PWM)は簡単にいうと,高速でモータ電源の ON/OFF を繰り返すことです.この ON/OFF の繰り返しにはルールがあり，ある一定時間での ON にする時間と OFF にする時間の比を変えてあげるのです．この比をデューティ比といいます．デューティ比が高いほどＯＮになっている期間が長く，デューティ比がゼロではずっとＯＦＦのままである様子を示しています．ここで重要なのは，先ほど述べた“ある一定時間”，つまり，パルスを与える周期です．何秒おきに ON/OFF を繰り返すか最適に設定しなければなりません.



パルス変調幅	
周期	20msec など.
デューティ比	一周期中の High の占める割合 0～100%.
電圧	一般的に 5V.

RC サーボは，マイクロチップから送られる PWM 制御信号のパルス幅を読み取り，モータを決められた角度に動かします．つまり，デューティ比がモータの角度に対応しているのです．一般の RC サーボモータにおいて，パルス：３～５V，周期：20ms，幅：0.9ms～2.1ms(中心 1.5ms)です．RC サーボは一般的に 180° 回転が限界ですので，0.1ms のパルス幅変化が 15° の角度変化に対応しています.

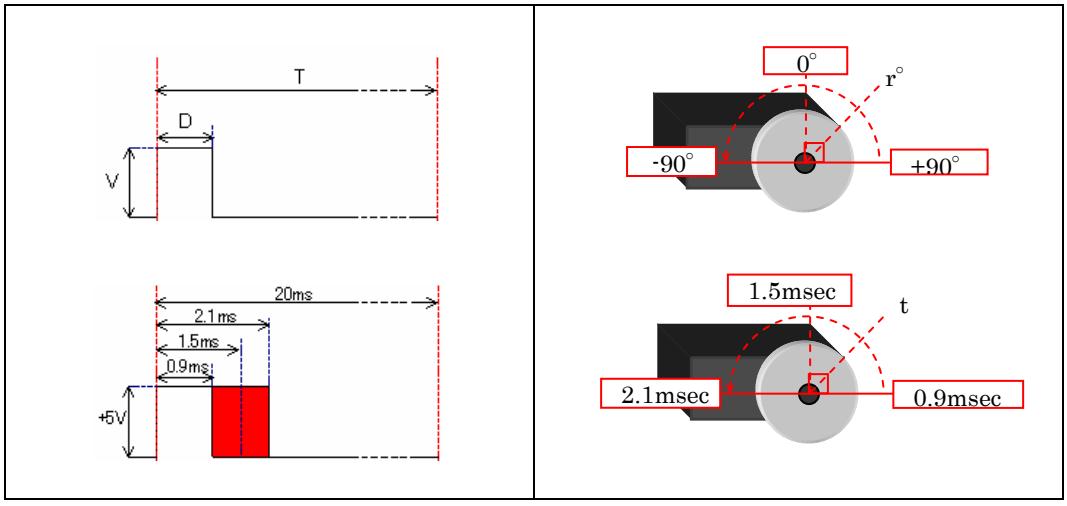


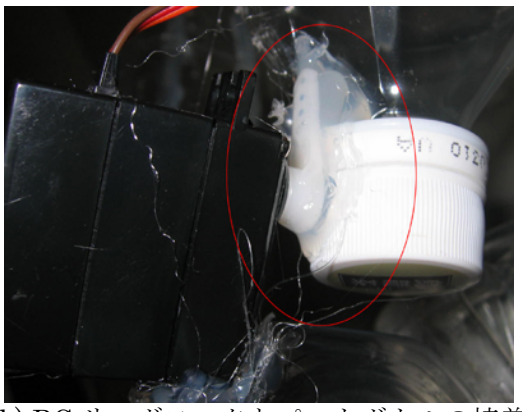
表　RC サーボモータの制御信号

PWM 制御信号				
周期 T	20msec(50Hz) PWM レジスタ GRA = 40000 (16MHz8 周分)			
制御		始点	中央	終点
	デューティ比 d	4.5%	7.5%	10.5%
	時間 t	0.9ms	1.5ms	2.1ms
	角度 r	-90°	0°	90°
	PWM レジスタ GRB, GRC, GRD	3000-1600	3000	3000+1600

2.5　ペットボトルで作るロボット（機構部）

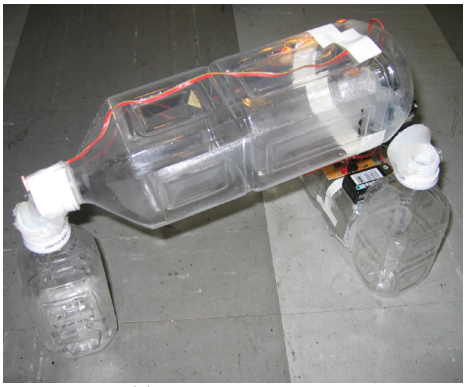
一般にロボットは金属材料やプラスチックで作られており，自分で作るのは難しい．そこで，この授業ではペットボトルなど家庭用品を用いて簡単にロボットを作る方法を紹介する．主に必要とされる道具は，ホームセンターなので簡単に購入できる「グルーガン」である．このグルーガンは，くっつけたい部分に高温で溶かし出し，冷えることで固まり，接着するものである．例えば図(b)に示すように，サーボモータの軸とペットボトルと接着することもできる．このような方法により，簡単にロボットを作れ，簡単にロボットの形を変れるという利点がある．ただし欠点としては，ボンドが簡単にとれることもあるので，こまめに手直しすることが必要となる場合もある．

(a) グルーガン



(b) RC サーボモータとペットボトルの接着

この方法により製作された 2 つのロボットを紹介する．一つは，サーボモータ 1 個で作られている移動ロボットである（左図）．周期的にサーボモータを振動させることで地面を蹴り前にすすむ．もう一つは，腕ロボットである．３個のサーボモータから構成され，軽い物を持ち，動かすことができる（右図）．



(c) 移動ロボット

(d) ロボットアーム

さて，この講義では，ロボット機構と EMG 制御器を製作し，**「生体信号を用いて制御するロボット」**を作ってみよう．

3　Ｈ８プログラム書き込み環境準備

3.1　必要ファイルの入手

圧縮ファイル「monotuskuri.exe」を手に入れ，解凍してください．このファイルは，自動解凍機能付なので，パスワードを入力すれば解凍できます．解凍すると，２個のフォルダを確認することができます．１つは，USB シリアルケーブルのドライバ．もう１つは，制御器にプログラムを書き込むためのコンパイラおよび書き込みソフトセットです．

ダウンロード：http://www.koj-m.sakura.ne.jp/edutainment/part2/monotsukuri.exe

圧縮ファイル	パスワード
monotsukuri.exe	

ファイル名	内容
arvel_usbtoserial_driver	USB シリアルケーブル用ドライバ
h8	H8/3664 用コンパイラおよび書込ソフト．C ドライブに置く．

3.2　USB シリアルケーブルのインストール

最近のノートパソコンには，図のようなシリアル通信用ポート［Dsub９ピン（オス)］が使用されなくなってきています．ノートパソコンに使用されてない場合は，USB シリアルケーブルを使用してもらうこととなります. このケーブルは USB を利用して仮想的にシリアル通信用ポートとする変換器となっています.

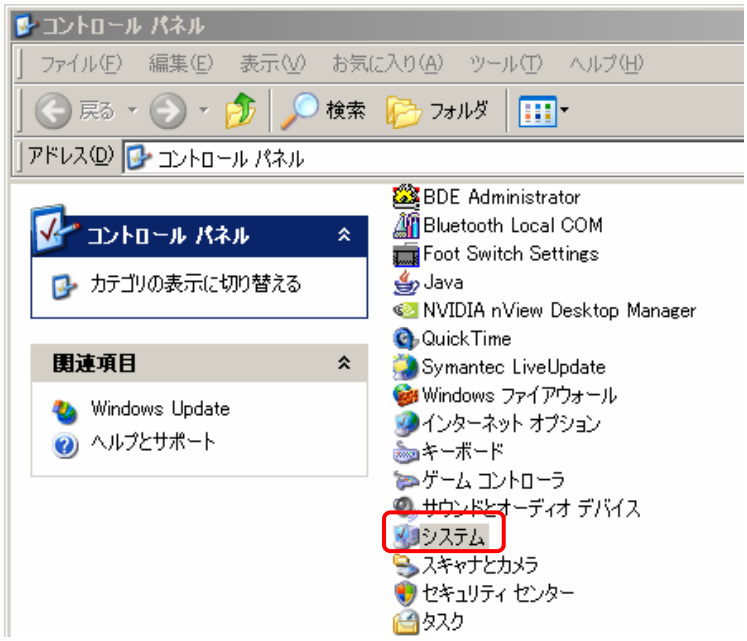
ドライバのインストールには，先に解凍した「arvel_usbtoserial_driver」フォルダを指定して行ってください．WindowsXP に認証されていないと表示されますが，問題なくインストールできます.

シリアル通信用ポート
Dsub9 ピン（オス）

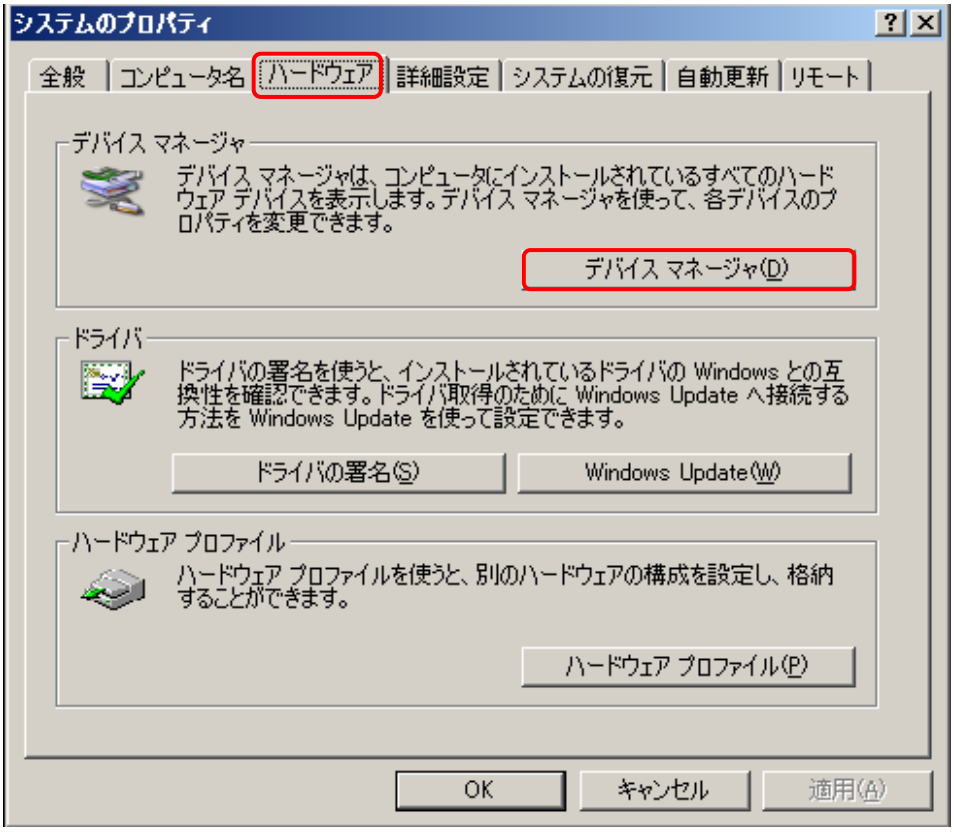
USB シリアルケーブル

次に，USB シリアル通信用ケーブルを「COM1」に設定する必要があります．COM1 とは，第一番目のシリアル通信用ポートに指定するということです. どのメーカの USB シリアル通信用ケーブルを使用する場合でもこの手順を行い，「COM1」であるかどうか確認してください．

まず,「コントロールパネル」を開き,「システム」のアイコンをダブルクリックしてください．

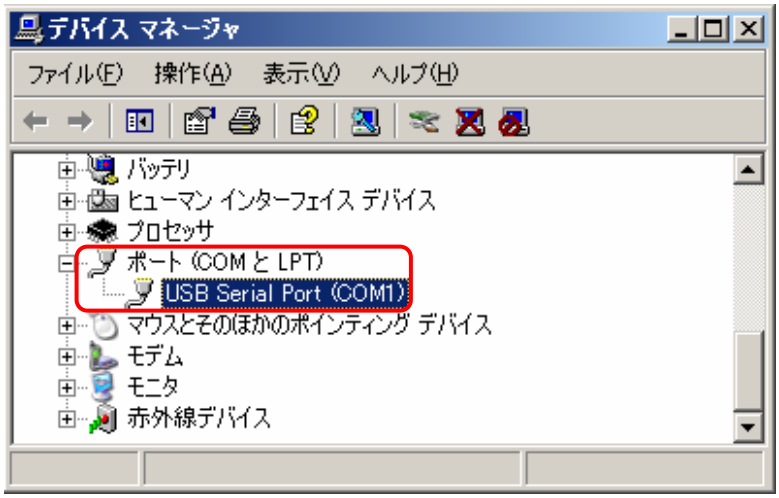


「システムのプロパティ」ウィンドウのタブである「ハードウェア」を選択肢,「デバイスマネージャ」を指定してください．

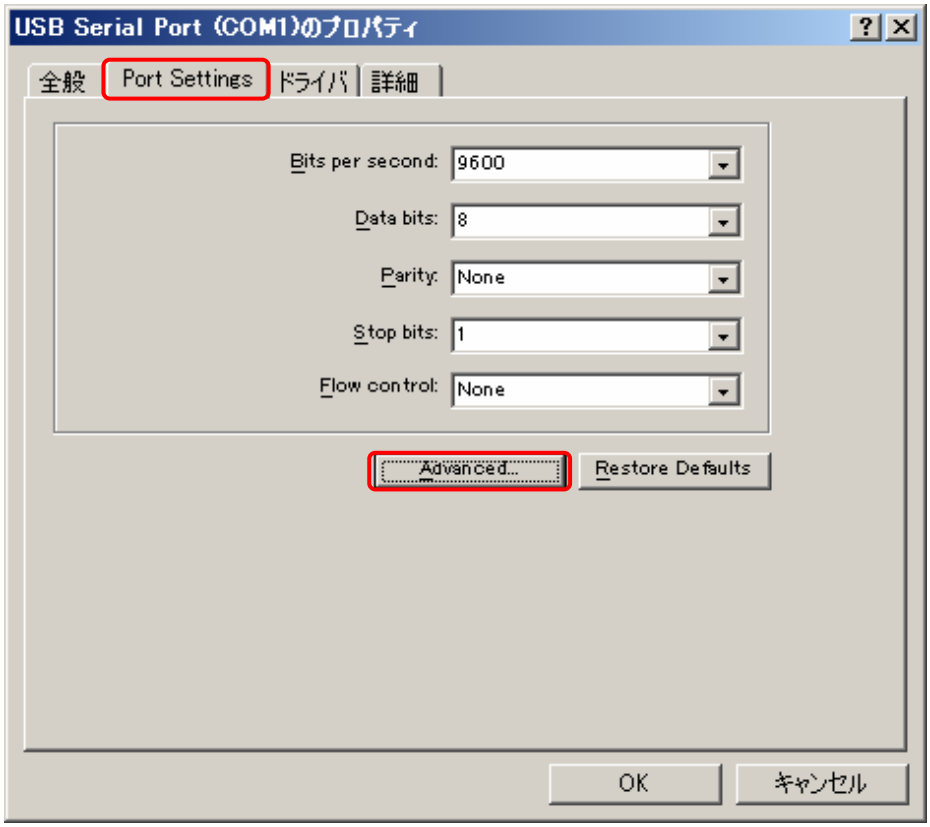


「デバイスマネージャ」のツリー構造中にある「ポート（COM と LPT)」を選択すると，「USB Serial Port (COM1)」と確認することができます．ここで「COM1」と表示されている場合は，問題ないのでこの手順を終了してもらって問題ありません．

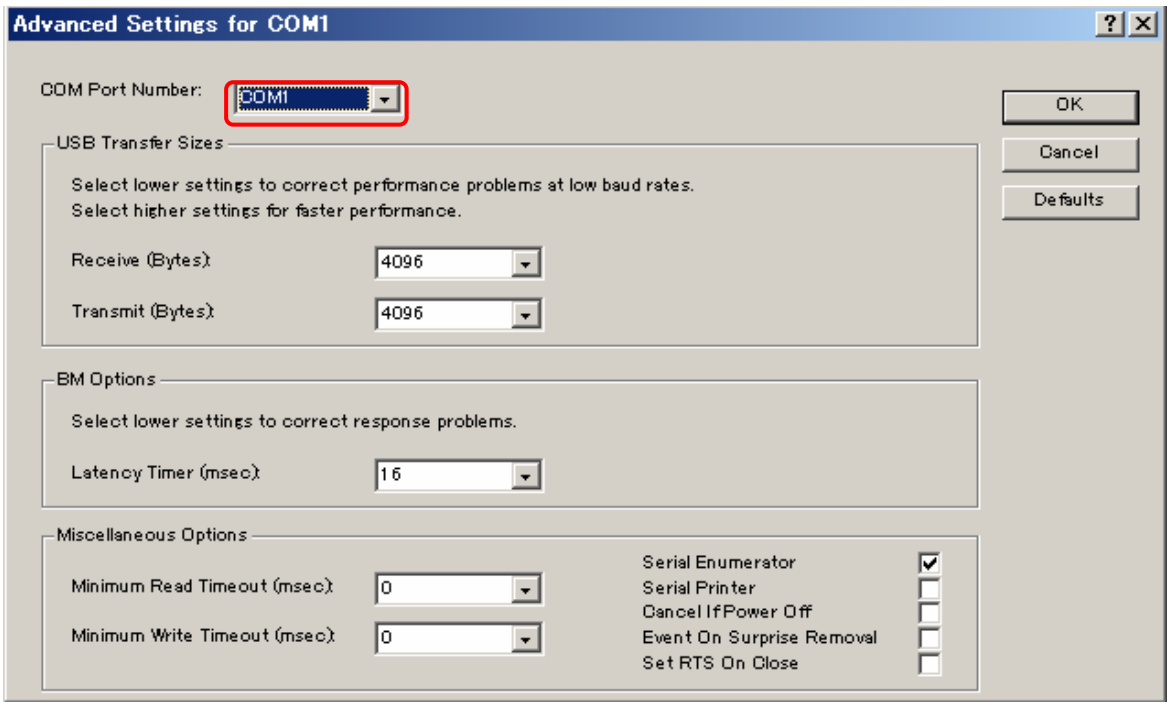
ここで「COM2」や「COM5」と表示されている場合は，「COM1」に直す必要があります．「USB Serial Port (COM2)」をダブルクリックしてください．



「USB Serial Port (COM2)のプロパティ」ウィンドウを開いき，「Port Settings」のタブを選択し，「Advanced…」をクリックしてください．

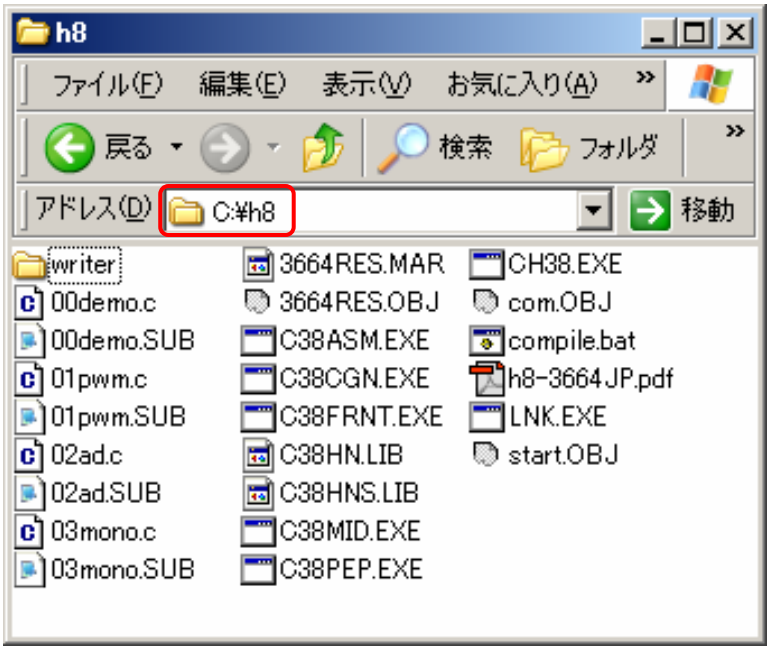


「Advanced Settings for COM2」ウィンドウの最上段に「COM Port Number:」という項目があります．ここの項目で「COM1」をして，「OK」ボタンを押し，この手順を終了してください．



3.3　Ｈ８プログラム書き込み環境準備

先ほど，解凍した「h8」フォルダを「C ドライブ」におき，「h8」のフォルダの内容を見た時そのアドレスバーの表示が「C:¥h8」と表示されているか確認してください．



4　Ｈ８プログラム作成方法

4.1　サンプルプログラム概要

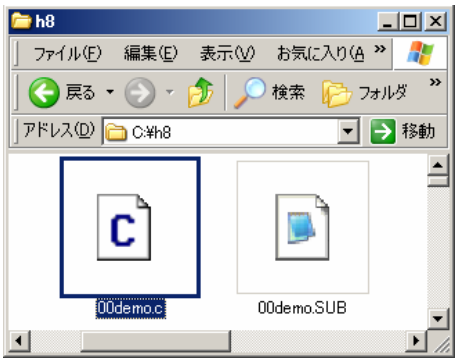
授業では，３個のサンプルファイルを用意しています．H8 のプログラムは C ファイルと Sub ファイルが必要となっており，下表に示すとおりサンプルプログラムにおいて，C と Sub ファイルのセットとなっております．

* 付録にサンプルプログラムがあり，色付部のみを変更することで利用できます．

	ファイル	内容
サンプル１	01pwm.c 01pwm.sub	RC サーボモータ振動制御
サンプル２	02ad.c 02ad.sub	EMG センサ読取および LED を用いたゲージ表示
サンプル３	03emg.c 03emg.sub	EMG の強弱をサーボの角度にマッピング

4.2　新しい H8 プログラムの作成方法手順

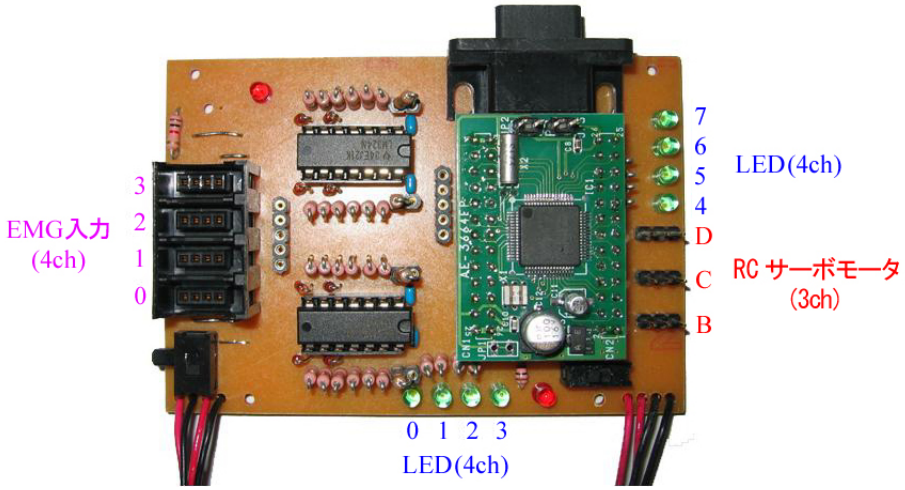
新しいプログラムを作成する場合は，「C:¥h8」フォルダ内でいずれかのサンプルの C ファイルと Sub ファイルをコピーし，それらの名前を変更し，且つ，Sub ファイル内のファイル名部分を変更してください．その後，C ファイルにプログラムを書き込んでください．C ファイル名，Sub ファイル名，Sub ファイル内容のファイル名部分（色付部）は，必ず統一してください．



ファイル名	01pwm.sub
OUTPUT 01pwm PRINT 01pwm INPUT 3664RES, 01pwm LIB c38hn START P(100) EXIT	

4.3　電子回路と H8 プログラム記述方式

電子回路の各ポートの H8 プログラムにおける記述方式を下表に示します.



電子回路		プログラム（C 言語）	
RC サーボモータ (PWM 出力) 3ch	B	TW.GRB	RC サーボの軸位置 左端 ← 中央 → 右端 1400　3000　4600
	C	TW.GRC	
	D	TW.GRD	
EMG 入力 （AD 入力） 4ch	0	x0	EMG の強弱（8bit） 弱い(0) ←→ 強い(255) *0～255 の値で表現
	1	x1	
	2	x2	
	3	x3	
LED （デジタル出力） 8ch	7	IO.PDR1.BIT.B7	LED 点灯・消灯 点灯：IO.PDR1.BIT.B7 = 1 消灯：IO.PDR1.BIT.B7 = 0 *2 種類のポートで構成 Port1 の B7-4，Port5 の B3-B0
	6	IO.PDR1.BIT.B6	
	5	IO.PDR1.BIT.B5	
	4	IO.PDR1.BIT.B4	
	3	IO.PDR5.BIT.B3	
	2	IO.PDR5.BIT.B2	
	1	IO.PDR5.BIT.B1	
	0	IO.PDR5.BIT.B0	

4.4　H8 におけるＣプログラミングの注意事項

● コメントは「// 」ではなく，「/* */」で記入すること.

● 二つの条件を含む if 文の場合，「if(x0<10 && x0>50)」とはせず，二つの条件必ず括弧で囲「if((x0<10)&&(x0>50))」とすること

● このコンパイラは，「if(x0=20)」の記述間違いにおいてエラーが表示されないので，気をつけること.（if 文が「if(x0==20)」というように「==」で記述されているのを確認すること.）

4.5　付録：サンプルプログラム

ファイル名	01pwm.c
内容	１秒ごとに，３個のＲＣサーボが左右交互に振動する． B: 振幅(小)　C: 振幅(中)　D: 振幅(大)

```
#include <3664f.h>
void main(void){
        int t=0;          /* Time           */
        int flag=0;       /* Right/Left     */

        IO.PCR8 = 0xff;
        TW.GRA = 40000;             /* 周期:20msec     */
        TW.TMRW.BYTE = 0xcf;        /* TimerW setting  */
        TW.TCRW.BYTE = 0xbf;        /* TimerW setting  */

        TW.GRB = 3000;              /* モータBch位置：中央 */
        TW.GRC = 3000;              /* モータCch位置：中央*/
        TW.GRD = 3000;              /* モータDch位置：中央*/

        while(1){
                if(TW.TSRW.BIT.IMFA==1){    /* GRA:20ミリ秒間隔で実行 */
                        TW.TSRW.BIT.IMFA=0;
                        if(t>50){/* 20msec*50=1000ミリ秒間隔で実行 */
                                /* フラッグの状態により交互に位置が変わる */
                                if(flag==0){
                                        TW.GRB = 2800;     /* モータBch位置：左端 */
                                        TW.GRC = 2600;     /* モータCch位置：左端 */
                                        TW.GRD = 2400;     /* モータDch位置：左端 */
                                        flag=1;
                                }
                                else {
                                        TW.GRB = 3200;     /* モータBch位置：右端 */
                                        TW.GRC = 3400;     /* モータCch位置：右端 */
                                        TW.GRD = 3600;     /* モータDch位置：右端 */
                                        flag=0;
                                }
                                t=0;
                        }
                        t++;
                }
        }
}
```

サーボモータの制御に関してのプログラム．GRB, GRC, GRD の値が各サーボモータの位置に対応しており．フラッグ「flag」の状態により，１秒ごとにモータの位置を切り替えている．

ファイル名	02ad.c
内容	20m 秒ごとに 1 個の EMG センサから得られる値を読み取り，それらの値により LED の点灯状態を決定する．EMG が強いほど多くの LED が点灯する．

```
#include <3664f.h>
typedef unsigned char UInt8;

void main(void){
        unsigned int* data;
        UInt8 x0, x1, x2, x3;       /* AD:0ch-3ch      */

        IO.PMR1.BYTE = 0x00;        /* Port1:DIO       */
        IO.PMR5.BYTE = 0x00;        /* Port5:DIO       */
        IO.PCR1 = 0xff;             /* Port1:Output    */
        IO.PCR5 = 0xff;             /* Port5:Output    */

        IO.PCR8 = 0xff;
        TW.GRA = 40000;             /* Freq:20msec     */
        TW.TMRW.BYTE = 0xcf;        /* TimerWsetting   */
        TW.TCRW.BYTE = 0xbf;        /* TimerWsetting   */

        AD.CSR.BIT.SCAN=0;
        AD.CSR.BIT.CKS=1;
        AD.CSR.BIT.CH=0;

        while(1){
                if(TW.TSRW.BIT.IMFA==1){    /* 20ミリ秒間隔で実行 */
                        TW.TSRW.BIT.IMFA=0;
                        /* AD0chのみの読取 */
                        AD.CSR.BIT.ADST=1;
                        while(AD.CSR.BIT.ADF==0){};
                        AD.CSR.BIT.ADF=0;
                        data=&AD.DRA; x0=(int)(*data>>8); /* x0は 0-255(8bit)で表現される  */
                        AD.CSR.BIT.ADST=0;

                        /* 0chの電位に対応したLED点灯ゲージ表示 */
                        if( (x0>=0) && (x0<30) ){
                                IO.PDR1.BIT.B7=0;  /* LED7：消灯      */
                                IO.PDR1.BIT.B6=0;  /* LED6：消灯      */
                                IO.PDR1.BIT.B5=0;  /* LED5：消灯      */
                                IO.PDR1.BIT.B4=0;  /* LED4：消灯      */
                                IO.PDR5.BIT.B3=0;  /* LED3：消灯      */
                                IO.PDR5.BIT.B2=0;  /* LED2：消灯      */
                                IO.PDR5.BIT.B1=0;  /* LED1：消灯      */
                                IO.PDR5.BIT.B0=0;  /* LED0：消灯      */
                        }
                        else if( (x0>=30) && (x0<60) ){
                                IO.PDR1.BIT.B7=1;  /* LED7：点灯      */
                                IO.PDR1.BIT.B6=0;  /* LED6：消灯      */
                                IO.PDR1.BIT.B5=0;  /* LED5：消灯      */
                                IO.PDR1.BIT.B4=0;  /* LED4：消灯      */
                                IO.PDR5.BIT.B3=0;  /* LED3：消灯      */
                                IO.PDR5.BIT.B2=0;  /* LED2：消灯      */
                                IO.PDR5.BIT.B1=0;  /* LED1：消灯      */
                                IO.PDR5.BIT.B0=0;  /* LED0：消灯      */
                        }
                        else if( (x0>=60) && (x0<90) ){
                                IO.PDR1.BIT.B7=1;
                                IO.PDR1.BIT.B6=1;
```

```
                IO.PDR1.BIT.B5=0;
                IO.PDR1.BIT.B4=0;
                IO.PDR5.BIT.B3=0;
                IO.PDR5.BIT.B2=0;
                IO.PDR5.BIT.B1=0;
                IO.PDR5.BIT.B0=0;
        }
        else if( (x0>=90) && (x0<120) ){
                IO.PDR1.BIT.B7=1;
                IO.PDR1.BIT.B6=1;
                IO.PDR1.BIT.B5=1;
                IO.PDR1.BIT.B4=0;
                IO.PDR5.BIT.B3=0;
                IO.PDR5.BIT.B2=0;
                IO.PDR5.BIT.B1=0;
                IO.PDR5.BIT.B0=0;
        }
        else if( (x0>=120) && (x0<150) ){
                IO.PDR1.BIT.B7=1;
                IO.PDR1.BIT.B6=1;
                IO.PDR1.BIT.B5=1;
                IO.PDR1.BIT.B4=1;
                IO.PDR5.BIT.B3=0;
                IO.PDR5.BIT.B2=0;
                IO.PDR5.BIT.B1=0;
                IO.PDR5.BIT.B0=0;
        }
        else if( (x0>=150) && (x0<180) ){
                IO.PDR1.BIT.B7=1;
                IO.PDR1.BIT.B6=1;
                IO.PDR1.BIT.B5=1;
                IO.PDR1.BIT.B4=1;
                IO.PDR5.BIT.B3=1;
                IO.PDR5.BIT.B2=0;
                IO.PDR5.BIT.B1=0;
                IO.PDR5.BIT.B0=0;
        }
        else if( (x0>=180) && (x0<210) ){
                IO.PDR1.BIT.B7=1;
                IO.PDR1.BIT.B6=1;
                IO.PDR1.BIT.B5=1;
                IO.PDR1.BIT.B4=1;
                IO.PDR5.BIT.B3=1;
                IO.PDR5.BIT.B2=1;
                IO.PDR5.BIT.B1=0;
                IO.PDR5.BIT.B0=0;
        }
        else if( (x0>=210) && (x0<240) ){
                IO.PDR1.BIT.B7=1;
                IO.PDR1.BIT.B6=1;
                IO.PDR1.BIT.B5=1;
                IO.PDR1.BIT.B4=1;
                IO.PDR5.BIT.B3=1;
                IO.PDR5.BIT.B2=1;
                IO.PDR5.BIT.B1=1;
                IO.PDR5.BIT.B0=0;
        }
        else {
                IO.PDR1.BIT.B7=1;
```

```
                                        IO.PDR1.BIT.B6=1;
                                        IO.PDR1.BIT.B5=1;
                                        IO.PDR1.BIT.B4=1;
                                        IO.PDR5.BIT.B3=1;
                                        IO.PDR5.BIT.B2=1;
                                        IO.PDR5.BIT.B1=1;
                                        IO.PDR5.BIT.B0=1;
                                }
                        }
                }
}
```

AD0ch に入力されている EMG センサの値を 8bit で読み取っている．すなわち，0=0V であり，255=5V である．更には，20 ミリごとにその値に対応して，条件分岐により LED の点灯状態を変化させている．

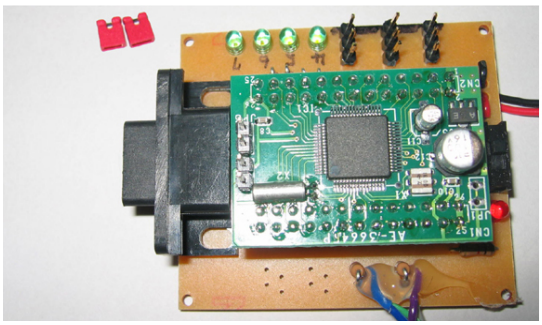
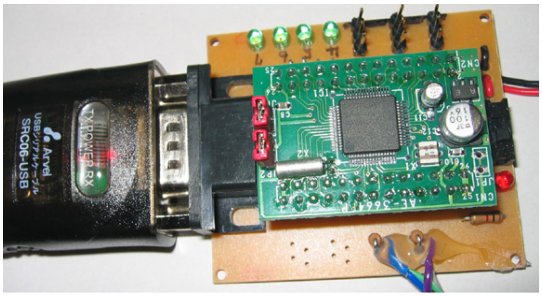
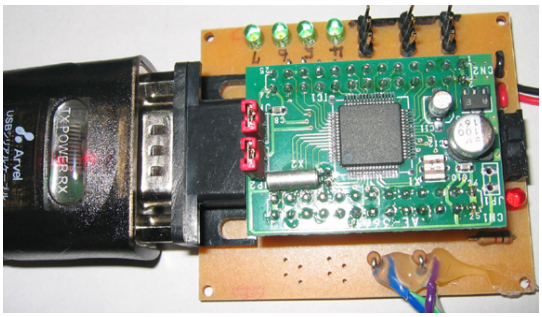
ファイル名	03emg.c（初期インストール済プログラム）
内容	「01pwm.c」と「02ad.c」に基に，20m 秒ごとに１個の EMG センサ値を読み取り，その電位の強弱を１個のＲＣサーボの角度に対応させる．

```
#include <3664f.h>
typedef unsigned char Uint8;
void main(void){
        int bpos=1400;
        int gain=5;
        unsigned int* data;
        Uint8 x0, x1, x2, x3;       /* AD:0ch-3ch  */
        IO.PMR1.BYTE = 0x00;        /* Port1:DIO       */
        IO.PMR5.BYTE = 0x00;        /* Port5:DIO       */
        IO.PCR1 = 0xff;             /* Port1:Output    */
        IO.PCR5 = 0xff;             /* Port5:Output    */
        IO.PCR8 = 0xff;
        TW.GRA = 40000;             /* Freq:20msec     */
        TW.TMRW.BYTE = 0xcf;        /* TimerW setting  */
        TW.TCRW.BYTE = 0xbf;        /* TimerW setting  */
        TW.GRB = bpos;              /* モータBch位置：0deg */
        AD.CSR.BIT.SCAN=1;
        AD.CSR.BIT.CKS=1;
        AD.CSR.BIT.CH=3;
        while(1){
                if(TW.TSRW.BIT.IMFA==1){            /* GRA:20msec間隔で実行 */
                        TW.TSRW.BIT.IMFA=0;
                        AD.CSR.BIT.ADST=1;          /* EMGセンサ0ch読取 */
                        while(AD.CSR.BIT.ADF==0){};
                        AD.CSR.BIT.ADF=0;
                        data=&AD.DRA; x0=(int)(*data>>8);     /* アナログ入力0ch */
                        AD.CSR.BIT.ADST=0;
                        TW.GRB = bpos+(int)x0*gain;          /* Bchの左端位置 */
                }
        }
}
```

AD0ch で EMG センサ電圧値が読み取られ，プログラム内において 0-255(8bit)で表現される．また，その 0-255 の電圧値をサーボモータの角度に対応させ，筋電位の強弱でモータの角度が変化するようにしている．

5　Ｈ８プログラムのコンパイルおよび書き込み方法

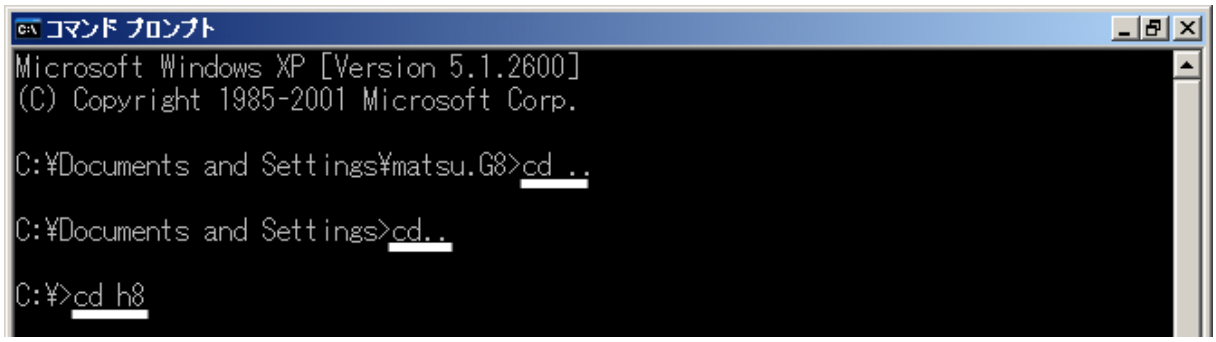
5.1　H8（ハードウェア）の書き込み準備

内容	写真
書込手順 1 書き込み用ジャンパを用意します.	
書込手順 2 電源を OFF にして，ジャンパを書き込み用ジャンパピンに取り付け，また Dsub 9 ピンをシリアル通信ポートに接続してください.	
書込手順 3 ジャンパを取り付けた状態で，電源をＯＮにして，書き込み準備ができました.「コマンドプロンプト」を立ち上げ，作成したプログラムのコンパイルおよび「hterm」による H8 への書き込みを行ってください.	

5.2　H8 プログラムのコンパイル

次に，ノートコンピュータから H8 のプログラムの書き込みをします.「スタート」＞「すべてのプログラム」＞「アクセサリ」＞「コマンドプロンプト」を起動してください.

コマンドプロンプト起動後，まず「C:¥h8」フォルダに移動してください．「cd ..」で上層のフォルダに移動することが出来ます．



「C:¥h8」に移動できたならば，プログラムのコンパイルを行います．ここでは「00demo.c」と「00demo.sub」のコンパイルを行うこととします．コンパイルは「compile 00demo」というように，「compile」をタイプし，スペース後に C ファイルおよび Sub ファイルの名前のみをタイプし，実行することです．無事に「LINKAGE EDITOR COMPLETED」が表示されれば，コンパイル終了です．表示されない場合，間違いのあるプログラムの行番号が表示されますので，改善してください．

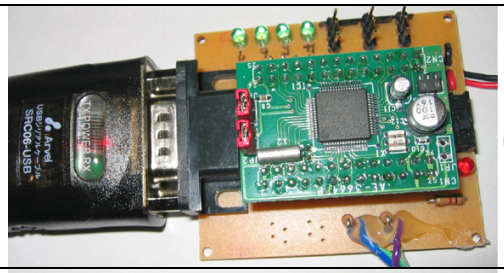
```
C:¥h8>compile 00demo

C:¥h8>ECHO OFF

H8S,H8/300 SERIES C Compiler Ver. 2.0D Evaluation software
Copyright (C) 1994,1996 Hitachi,Ltd.
Licensed Material of Hitachi,Ltd.
Licensed Material of Hitachi Engineering Co.,Ltd.

H SERIES LINKAGE EDITOR Ver. 5.3B Evaluation software
Copyright (C) Hitachi, Ltd.1989,1998
Copyright (C) HITACHI MICROCOMPUTER SYSTEM LTD. 1990,1998
Licensed Material of Hitachi, Ltd.

: OUTPUT 00demo
: PRINT 00demo
: INPUT 3664RES, 00demo
: ;INPUT 3664RES, start, com, 00demo
: LIB c38hn
: ;ROM     (D,X)
: ;START        VECT(0FC40),P,C,D,X,B(0F780),PASM,BASM(0FC74)
: ;START        VECT(0FC40),P,C,D,X,B(0F780),STK(0FF80),PASM,BASM(0FC74)
: ;START        VECT(00000),P,C,D,ASM_P(00034),X,B,ASM_B(0F780),STK(0FF80)
: START P(100)
: EXIT

LINKAGE EDITOR COMPLETED
```

5.3　H8 プログラムの書き込み方法

次に「C:¥h8」フォルダ内にある「writer」フォルダに移動するため，「cd writer」とタイプし，リターンボタンを押してください．そして書き込み用ソフト「hterm」を起動するため，「hterm」タイプし，リターンボタンを押してください．

「Ctrl」ボタンを押しながら「f」を押して「Set Boot Mode and Hit Any key」が表示された後，「Return」ボタンを押してください．そして，「Input Control Program Name:」が表示されたら，「3664.mot」と入力し，リターンを押してください．「Input Program Name:」の表示がありましたら，作成したプログラム名，ここでは「00demo」を入力し，リターンを押してください．プログラムが無事に書き込まれると「`Program Completed.」と表示されますので，「Esc」ボタンを押して，シリアル通信を終了し，プログラムの書き込み終了となります．

5.4　H8（ハードウェア）のプログラム実行準備

内容	写真
実行手順１ 書き込み完了後，まず電源を OFF して，取り付けられている書き込み用ジャンパを取り外し，かつ，シリアル通信用ポートからはずして下さい．	
実行手順２ 全て取り外した後，電源をＯＮにすれば，書き込みしたプログラムが実行されます．