1. 実験目的

シリアル通信機能を用いた機能を持つプログラムを開発する。

2．実験概要

・マイコンとは

　マイコンとはマイクロコンピュータの略称であり、CPU、メモリ、周辺機器などの要素からできている。入力された情報をマイコンの中で処理し、結果を出力し「もの」を制御するために用いられている。温度計を作成したり、おもちゃに使用されたり幅広い分野で用いられている。

・今回の実験概要

　シリアル通信をすることで、送受信可能な制御をするプログラムを作成した。じゃんけん、早押し、チキンレースの３種類のゲームを実装した。

・じゃんけん

SW1をグー、SW2をチョキ、SW3をパーとしてじゃんけんをする。２つのマイコンで手を選んだ後にSW4を押すと勝敗を表示する。

　　・早押し

SW2を押して数秒たつとブザーが鳴る。ブザーが鳴ってから先にSW1を押したほうの勝ち。

　　・チキンレース

あらかじめカウントの値が決まっており、それを交互に減らしていき0にしたほうの負け。SW1で1つだけ減らす、SW2で2つだけ減らす、SW3で減らす値をカウントに反映させる。

1. 実験方法

・開発環境

演習室のPC（Windous 7）、High-performance Embedded Workshop（HEW）、Flash Development Toolkit（FDT）

・使用機器

　・H8/3687Fマイコン

　・LCD

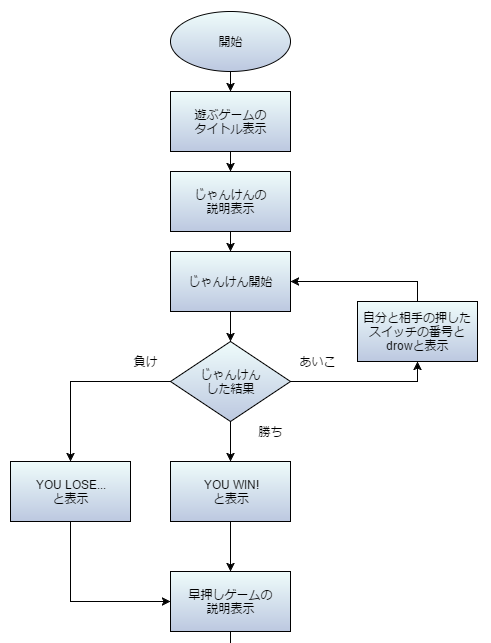
・開発工程

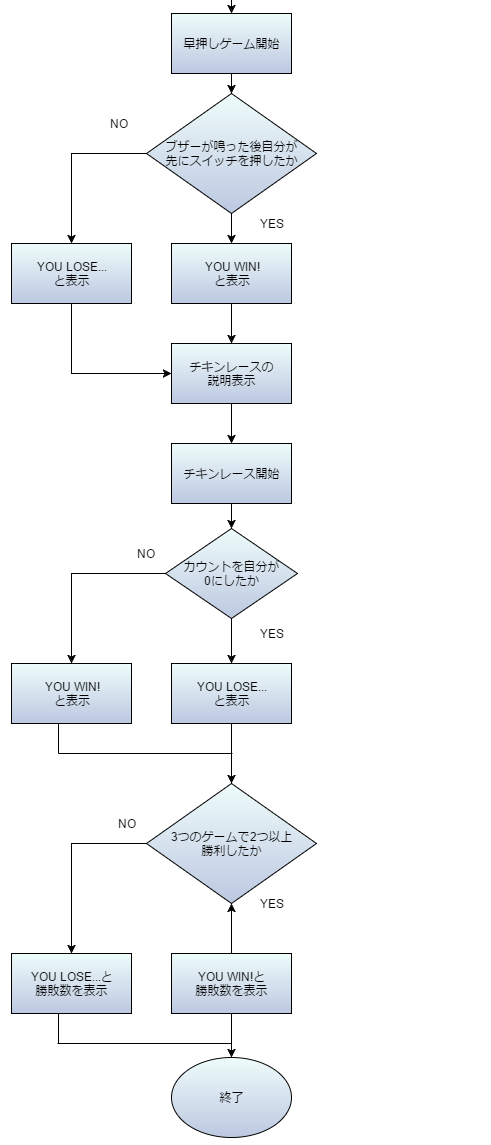
　HEWを用いてｃ言語でソースコードを書いた。その後、FDTを用いてマイコンに書き込みをした。

1. 実験内容

・プログラムのフローチャート

　以下にプログラムのフローチャートを示す。





・ソースコード

　以下、ソースコードを順に解説する。また、私が担当した部分のソースコードは

-start- と -end-で挟んでいる。

冒頭でグローバル変数とマクロの定義をした。①

①のプログラム/////////////////////////

#define WIN 1

#define LOSE 2

#define DRAW 3

int num\_game = 3; //今やってるゲームの指定をする

int num\_win = 0; //自分の勝ち数

int num\_lose = 0; //自分の負け数

int flag\_game = 0; //どのスイッチを押したかのフラグ（値 ＝スイッチの番号）

-----------------------------------------start-------------------------------------------int flag\_timer = 0; //タイマーを起動させるかのフラグ（0＝ 動作しない、1＝動作する）

int flag\_transmission = 0; //通信するかのフラグ（0＝送信できない、1＝送信できる）

int flag\_end = 0; //早押しゲームの時に押したかどうかを 判定させるフラグ

int game\_end = 0; //それぞれのゲームが終わると1になり、 次のゲームにすすむ

int set\_count = 10; //チキンレースの目標回数

int TR\_count = 0; //チキンレースの自分が選択した減少させる回数

------------------------------------------end--------------------------------------------

int myself; //ジャンケンの自分の手

int rival; //ジャンケンの相手の手

-----------------------------------------start-------------------------------------------

int n = 0; //どこのタイミングで割り込みが発生したとしても確実 に送信をしてからゲームを進めるための変数

------------------------------------------end--------------------------------------------

unsigned char RevDT; //受信したデータを格納する変数////////////////////////////////////////

次は空ループを発生させる関数waitである。これは、処理に必要な時間や、LCDに表示するための時間を作るために利用した。②

②のプログラム/////////////////////////

void wait(unsigned int count)

{

unsigned int i, j;

for(i=0; i<=count; i++){

for(j=0; j<=50000; j++){

;

}

}

}

////////////////////////////////////////

　次はゲームのラインナップとじゃんけん、早押し、チキンレースの説明をする関数、indicate\_lineup 、description\_RPS、description\_FP、description\_TRである。これらは文字を表示するだけなので省略する。description\_RPS、description\_FP、description\_TRの3つが私の担当である。

　次は初期化等の処理がまとめられてある関数Initである。

③のプログラム////////////////////////

if(SetSecond == 0)

{

SetMinite--;

SetSecond = 60;

}

setSecond--;

////////////////////////////////////////

　次に、30秒毎にLED0のビットを0にして、LEDのビットを右へシフトさせる。LED0のビットを0にする処理をしないとLED7が点灯し続けてしまうことになる。その後、”second”を0にしてもう1度30秒カウントできるようにした。（④）

④のプログラム////////////////////////

if(second == 30)

{

IO.PDR3.BIT.B0 = 0;

IO.PDR3.BYTE = IO.PDR3.BYTE >> 1;

second = 0;

}

////////////////////////////////////////

また、ビットすべてをシフトしているので1秒毎に点滅するLED7もシフトしてしまう。それに対応するために次の処理を作った。まず、LED7のビットがLED6にシフトするのでLED6のビットが1の場合は、0にする。次にLED7を１秒毎に点滅させる処理をした。LED7を最初から点灯させておくことで、ビットシフトをしても問題なく点滅できるようにした。偶数秒めはLED7のビットは1、奇数秒めはLED7のビットは0となる。30秒毎にビットがシフトしてLED7のビットは0になる。先にシフトさせる処理を入れることでLED7を点灯できるようにした。最後に”count”を0にしてもう1度1秒を測れるようにした。（⑤）

⑤のプログラム//////////////////////////

if(second == 30)

{

IO.PDR3.BIT.B0 = 0;

IO.PDR3.BYTE = IO.PDR3.BYTE >> 1;

second = 0;

}

if(IO.PDR3.BIT.B6 == 1)

{

IO.PDR3.BIT.B6 = 0;

}

if(IO.PDR3.BIT.B7 == 1)

{

IO.PDR3.BIT.B7 = 0;

}

else

{

IO.PDR3.BIT.B7 = 1;

}

count=0;

////////////////////////////////////////

関数SW1、SW2、SW3はそれぞれのスイッチ押したときに点灯させるLEDと時間を設定している。点灯させるLEDは順にLED0・1・7、LED0・1・2・3・7、LED0・1・2・3・4・5・7である。これらの関数は割り込みで動作する。（⑥）

⑥のプログラム//////////////////////////

void SW01(void)

{

IO.PDR3.BYTE = 0x83;

IRR1.BIT.IRRI0 = 0;

SetMinite = 1;

SetSecond = 0;

}

/\*SW2を押したときの処理\*/

void SW02(void)

{

IO.PDR3.BYTE = 0x8F;

IRR1.BIT.IRRI1 = 0;

SetMinite = 2;

SetSecond = 0;

}

/\*SW3を押したときの処理\*/

void SW03(void)

{

IO.PDR3.BYTE = 0xBF;

IRR1.BIT.IRRI2 = 0;

SetMinite = 3;

SetSecond = 0;

}

////////////////////////////////////////

関数SW4は、タイマを起動させる関数である。”flag”は0で初期化してあるためSW4を押さない限りタイマは動作しない。この関数は割り込みで動作する（⑦）

⑦のプログラム//////////////////////////

void SW04(void)

{

flag = 1;

IRR1.BIT.IRRI3 = 0;

IRR2.BIT.IRRTB1 = 0;

}

////////////////////////////////////////

関数resetは、設定した時間が経過した後にいずれかのスイッチを押すと、もう1度時間を設定するところからやり直すことをできるようにする関数である。設定した時間が経過したときになるブザーを停止させ、”flag”を0にしてタイマを起動させないようにさせ、LCDを初期化する。この関数は割り込みで動作する。（⑧）

⑧のプログラム//////////////////////////

void reset()

{

if(IO.PDR2.BIT.B0 == 1)

{

IO.PDR2.BIT.B0 = 0;

flag = 0;

lcd\_init();

}

////////////////////////////////////////

次に割り込みのプログラムの変更部分を以下に載せる。

S10\_TIMERint.c

////////////////////////////////////////

// vector 14 IRQ0

\_\_interrupt(vect=14) void INT\_IRQ0(void)

{

SW01();

reset();

}

// vector 15 IRQ1

\_\_interrupt(vect=15) void INT\_IRQ1(void)

{

SW02();

reset();

}

// vector 16 IRQ2

\_\_interrupt(vect=16) void INT\_IRQ2(void)

{

SW03();

reset();

}

// vector 17 IRQ3

\_\_interrupt(vect=17) void INT\_IRQ3(void)

SW04();

reset();

}

////////////////////////////////////////

1. 実験結果

SW3を押すと、モータがduty比に応じた速度で回転した。SW1を押すと、モータが停止した。SW2を押すと、偶数回押した場合はモータが時計回りに回転し、奇数回押した場合はモータが反時計回りに回転した。

VR1を反時計回りに最大に回したときLCDに表示されるduty比は0％となり、モータは回転しなかった。VR1を徐々に時計回りに回転させていくとそれに応じてLCDに表示されるduty比が大きくなり、モータの回転速度も速くなっていた。VR1を時計周りに最大に回したときLCDに表示されるduty比は100％となり、モータの回転速度も最も速くなった。実際にはVR1を時計回りに回し切らなくてもduty比は100％となり、モータの回転速度も最大となった。

1. 考察

　工夫した点は２つある。１つ目は、LEDの表示の処理である。初期状態でLEDを操作する必要はあるが、ビットシフトとLED6を消す2つの処理でLEDを表示できたことはいいことだと考えた。しかし、若干考え方が難しい気がし、ほかの人の同じ処理をする部分を見たところswitch文で作っていた。こちらはちょっとだけ長くなるが、パッと見てどんな動作をしてどんな結果になるかがわかった。マシンの処理時間を考えれば短いほうがいいと思うが、個人的には見る人にもわかりやすいほうがいいと考える。難しければコメント文を書けばいいと思うが、それでも見てすぐわかるようなものより理解することに時間がかかる。このことは深く考える余地がある。

　2つ目は、リセット機能を付けたことだ。これはスイッチを押すことで処理をするがスイッチを押すたびにこの関数に入っている。割り込みのソースのほうで条件分岐させられれば良かったがそうする方法がわからなかった。ここで分岐できないせいで、無駄な処理が発生してしまっている。ここを改善できればもっといいプログラムになると考えた。