# 1　実験の目的

現在ではほとんどの通信機器が計算機から制御できるようになっている。世界で誰も行なっていない計測を行うためには、自分でソフトウェア開発をすることが必要となる。本実験の目的は、プログラミング言語としては C 言語を用い、計算機から外部機器を制御する基本的な考え方とその方法を学ぶことである。

# 2　実験の方法

C 言語を用いて、課題 1〜11 に取り組んだ。

# 3　実験

## 3.1　実験 1

10 進数を 2 進数に変換するプログラムを作成した。10 進数を 2 で割った余りを配列に代入して、商がゼロになるまで繰り返した。

ソースコード 1: 10 進数を 2 進数へ変換

```c
#include <stdio.h>
#include <math.h>

int main(void){
  int value,n;
  n=1;
  scanf("%d",&value);

  while(value >=  pow(2,n)){
    n+=1;
    continue;
  }
  //printf("%d\n",n);


  int x,result[n],a;
  a=value;
    for(x=1; x<=n; x++){
      result[x]=a % 2;
      a = a/2;
      //printf("%d\n",result[x]);
      if(a==0)
        break;
      else
        continue;
    }

    for(x=n; x>=1; x--){
      printf("%d",result[x]);
    }

  printf("\n");

  return 0;

}
```

## 3.2 実験 2

### 3.2.1 方法

CPU 処理時間の測定を行なった。入力した回数だけ for ループを回すように設計した。関数の引数と for ループの変数の型を long 型から longlong 型にした場合の処理時間の差異も検証した。clock-gettime 関数を使用するため、time.h を include し、コンパイル時に-lrt とリンクをした。

ソースコード 2: CPU 処理時間の計測

```
#include <stdio.h>
#include <time.h>

double measure(long count){
        long i;
        double t;

        struct timespec start_time, end_time;
        clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &start_time);
                for(i=0;i<count;i++)
                ;
        clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &end_time);
        t = end_time.tv_sec - start_time.tv_sec +
           (end_time.tv_nsec - start_time.tv_nsec)/1.0E9;

        return t;
        }

int main(void){
        //input
        long count;

        printf("Enter count num\n");
        scanf("%d",&count);
        printf("Time is %f\n",measure(count));
}
```

### 3.2.2 結果

表 1: for ループ回数および変数の型の違いにおける処理時間

| ループ回数 [回] | long 型 [sec] | long lomg 型 [sec] |
|---|---|---|
| $10^0$ | 0.000003 | 0.000003 |
| $10^1$ | 0.000003 | 0.000003 |
| $10^2$ | 0.000003 | 0.000004 |
| $10^3$ | 0.000005 | 0.000011 |
| $10^4$ | 0.000023 | 0.000083 |
| $10^5$ | 0.000203 | 0.000803 |
| $10^6$ | 0.002026 | 0.008032 |
| $10^7$ | 0.020057 | 0.083567 |
| $10^8$ | 0.203523 | 0.835881 |
| $10^9$ | 2.050905 | 8.041809 |

グラフを図?? に示す。図の直線はループ回数 $10^4$ 以降のデータによって得られた回帰直線である。

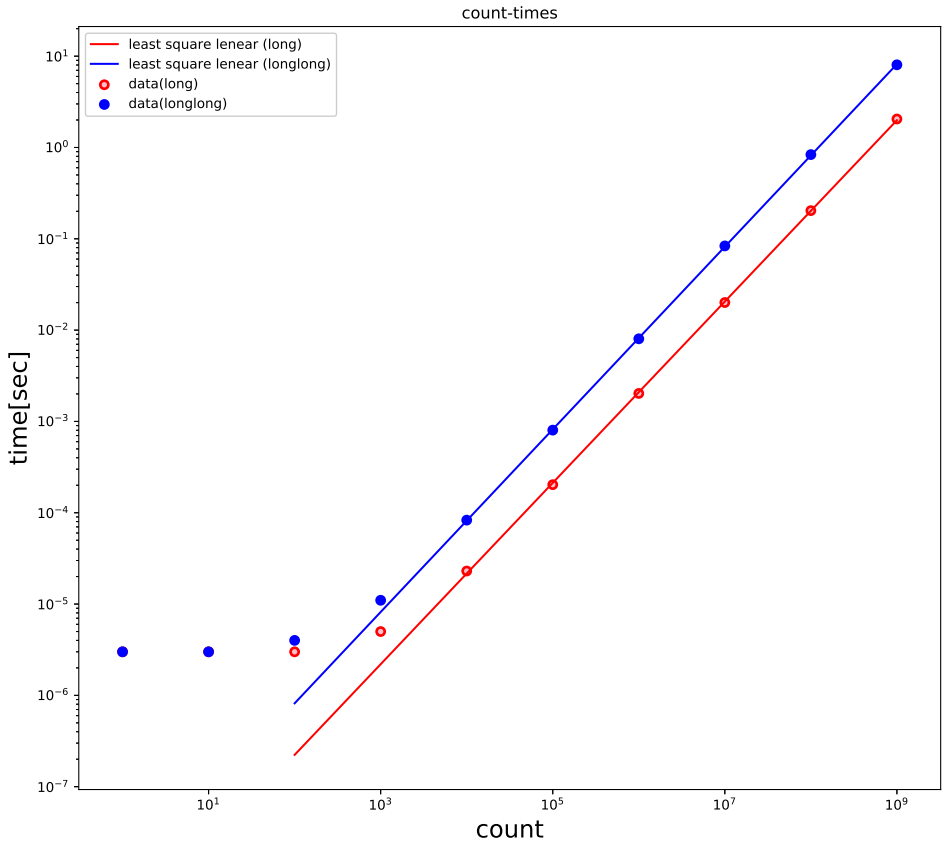


図 1: for ループ回数における処理速度

### 3.2.3 考察

図??より、処理時間がループ回数に比例していることが読み取れる。$10^3$ 未満のデータが直線上にのらない要因は for 文以外に, ある一定の処理時間が生じているためだと考えられる。
for ループ 1 回あたりの時間は回帰直線より long 型は $2.05 \times 10^{-9}[sec]$,longlong 型は $8.04 \times 10^{-9}[sec]$ であると考えられる。実験で用いた CPU のクロック数は 2.00[GHz] であるので 1 回の処理時間は $0.5 \times 10^{-9}[sec]$ である。ゆえに計測の際の処理時間はこの整数倍であるので、long 型は $2.05 \times 10^{-9}[sec]$ で 4 回,longlong 型は $8.04 \times 10^{-9}[sec]$ で 16 回の処理が行われていると考えられる。

## 3.3 実験 3

実験 3〜10 ではデジタル入力ボードを用いて実験を行った。これを制御する関数を用いるため、fbippi.h を include し、コンパイル時に-lgpg2746 とリンクをした。

```
1  /* on_led.c program*/
2  /*LED  no tentou Adake*/
3
4  #include <stdio.h>
5  #include <fbippi.h>
6
7  int main(void)
8  {
9          int dnum=1,isError,nValue;
```

```
        isError = PpiOpen(dnum,0);
        if (isError)
        {
                printf("PpiOpen error; Error No=0x%x\n",isError);
                return;
        }
        /*0x80ha,portAtoCwo syuturyoku mode ni*/
        PpiControl(dnum,FBIPPI_8255_CONTROLLER1,0x80);
        nValue=0xff;
        PpiOutputPort(dnum,FBIPPI_8255_CONTROLLER1,FBIPPI_PORT_A,(unsigned char)nValue);
        PpiClose(dnum); /* Close */
}
```

### 3.4 実験 4

```
/* on_led.c program*/
/*LED  no tentou Adake*/

#include <stdio.h>
#include <fbippi.h>

int safe_scanf_d();
int msleep(long msec);

int main(void)
{
        int dnum=1,isError,nValue,st,count,i;

        printf("switching period(msec):");
        st = safe_scanf_d();
        printf("switching count:");
        count = safe_scanf_d();

        isError = PpiOpen(dnum,0);
        if (isError)
        {
                printf("PpiOpen error; Error No=0x%x\n",isError);
                return;
        }
        /*0x80ha,portAtoCwo syuturyoku mode ni*/
        PpiControl(dnum,FBIPPI_8255_CONTROLLER1,0x80);

        for(i=0;i<count;i++){
        nValue=0xff;
        PpiOutputPort(dnum,FBIPPI_8255_CONTROLLER1,FBIPPI_PORT_A,(unsigned char)nValue);
        PpiOutputPort(dnum,FBIPPI_8255_CONTROLLER1,FBIPPI_PORT_B,(unsigned char)nValue);
        PpiOutputPort(dnum,FBIPPI_8255_CONTROLLER1,FBIPPI_PORT_C,(unsigned char)nValue);
        msleep(st);

        nValue=0x0;
        PpiOutputPort(dnum,FBIPPI_8255_CONTROLLER1,FBIPPI_PORT_A,(unsigned char)nValue);
        PpiOutputPort(dnum,FBIPPI_8255_CONTROLLER1,FBIPPI_PORT_B,(unsigned char)nValue);
        PpiOutputPort(dnum,FBIPPI_8255_CONTROLLER1,FBIPPI_PORT_C,(unsigned char)nValue);
        msleep(st);
        }

        PpiClose(dnum); /* Close */
}

int safe_scanf_d()
{
        int c;
        while(scanf("%d",&c)!=1)
        {
```

```
50          scanf("%*s");
51          }
52          scanf("%*c");
53          return c;
54  }
55
56  int msleep(long msec){
57          return usleep(msec*1000);
58  }
```

### 3.5 実験 5

```
1   /* bin_count24.c program*/
2   /*LED  no tentou Adake*/
3
4   #include <stdio.h>
5   #include <unistd.h>
6   #include <fbippi.h>
7
8   int safe_scanf_d();
9   int msleep(long msec);
10
11  int main(void)
12  {
13          int dnum=1,x,i,j,k,st;
14          printf("switching period(msec):");
15          st = safe_scanf_d();
16
17          while(1)
18          {
19          PpiOpen(dnum,0);
20          PpiControl(dnum,0,0x80);
21
22          i=1;
23          for(x=0;x<24;x++)
24          {
25          i = 2*i;
26          j = i>>8;
27          k = j>>8;
28          PpiOutputPort(dnum,0,FBIPPI_PORT_C,(unsigned char)i);
29          PpiOutputPort(dnum,0,FBIPPI_PORT_B,(unsigned char)j);
30          PpiOutputPort(dnum,0,FBIPPI_PORT_A,(unsigned char)k);
31          msleep(st);
32          }
33          PpiClose(dnum); /* Close */
34          }
35  }
36
37  int safe_scanf_d()
38  {
39          int c;
40          while(scanf("%d",&c)!=1)
41          {
42          scanf("%*s");
43          }
44          scanf("%*c");
45          return c;
46  }
47
48  int msleep(long msec)
49  {
50          return usleep(msec*1000);
51  }
```

### 3.6 実験 6

```c
#include <stdio.h>
#include <fbippi.h>

int main(void)
{
        int dnum=1,j,k,max=256*256*256;
        char i;

        PpiOpen(dnum,0);
        PpiControl(dnum,0,0x80);

        printf("input text number or Esc(end)\n");

        do{
        scanf("%c%*c",&i);
        j = i>>8;
        k = j>>8;

        printf("%3d\n ",i);
        printf("0x%x\n",i);

        if(i==27){
                break;
        };

        PpiOutputPort(dnum,0,FBIPPI_PORT_C,(unsigned char)i);
        PpiOutputPort(dnum,0,FBIPPI_PORT_B,(unsigned char)j);
        PpiOutputPort(dnum,0,FBIPPI_PORT_A,(unsigned char)k);
        } while(i);
        PpiClose(dnum); /* Close */

}
```

### 3.7 実験 7

```c
/*j7.c*/

#include <stdio.h>
#include <fbippi.h>

int safe_scanf_d();
int msleep(long msec);

int main(void)
{
        int dnum=1,st;
        unsigned char Value;

        printf("switching period(msec):");
        st = safe_scanf_d();

        PpiOpen(dnum,0);
        PpiControl(dnum,FBIPPI_8255_CONTROLLER1,0x89);

        do{
        PpiInputPort(dnum,FBIPPI_8255_CONTROLLER1,FBIPPI_PORT_C,&Value);
        PpiOutputPort(dnum,0,FBIPPI_PORT_A,(unsigned char)Value);
        printf("InputData; 0x%x %d %c \n",Value,Value,Value);
        msleep(st);
        }while(Value!=255);

```

```
27          PpiClose(dnum); /* Close */
28
29  }
30
31  int safe_scanf_d()
32  {
33          int c;
34          while(scanf("%d",&c)!=1)
35          {
36          scanf("%*s");
37          }
38          scanf("%*c");
39          return c;
40  }
41
42  int msleep(long msec){
43          return usleep(msec*1000);
44  }
```

## 3.8 実験 8

### 3.8.1 方法

```
1   /*j8.c*/
2
3   #include <stdio.h>
4   #include <fbippi.h>
5
6   int safe_scanf_d();
7   int msleep(long msec);
8   int check(int led,long result);
9
10  int main(void)
11  {
12          int dnum=1,x,i,j,k,st,t,times;
13          printf("switching period(msec):");
14          st = safe_scanf_d();
15          printf("times:");
16          times = safe_scanf_d();
17
18          PpiOpen(dnum,0);
19          PpiControl(dnum,0,0x80);
20
21          int count[24];
22          for(x=1;x<24;x++){
23                  count[x]=0;
24                  }
25
26          for(t=1;t<times;t++){
27
28          i = random();
29  /*
30          j = i>>8;
31          k = j>>8;
32          PpiOutputPort(dnum,0,FBIPPI_PORT_C,(unsigned char)i);
33          PpiOutputPort(dnum,0,FBIPPI_PORT_B,(unsigned char)j);
34          PpiOutputPort(dnum,0,FBIPPI_PORT_A,(unsigned char)k);
35  */
36          printf("InputData; 0x%x %d %c \n",i,i,i);
37
38          for(x=1;x<24;x++){
39                  count[x] += check(x,(int)i);
40                  }
41
```

```
42          msleep(st);
43          }
44
45          PpiClose(dnum); /* Close */
46
47          printf("*****************************************\n");
48          for(x=1;x<24;x++){
49                  printf("%d\n",count[x]);
50                  }
51
52  }
53
54  int safe_scanf_d()
55  {
56          int c;
57          while(scanf("%d",&c)!=1)
58          {
59          scanf("%*s");
60          }
61          scanf("%*c");
62          return c;
63  }
64
65  int msleep(long msec){
66          return usleep(msec*1000);
67  }
68
69  int check(int led,long result){
70          int i, a[24], b, c;
71          a[0]=1;
72          for(i=1;i<24;i++){
73                  a[i]=a[i-1]<<1;
74                  }
75
76          b = a[led]&result;
77          if(b==0){
78                  c = 0;
79          }else{
80                  c = 1;
81          }
82          return c;
83  }
```

### 3.8.2 結果

表 2: 総点灯回数における各 LED の点灯回数

| 総点灯回数 [回] | 平均回数 [回] | 標準偏差 [回] | 出現確率 | 誤差率 |
|---|---|---|---|---|
| $10^2$ | 49.2 | 5.1 | 0.492 | 0.051 |
| $10^3$ | 49.2 | 5.1 | 0.492 | 0.051 |
| $10^4$ | 4984.3 | 49.4 | 0.498 | 0.049 |
| $10^5$ | 50061.5 | 179.5 | 0.500 | 0.018 |

### 3.8.3 考察

総点灯回数が増加するごとに出現確率が 0.5 に近づき, 誤差率が減少していくことが読み取れる。

## 3.9 実験 9

### 3.9.1 方法

```
1
2  #include <stdio.h>
3  #include <fbippi.h>
4  #include <unistd.h>
5
6  int safe_scanf_d();
7
8  int main(void)
9  {
10         int dnum=1,n,count,i,c;
11         int b[8]={1,3,2,6,4,12,8,9};
12
13         printf("Enter switching period (usec):");
14         c = safe_scanf_d();
15         printf("Enter step number:");
16         n = safe_scanf_d();
17
18         PpiOpen(dnum,0);
19         PpiControl(dnum,FBIPPI_8255_CONTROLLER1,0x80);
20
21         for(i=0;i<8;i++)
22         {
23                 PpiOutputPort(dnum,FBIPPI_8255_CONTROLLER1,FBIPPI_PORT_A,b[i]);
24                 usleep(10000);
25         }
26
27         printf("This is starting point. Waiting 3 sec.....\n");
28         sleep(3);
29         count=0;
30         while(count<n)
31         {
32                 for(i=0;i<8;i++)
33                 {
34                         PpiOutputPort(dnum,FBIPPI_8255_CONTROLLER1,FBIPPI_PORT_A,b[i]);
35                         count++;
36                 if(count>=n) break;
37                 printf(" output=0x%02x count=%d\n",b[i],count);
38                 usleep(c);
39                 }
40         }
41         PpiClose(dnum); /* Close */
42 }
43
44 int safe_scanf_d()
45 {
46         int c;
47         while(scanf("%d",&c)!=1)
48         {
49         scanf("%*s");
50         }
51         scanf("%*c");
52         return c;
53 }
```

表 3: ステップモータの回転

| | 360 度回転 [step] | 1step[deg] | 最小間隔 [$\mu s$] |
|---|---|---|---|
| 1 相励磁 | 48 | 0.133 | 6300 |
| 2 相励磁 | 48 | 0.133 | 4600 |
| 1 − 2 相励磁 | 96 | 0.267 | 2400 |

#### 3.9.2 結果

#### 3.9.3 考察

動作原理より 1-2 相励磁は、1 相励磁に比べ 1step あたりの回転角度が小さいことが分かる。実験によりこのことが確かめられた。

### 3.10 実験 10

```
#include <stdio.h>
#include <fbippi.h>
#include <unistd.h>
#include <pthread.h>

int flag = 1;
char no = '1';
int change = 0;

void *kybd();
int msleep(long count);

int main(void){

        int dnum=1,n,i,c;
        int b[8]={1,3,2,6,4,12,8,9};

        PpiOpen(dnum,0);
        PpiControl(dnum,FBIPPI_8255_CONTROLLER1,0x80);

        long speed, turn, count;
        pthread_t t1;
        pthread_create(&t1,NULL,kybd,NULL);
        speed = 400;
        count=0;

        for(i=0;i<8;i++)
        {
                PpiOutputPort(dnum,FBIPPI_8255_CONTROLLER1,FBIPPI_PORT_A,b[i]);
                usleep(10000);
        }

        printf("This is starting point. Waiting 3 sec.....\n");
        sleep(3);

        while(flag)
        {
                if(change)
                {
                        switch(no)
                        {
                        case 'd':
                        speed*=2;
                        break;
```

```
45
46                              case 'a':
47                              speed/=2;
48                              break;
49
50                              case 'c':
51                              turn=1;
52                              break;
53
54                              case 'w':
55                              turn=-1;
56                              break;
57                              }
58                      change=0;
59                      }
60              if(speed<10) speed*=2;
61
62              if(turn==1){
63                      i++;
64              } else{
65                      i--;
66              }
67
68              if(i==8)i=0;
69              if(i==-1)i=8;
70
71              PpiOutputPort(dnum,FBIPPI_8255_CONTROLLER1,FBIPPI_PORT_A,b[i]);
72
73              count++;
74              msleep(speed);
75              printf("speed=%dmsec count=%d\n",speed,count);
76              }
77              pthread_join(t1,NULL);
78
79              PpiClose(dnum);
80      }
81
82      void *kybd(){
83              char a;
84              while(flag){
85                      scanf("%c%*c",&a);
86                      if(a=='q'){
87                              flag=0;
88                      } else {
89                              no = a;
90                              change = 1;
91                      }
92              }
93      }
94
95      int msleep(long msec){
96              return usleep(msec*1000);
97      }
```

## 3.11 実験 11

```
1   #include <stdio.h>
2   #include <string.h>
3   #include <unistd.h>
4   #include <pcigpib.h>
5
6   int msleep(long msec);
7   int safe_scanf_d();
8
```

```
9  int main(void){
10   int adrstbl[2] = {10,-1};
11   char *init = "Z";
12   char *cmd = "F1,PR3,R0,M0,H1";
13   char *trigger = "E";
14   char *R[] = {"R2","R3","R4","R5","R6","R7"};
15   char recbuf[100];
16   long i,delay,len=100;
17
18   printf("delay(msec)=");
19   delay=safe_scanf_d();
20
21   PciGpibExInitBoard(0,0);
22   PciGpibExSetIfc(0,1);
23   PciGpibExSetRen(0);
24   PciGpibExSendData(0,adrstbl,strlen(init),init,0);
25   PciGpibExSendData(0,adrstbl,strlen(cmd),cmd,0);
26
27   for(i=0;i<6;i++){
28     PciGpibExSendData(0,adrstbl,strlen(R[i]),R[i],0);
29     PciGpibExSendData(0,adrstbl,strlen(trigger),trigger,0);
30     msleep(delay);
31     PciGpibExRecvData(0,adrstbl,&len,recbuf,0);
32     recbuf[len]='\0';
33     printf("%s\n",recbuf);
34   }
35   PciGpibExFinishBoard(0);
36 }
37
38 int safe_scanf_d()
39 {
40         int c;
41         while(scanf("%d",&c)!=1)
42         {
43         scanf("%*s");
44         }
45         scanf("%*c");
46         return c;
47 }
48
49 int msleep(long msec)
50 {
51         return usleep(msec*1000);
52 }
```

12 行目の H0 を H1 に帰ると、ヘッダが現れた。

# 4　全体の感想

例題から取り組み始めたため、課題を全て終了することはできなかったが、C 言語および計算機制御について、深く学ぶことができた。私は普段、スクリプト言語である Python を使うことが多い。そのためセミコロン等のつけ忘れなどによる文法エラー、コンパイル時のミス、型宣言など、基本的な点でつまづくことが多かったように思う。今回の実験を通してプログラミングの基礎を学び直せたのは非常によかった。また個人的に C 言語と各言語との違いを調べルコとで、C 言語が PHP や JavaScript などに大きな影響を与えていることも学べた。