1　目的

H8 マイコンの A/D、D/A 変換器について理解を深め、それらを用いて割り込みによるデータ入出力プログラムを作成する。

2　原理

D/A、A/D 変換の原理については前期実験の実験 4「H8 マイコンの A/D、D/A 変換を用いた音声録音再生実験」を参照のこと。

3　使用機器

1. H8 マイコン (メーカ:Beyond The River, 型番:H8-3052)
2. USB ケーブル
3. パーソナルコンピュータ (メーカ:Dell, 型番:H8-3052)

4　実験方法

D/A、A/D に関わるファイル及び関数定義などは、前期実験の実験 4「H8 マイコンの A/D、D/A 変換を用いた音声録音再生実験」を参照のこと。

実験準備として、「/home/class/j3/jikken/kouki/no3」以下のファイルを各自コピーした上で、以下の課題についてプログラム (rec.c) を作成し動作確認を行う。

4.1　音声の記録再生 1

H8 マイコンボード上の「*」キーを押すと録音モードとなり、「#」キーを押すと再生モードとなるプログラム (ad-da.c) を作成しなさい。スピーカーは録音モード時にマイクとして機能し、再生モード時にはスピーカとして機能するようにしなさい。マイクに向かって音声を入力すると音声データが保存され、その後、スピーカにより再生が可能になる。

4.2　音声の記録再生 2

H8 マイコンボード上の録音モード「*」キーを押すと、上段の LCD 表示が「Push * or # key」、下段の LCD 表示が「1 」、「12 」、…、「12345」になるようにする。また、再生モード「#」キーを押すと、上段の LCD 表示が「Push *

or # key」、下段の LCD 表示が「1 」、「12 」、…、「12345」、「Push * or # key」になるように作成しなさい。なお、「Now Playing…」、「Now Sampling…」は表示しない。また、1、2、3、4、5 の表示は、定義された TIME を均等時間になるようにする。

4.3 音声の録音再生 3

H8 マイコンボード上の録音モード「1」キーを押すと、上段の LCD 表示が「Push 1 or 2 key」、下段の LCD 表示が「Now Sampling…」となるようにする。再生モード「2」キーを押すと、上段の LCD 表示が「Push 1 or 2 key」、下段の LCD 表示が「Now Playing…」になるように作成しなさい。つまり、前節での録音モード「*」キーを「1」キーに、再生モード「#」キーを「2」に切り替えなさい。なお、駆動時間は TIME とする。

4.4 音声の記録再生・逆再生

H8 マイコンボード上の録音モード「*」キーを押すと、上段の LCD 表示が「Push * or 5 key」、下段の LCD 表示が「Now Sampling…」を表示して TIME 時間に録音する。逆再生モード「5」キーを押すと、上段の LCD 表示が「Push * or 5 key」、下段の LCD 表示が「Now Inverse…」になるようにして、逆再生モードが TIME=12000~ 0 に設定して逆再生するプログラムを作成しなさい。

4.5 音声の記録再生・統合

H8 マイコンボード上の録音モード「*」キーを押すと、上段の LCD 表示が「Push *, # or 5」、下段の LCD 表示が「Now Sampling…」を表示して TIME 時間に録音する。再生モード「#」キーを押すと、上段の LCD 表示が「Push *, # or 5」、下段の LCD 表示が「Now Playing…」を表示して TIME 時間に再生する。逆再生モード「5」キーを押すと、上段の LCD 表示が「Push *, # or 5」、下段の LCD 表示が「Now Inverse…」を表示して TIME 時間に逆再生するプログラムを作成しなさい。

5 実験結果

5.1 音声の記録再生 1

要件を満たす動作をするプログラムを作成した。ソースファイルをリスト 1 に、Makefile をリスト 2 に示す。

5.2　音声の記録再生 2

要件を満たす動作をするプログラムを作成した。ソースファイルをリスト 3 に示す。Makefile はリスト 2 中の「ad-da」を「no2」に置換したものである。

5.3　音声の録音再生 3

要件を満たす動作をするプログラムを作成した。ソースファイルをリスト 4 に示す。Makefile はリスト 2 中の「ad-da」を「no3」に置換したものである。

5.4　音声の記録再生・逆再生

要件を満たす動作をするプログラムを作成した。ソースファイルをリスト 5 に示す。Makefile はリスト 2 中の「ad-da」を「no4」に置換したものである。

5.5　音声の記録再生・統合

要件を満たす動作をするプログラムを作成した。ソースファイルをリスト 6 に示す。Makefile はリスト 2 中の「ad-da」を「no5」に置換したものである。

6　検討課題

・検討課題 1

割り込みのサンプリング間隔を変更すると、録音・再生する音の情報にどのような影響を与えるか、考察せよ。音情報の変化と再現性などをふまえて考察すること。

サンプリング間隔を短くすると、音質は上がるがある容量で録音できる時間が短くなる。長くした場合は音質は下がるが、録音できる時間は長くなる。音源の周波数が高いほど、短いサンプリング間隔が要求される。

・検討課題 2

A/D のサンプリング間隔と D/A の再生間隔を一致させなかった場合、出力の音情報はどういう変化をするのか、A/D に対して D/A を短くする/長くするの 2 つの観点から検討しなさい。

短くした場合は周波数が高くなるために音が高く聞こえる。長くしたその逆で低く聞こえる。

リスト 1: ad-da.c

```
#include "h8-3052-iodef.h"

int main(void)
{
  unsigned char cf, key_data;

  P9DDR = 0x30;  /* ポートの初期化9(P95-を出力に設定P94) */
  P9DR = 0x30;

  P6DDR &= ~0x07;  /* P60,1,2 入力   */
  PADDR |= 0x0f;   /* PA0,1,2,3 出力 */

  while(1) {
    key_data = 0;

    //key 1,2,3
    PADR = 0x07; // PA3 = L
    cf = P6DR;   // データ入力
    cf = ~cf;    // の反転cf
    cf &= 0x07;  // P60のみ見る,1,2
    switch(cf) {
    case 1 : key_data = '1'; break;
    case 2 : key_data = '2'; break;
    case 4 : key_data = '3'; break;
    }

    //key 4,5,6
    PADR = 0x0b;
    cf = P6DR;
    cf = ~cf;
    cf &= 0x07;
    switch(cf) {
    case 1 : key_data = '4'; break;
    case 2 : key_data = '5'; break;
    case 4 : key_data = '6'; break;
    }

    //key 7,8,9
    PADR = 0x0b; /* This is a mistake code. */
    cf = P6DR;
    cf = ~cf;
    cf &= 0x07;
    switch(cf) {
    case 1 : key_data = '7'; break;
    case 2 : key_data = '8'; break;
    case 4 : key_data = '9'; break;
    }

    //key *,0,#
    PADR = 0x0e;
    cf = P6DR;
    cf = ~cf;
    cf &= 0x07;
    switch(cf) {
    case 1 : key_data = '*'; break;
    case 2 : key_data = '0'; break;
    case 4 : key_data = '#'; break;
    }

    if(key_data == '1' ) {
      P9DR = 0x20;  /* D1赤点灯(), D2緑消灯() */
    }
    if(key_data == '0'){
      P9DR = 0x30;  /* D1赤消灯(), D2緑消灯() */
    }

  }

}
```

リスト 2: Makefile

```
# H8/3052 の雛型  Makefile
# 手順 1. 必要な設定を変更して、違うファイル名で保存する例： (make-test)
#   TARGET = , SOURCE_C = , SOURCE_ASM = を指定する
# リモートデバッキングのときは、  GDBREMOTE_DBG = true とする
# その他は通常、変更の必要はない
# 手順 2. make -f 名makefile で make する例： (make -f make-test)

# 生成するファイルとソースファイルの指定
# 1. 生成するオブジェクトのファイル名を指定
TARGET = no1.mot
# 2. 生成に必要なのファイル名を空白で区切って並べるC
SOURCE_C = no1.c
# 3. 生成に必要なアセンブラのファイル名を空白で区切って並べる
# スタートアップルーチンは除く()
SOURCE_ASM =

# 生成するオブジェクトの種類を指定
# ※の項目は通常変更する必要がない()
#
# 1. によるリモートデバッキング指定GDB
# true : 指定する    その他：指定しない
REMOTE_DBG =

# 2. 上デバッグまたは化指定RAMROM ※
# ram : 上で実行RAM rom : 化ROM
ON_RAM = ram

# 3. 使用領域の指定RAM ※
# ：化→プログラムとスタックは外部を使用extRAMRAM
# 化→スタックは外部      ROMRAM
# ：化→プログラムとスタックは内部を使用intRAMRAM
# 化→スタックは内部      ROMRAM
# 指定なし：化→プログラムは外部、スタック変更なしRAMRAM
#   化→スタックは外部  ROMRAM
RAM_CAP = ext

# 4. によるデバッグを行うかどうかの指定GDB ※
USE_GDB = true

#
# パスの設定
#
CMD_PATH = /usr/local/bin
LIB_PATH = /home/class/common/H8/lib

#
# クロスコンパイラ関係
#
CC = $(CMD_PATH)/h8300-hms-gcc
LD = $(CMD_PATH)/h8300-hms-ld
OBJCOPY = $(CMD_PATH)/h8300-hms-objcopy
SIZE = $(CMD_PATH)/h8300-hms-size

#
# ターゲット指定
#
TARGET_COFF = $(TARGET:.mot=.coff)
MAP_FILE = $(TARGET:.mot=.map)

#
# 出力フォーマット
# binary : binary, srec : Motorola S record, ihex : Intel Hex
#
OUTPUT_FORMAT = -O srec --srec-forceS3

#
# コンパイラオプション
#
# インクルードディレクトリの追加("*****.h"指定のみ有効)
INCLUDES = -I/home/class/common/H8/include
# コンパイラオプションの指定
# -：mhH8/300シリーズ指定H
```

```
# -：条件分岐コードの最適化mrelax
# -：型変数のビット数指定mint32int
# -：の最適化レベルの指定O2gcc
# -：コンパイル時の警告メッセージの選択Wall全て()
CFLAGS = -mh -mrelax -mint32 -O2 $(INCLUDES) -Wall

#
# 指定に合わせたスタートアップルーチンとリンカスクリプトの選択
#
ifeq ($(REMOTE_DBG), true)
  USE_GDB = true
  ON_RAM = ram
  RAM_CAP =
endif

ifeq ($(USE_GDB), true)
  CFLAGS := $(CFLAGS) -g
endif

ifeq ($(ON_RAM), ram)
  LDSCRIPT = $(LIB_PATH)/h8-3052-ram.x
  STARTUP = $(LIB_PATH)/ramcrt.s
  ifeq ($(RAM_CAP), int)
        LDSCRIPT = $(LIB_PATH)/h8-3052-ram8k.x
    STARTUP = $(LIB_PATH)/ramcrt-8k.s
  endif
  ifeq ($(RAM_CAP), ext)
    LDSCRIPT = $(LIB_PATH)/h8-3052-ram.x
    STARTUP = $(LIB_PATH)/ramcrt-ext.s
  endif
  ifeq ($(REMOTE_DBG), true)
    LDSCRIPT = $(LIB_PATH)/h8-3052-ram-dbg.x
    STARTUP = $(LIB_PATH)/ramcrt-dbg.s
  endif
else
  ifeq ($(RAM_CAP), int)
    LDSCRIPT = $(LIB_PATH)/h8-3052-rom8k.x
    STARTUP = $(LIB_PATH)/romcrt-8k.s
  else
        LDSCRIPT = $(LIB_PATH)/h8-3052-rom.x
    STARTUP = $(LIB_PATH)/romcrt-ext.s
  endif
endif

#
# リンク時のコンパイラオプションの指定
# -T ：リンカスクリプトファイルの指定filename
# -：標準のスタートアップを使用しないnostartfiles
# -Wlパラメータ…：リンカに渡すパラメータ指定,,
#   -Map ：メモリマップをに出力mapfilenamemapfilename
LDFLAGS = -T $(LDSCRIPT) -nostartfiles -Wl,-Map,$(MAP_FILE)

#
# オブジェクトの指定
#
OBJ = $(STARTUP:.s=.o) $(SOURCE_C:.c=.o) $(SOURCE_ASM:.s=.o)

#
# サフィックスルール適用の拡張子指定
#
.SUFFIXES: .c .s .o

#
# ルール
#
$(TARGET) : $(TARGET_COFF)
  $(OBJCOPY) -v $(OUTPUT_FORMAT) $(TARGET_COFF) $(TARGET)

$(TARGET_COFF) : $(OBJ)
  $(CC) $(CFLAGS) $(LDFLAGS) $(OBJ) -o $(TARGET_COFF)
  $(SIZE) -Ax $(TARGET_COFF)

clean :
  rm -f *.o $(TARGET) $(TARGET_COFF) $(MAP_FILE)
```

```
147
148 #
149 # サフィックスルール
150 #
151 .c.o:
152   $(CC) -c $(CFLAGS) $<
153 .s.o:
154   $(CC) -c $(CFLAGS) $<
```

リスト 3: 課題 2 のソース

```
1  #include "h8-3052-iodef.h"
2  #include "lcd.h"
3
4  int main(void)
5  {
6    unsigned char cf, key_data;
7    volatile char hyouzi_you = ' ';
8
9    P9DDR = 0x30;  /* ポートの初期化9(P95-を出力に設定P94) */
10
11   P6DDR &= ~0x07;  /* P60,1,2 入力   */
12   PADDR |= 0x0f;   /* PA0,1,2,3 出力 */
13
14   lcd_init();
15   lcd_cursor( 0, 0 );
16
17   while(1) {
18     key_data = 0;
19
20     //key 1,2,3
21     PADR = 0x07; // PA3 = L
22     cf = P6DR;   // データ入力
23     cf = ~cf;    // の反転cf
24     cf &= 0x07;  // P60のみ見る,1,2
25     switch(cf) {
26     case 1 : key_data = '1'; break;
27     case 2 : key_data = '2'; break;
28     case 4 : key_data = '3'; break;
29     }
30
31     //key 4,5,6
32     PADR = 0x0b;
33     cf = P6DR;
34     cf = ~cf;
35     cf &= 0x07;
36     switch(cf) {
37     case 1 : key_data = '4'; break;
38     case 2 : key_data = '5'; break;
39     case 4 : key_data = '6'; break;
40     }
41
42     //key 7,8,9
43     PADR = 0x0d; /* This is a mistake code. */
44     cf = P6DR;
45     cf = ~cf;
46     cf &= 0x07;
47     switch(cf) {
48     case 1 : key_data = '7'; break;
49     case 2 : key_data = '8'; break;
50     case 4 : key_data = '9'; break;
51     }
52
53
54     //key *,0,#
55     PADR = 0x0e;
56     cf = P6DR;
57     cf = ~cf;
58     cf &= 0x07;
59     switch(cf) {
60     case 1 : key_data = '*'; break;
61     case 2 : key_data = '0'; break;
```

```
62      case 4 : key_data = '#'; break;
63      }
64
65      if( key_data != 0 )
66        hyouzi_you = key_data;
67
68      lcd_cursor(0,0);
69      lcd_printch( hyouzi_you );
70
71    }
72
73  }
```

リスト 4: 課題 3 のソース

```
1   #include "h8-3052-iodef.h"
2   #include "lcd.h"
3
4   #define LOOP (13200)
5
6   void delay( unsigned long int ms )
7   {
8     unsigned long int i, j;
9
10    for( i = 0; i < ms; i++ )
11      for( j = 0; j < LOOP; j++);
12  }
13
14  int main(void)
15  {
16    unsigned char cf, key_data;
17    char prev_pressed_key = 0;
18
19    P9DDR = 0x30;  /* ポートの初期化9(P95-を出力に設定P94) */
20
21    P6DDR &= ~0x07;  /* P60,1,2 入力   */
22    PADDR |= 0x0f;   /* PA0,1,2,3 出力 */
23
24    int pos = 0;
25
26    lcd_init();
27    lcd_clear();
28
29    while(1) {
30      key_data = 0;
31
32      //key 1,2,3
33      PADR = 0x07; // PA3 = L
34      cf = P6DR;   // データ入力
35      cf = ~cf;    // の反転cf
36      cf &= 0x07;  // P60のみ見る,1,2
37      switch(cf) {
38      case 1 : key_data = '1'; break;
39      case 2 : key_data = '2'; break;
40      case 4 : key_data = '3'; break;
41      }
42
43      //key 4,5,6
44      PADR = 0x0b;
45      cf = P6DR;
46      cf = ~cf;
47      cf &= 0x07;
48      switch(cf) {
49      case 1 : key_data = '4'; break;
50      case 2 : key_data = '5'; break;
51      case 4 : key_data = '6'; break;
52      }
53
54      //key 7,8,9
55      PADR = 0x0d; /* This is a mistake code. */
56      cf = P6DR;
57      cf = ~cf;
```

```
58      cf &= 0x07;
59      switch(cf) {
60      case 1 : key_data = '7'; break;
61      case 2 : key_data = '8'; break;
62      case 4 : key_data = '9'; break;
63      }
64
65      //key *,0,#
66      PADR = 0x0e;
67      cf = P6DR;
68      cf = ~cf;
69      cf &= 0x07;
70      switch(cf) {
71      case 1 : key_data = '*'; break;
72      case 2 : key_data = '0'; break;
73      case 4 : key_data = '#'; break;
74      }
75
76      if( key_data != prev_pressed_key ){
77        delay( 15 );
78        if( key_data != 0 ){
79          lcd_cursor( pos % 16, pos / 16 );
80          lcd_printch( key_data );
81          pos++;
82        }
83      }
84      prev_pressed_key = key_data;
85    }
86
87  }
```

リスト 5: 課題 4 のソース

```
1   #include "h8-3052-iodef.h"
2   #include "lcd.h"
3
4   #define LOOP (13200)
5
6   void delay( unsigned long int ms )
7   {
8     unsigned long int i, j;
9
10    for( i = 0; i < ms; i++ )
11      for( j = 0; j < LOOP; j++);
12  }
13
14  int main(void)
15  {
16    unsigned char cf, key_data;
17    char prev_pressed_key = 0;
18
19    P9DDR = 0x30;  /* ポートの初期化9(P95-を出力に設定P94) */
20
21    P6DDR &= ~0x07;  /* P60,1,2 入力    */
22    PADDR |= 0x0f;   /* PA0,1,2,3 出力 */
23
24    int pos = 0;
25    int key_count = 10;
26    int alpha = 0;
27
28    lcd_init();
29    lcd_clear();
30
31    while(1) {
32      key_data = 0;
33
34      //key 1,2,3
35      PADR = 0x07; // PA3 = L
36      cf = P6DR;   // データ入力
37      cf = ~cf;    // の反転cf
38      cf &= 0x07;  // P60のみ見る,1,2
39      switch(cf) {
```

```
    case 1 : key_data = '1'; break;
    case 2 : key_data = '2'; break;
    case 4 : key_data = '3'; break;
    }

    //key 4,5,6
    PADR = 0x0b;
    cf = P6DR;
    cf = ~cf;
    cf &= 0x07;
    switch(cf) {
    case 1 : key_data = '4'; break;
    case 2 : key_data = '5'; break;
    case 4 : key_data = '6'; break;
    }

    //key 7,8,9
    PADR = 0x0d; /* This is a mistake code. */
    cf = P6DR;
    cf = ~cf;
    cf &= 0x07;
    switch(cf) {
    case 1 : key_data = '7'; break;
    case 2 : key_data = '8'; break;
    case 4 : key_data = '9'; break;
    }

    //key *,0,#
    PADR = 0x0e;
    cf = P6DR;
    cf = ~cf;
    cf &= 0x07;
    switch(cf) {
    case 1 : key_data = '*'; break;
    case 2 : key_data = '0'; break;
    case 4 : key_data = '#'; break;
    }


    if( key_data != prev_pressed_key ){
      delay( 15 );
      if( key_data != 0 ){
        alpha += key_count * (key_data - '0');
        key_count/=10;
        if( key_count == 0 ){
          lcd_cursor( pos % 16, pos / 16 );
          if( alpha >= 0 && alpha <= 25 ){
            lcd_printch( 'a' + alpha );
            key_count = 10;
            pos++;
            alpha = 0;
          }
        }
      }
    }
    prev_pressed_key = key_data;
  }

}
```