

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 (メーカー: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 (メーカー: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、出力の音情報はどのような変化をするのか、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: 課題 1 のソース

```
1  #include "h8-3052-iodef.h"
2  #include "h8-3052-int.h"
3  #include "lcd.h"
4  #include "ad.h"
5  #include "da.h"
6  #include "timer.h"
7
8  #define BUFSIZE      30 /* バッファの大きさ(kB) */
9  #define SAMPLINGTIME 100 /* 録音再生時のサンプリング周期(us) */
10 #define SAMPLE      0 /* 動作選択値録音() */
11 #define PLAY        1 /* 動作選択値再生() */
12 #define NOSELECT     -1 /* 動作選択値未選択() */
13
14 volatile unsigned char databuf[(unsigned long)BUFSIZE * 1024];
15 volatile unsigned long bufptr;
16 volatile int play_mode;
17
18 int      main(void);
19 unsigned char menu(void);
20 void     sample_replay(int mode);
21 void     int_imia0(void);
22
23 int main(void)
24 {
25     /* キー入力情報を取得するための変数を宣言する */
26     unsigned char key_data;
27
28     ROMEMU(); /* エミュレーションをROMON */
29     lcd_init(); /* LCD の初期化 */
30     ad_init(); /* A/D 変換器の初期化 */
31     da_init(); /* D/A 変換器の初期化 */
32     timer_init(); /* タイマの初期化 */
33
34     /* タイマチャネルの割り込み間隔(0) */
35     timer_set(0, SAMPLINGTIME);
36
37     /* ここにキー入力取得のためのポートの初期化を記述する */
38     P6DDR &= ~0x07; /* P60,1,2 入力 */
39     PADDR |= 0x0f; /* PA0,1,2,3 出力 */
40
41     while (1) {
42         play_mode = NOSELECT;
43         key_data = menu(); /* メニューを選ぶ */
44
45         /* 録音キーが押されたら、再生キーが押されたらSAMPLEPLAY */
46         /* をに格納する処理を記述するplay_mode */
47         switch( key_data ){
48             case ' ':
49                 play_mode = SAMPLE;
50                 break;
51
52             case '#':
53                 play_mode = PLAY;
54                 break;
```

```

55     }
56
57     /* キー入力されていれば録音再生の関数を呼び出す処理を記述する */
58     if(play_mode != NOSELECT){
59         sample_replay( play_mode );
60     }
61 }
62
63 return 1;
64 }
65
66 unsigned char menu(void)
67 /* LCD にメニューを書いて動作を選択するための関数 */
68 /* 戻り値は入力キー情報 */
69 {
70     /* キー入力取得のための変数を宣言する */
71     unsigned char cf, key_data;
72
73     /* キー入力判定用変数の初期化 */
74     cf = 0;
75     key_data = 0;
76
77     while (cf == 0 ){ /* キー入力するまでループする */
78         //key 0
79         PADR = 0x0e;
80         cf = P6DR;
81         cf = ~cf;
82         cf &= 0x07;
83         switch(cf) {
84             case 1 : key_data = '*'; break;
85             case 4 : key_data = '#'; break;
86         }
87     }
88
89     /* 入力されたキーの情報を返す */
90     return key_data;
91 }
92
93 void sample_replay(int mode)
94 /* 録音または再生を行う関数 */
95 /* mode: PLAY, SAMPLE */
96 {
97     if (mode == PLAY){ /* 再生モードの処理 */
98         /* ここにスピーカをスピーカとして使用する命令を記述する */
99         speaker_switch(SPEAKER);
100     }
101     if (mode == SAMPLE){ /* 録音モードの処理 */
102         /* ここにスピーカをマイクとして使用する命令を記述する */
103         speaker_switch(MIC);
104     }
105     bufptr = 0; /* バッファポインタを初期化 */
106     timer_start(0); /* サンプリングタイマチャネルのスタート(0) */
107     ENINT(); /* 割り込み許可CPU */
108
109     unsigned long temp;
110     while (bufptr < ((unsigned long)BUFSIZE * 1024));
111
112     /* バッファが一杯になるまで実行 */
113     speaker_switch(MIC); /* スピーカーオフ */
114     timer_stop(0); /* タイマのストップ */
115 }
116
117 #pragma interrupt
118 void int_imia0(void)
119 /* 録音・再生用のタイマ割り込みハンドラ */
120 /* プレイモードによってデータの格納か出力を行う */
121 {
122     if (play_mode == SAMPLE){
123         /* ここに録音のときの処理を記述する以下のコメントを参照のこと */
124         ad_start( 0, 0 ); /* ◎A/D変換スタート */
125         while(ADSTATUS() == 0); /* A/変換終了まで待つD 約5us */
126         databuf[bufptr] = ADREAD(); /* ◎変換データを格納 */
127     }
128
129     if (play_mode == PLAY){
130         /* ここに再生のときの処理を記述する以下のコメントを参照のこと */
131         da_out(0, databuf[bufptr]); /* ◎D/Aにデータを出力 */
132     }
133
134     bufptr++; /* バッファポインタを +1 */
135     timer_intflag_reset(0); /* タイマの割り込みフラグをクリア0 */
136     ENINT(); /* を割り込み許可状態にCPU */
137 }

```

リスト 2: Makefile

```

1  # H8/3052 の雛型 Makefile
2  # 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)
3  #   TARGET = , SOURCE_C = , SOURCE_ASM = を指定する
4  #   リモートデバッグのときは、   GDBREMOTE_DBG = true とする
5  #   その他は通常、変更の必要はない
6  # 手順 2. make -f 名makefile で make する例: (make -f make-test)
7
8  # 生成するファイルとソースファイルの指定
9  # 1. 生成するオブジェクトのファイル名を指定
10 TARGET = ad-da.mot
11 # 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
12 SOURCE_C = ad-da.c lcd.c ad.c da.c timer.c
13 # 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
14 # スタートアップルーチンは除く()
15 SOURCE_ASM =
16
17 # 生成するオブジェクトの種類を指定
18 # ※の項目は通常変更する必要がない()
19 #
20 # 1. によるリモートデバッグ指定GDB
21 # true : 指定する   その他: 指定しない
22 REMOTE_DBG =
23
24 # 2. 上デバッグまたは化指定RAMROM ※
25 # ram : 上で実行RAM rom : 化ROM
26 ON_RAM = ram
27
28 # 3. 使用領域の指定RAM ※
29 # : 化→プログラムとスタックは外部を使用extRAMRAM
30 # 化→スタックは外部   ROMRAM
31 # : 化→プログラムとスタックは内部を使用intRAMRAM
32 # 化→スタックは内部   ROMRAM
33 # 指定なし: 化→プログラムは外部、スタック変更なしRAMRAM
34 #   化→スタックは外部   ROMRAM
35 RAM_CAP = ext
36
37 # 4. によるデバッグを行うかどうかの指定GDB ※
38 USE_GDB = true
39
40 #
41 # バスの設定
42 #
43 CMD_PATH = /usr/local/bin
44 LIB_PATH = /home/class/common/H8/lib
45
46 #
47 # クロスコンパイラ関係
48 #
49 CC = $(CMD_PATH)/h8300-hms-gcc
50 LD = $(CMD_PATH)/h8300-hms-ld
51 OBJCOPY = $(CMD_PATH)/h8300-hms-objcopy
52 SIZE = $(CMD_PATH)/h8300-hms-size
53
54 #
55 # ターゲット指定
56 #
57 TARGET_COFF = $(TARGET:.mot=.coff)
58 MAP_FILE = $(TARGET:.mot=.map)
59
60 #
61 # 出力フォーマット
62 # binary : binary, srec : Motorola S record, ihex : Intel Hex
63 #
64 OUTPUT_FORMAT = -O srec --srec-forceS3
65
66 #
67 # コンパイラオプション
68 #
69 # インクルードディレクトリの追加("*****.h"指定のみ有効)
70 INCLUDES = -I/home/class/common/H8/include
71 # コンパイラオプションの指定
72 # - : mhH8/300シリーズ指定H
73 # - : 条件分岐コードの最適化mrelax
74 # - : 型変数のビット数指定mint32int
75 # - : の最適化レベルの指定O2gcc
76 # - : コンパイル時の警告メッセージの選択Wall全て()
77 CFLAGS = -mh -mrelax -mint32 -O2 $(INCLUDES) -Wall
78
79 #
80 # 指定に合わせたスタートアップルーチンとリンカスクリプトの選択
81 #
82 ifeq ($(REMOTE_DBG), true)

```

```

83  USE_GDB = true
84  ON_RAM = ram
85  RAM_CAP =
86  endif
87
88  ifeq ($(USE_GDB), true)
89  CFLAGS := $(CFLAGS) -g
90  endif
91
92  ifeq ($(ON_RAM), ram)
93  LDSCRIPT = $(LIB_PATH)/h8-3052-ram.x
94  STARTUP = $(LIB_PATH)/ramcrt.s
95  ifeq ($(RAM_CAP), int)
96  LDSCRIPT = $(LIB_PATH)/h8-3052-ram8k.x
97  STARTUP = $(LIB_PATH)/ramcrt-8k.s
98  endif
99  ifeq ($(RAM_CAP), ext)
100  LDSCRIPT = $(LIB_PATH)/h8-3052-ram.x
101  STARTUP = $(LIB_PATH)/ramcrt-ext.s
102  endif
103  ifeq ($(REMOTE_DBG), true)
104  LDSCRIPT = $(LIB_PATH)/h8-3052-ram-dbg.x
105  STARTUP = $(LIB_PATH)/ramcrt-dbg.s
106  endif
107  else
108  ifeq ($(RAM_CAP), int)
109  LDSCRIPT = $(LIB_PATH)/h8-3052-rom8k.x
110  STARTUP = $(LIB_PATH)/romcrt-8k.s
111  else
112  LDSCRIPT = $(LIB_PATH)/h8-3052-rom.x
113  STARTUP = $(LIB_PATH)/romcrt-ext.s
114  endif
115  endif
116
117  #
118  # リンク時のコンパイラオプションの指定
119  # -T : リンカスクリプトファイルの指定filename
120  # - : 標準のスタートアップを使用しないnostartfiles
121  # -Wlパラメータ...: リンカに渡すパラメータ指定,,
122  # -Map : メモリマップをに出力mapfilenamemapfilename
123  LDFLAGS = -T $(LDSCRIPT) -nostartfiles -Wl,-Map,$(MAP_FILE)
124
125  #
126  # オブジェクトの指定
127  #
128  OBJ = $(STARTUP:.s=.o) $(SOURCE_C:.c=.o) $(SOURCE_ASM:.s=.o)
129
130  #
131  # サフィックスルール適用の拡張子指定
132  #
133  .SUFFIXES: .c .s .o
134
135  #
136  # ルール
137  #
138  $(TARGET) : $(TARGET_COFF)
139  $(OBJCOPY) -v $(OUTPUT_FORMAT) $(TARGET_COFF) $(TARGET)
140
141  $(TARGET_COFF) : $(OBJ)
142  $(CC) $(CFLAGS) $(LDFLAGS) $(OBJ) -o $(TARGET_COFF)
143  $(SIZE) -Ax $(TARGET_COFF)
144
145  clean :
146  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 "h8-3052-int.h"
3  #include "lcd.h"
4  #include "ad.h"
5  #include "da.h"
6  #include "timer.h"
7

```

```

8  #define BUFSIZE          30 /* バッファの大きさ(kB) */
9  #define SAMPLINGTIME 100 /* 録音再生時のサンプリング周期(us) */
10 #define SAMPLE          0 /* 動作選択値録音() */
11 #define PLAY            1 /* 動作選択値再生() */
12 #define NOSELECT        -1 /* 動作選択値未選択() */
13
14 volatile unsigned char databuf[(unsigned long)BUFSIZE * 1024];
15 volatile unsigned long bufptr;
16 volatile int play_mode;
17
18 int      main(void);
19 unsigned char menu(void);
20 void     sample_replay(int mode);
21 void     int_imia0(void);
22
23 int main(void)
24 {
25     /* キー入力情報を取得するための変数を宣言する */
26     unsigned char key_data;
27
28     ROMEMU(); /* エミュレーションをROMON */
29     lcd_init(); /* LCD の初期化 */
30     ad_init(); /* A/D 変換器の初期化 */
31     da_init(); /* D/A 変換器の初期化 */
32     timer_init(); /* タイマの初期化 */
33
34     /* タイマチャネルの割り込み間隔(0) */
35     timer_set(0, SAMPLINGTIME);
36
37     /* ここにキー入力取得のためのポートの初期化を記述する */
38     P6DDR &= ~0x07; /* P60,1,2 入力 */
39     PADDR |= 0x0f; /* PA0,1,2,3 出力 */
40
41     while (1) {
42         play_mode = NOSELECT;
43         key_data = menu(); /* メニューを選ぶ */
44
45         /* 録音キーが押されたら、再生キーが押されたらSAMPLEPLAY */
46         /* をに格納する処理を記述するplay_mode */
47         switch( key_data ){
48             case '*':
49                 play_mode = SAMPLE;
50                 break;
51
52             case '#':
53                 play_mode = PLAY;
54                 break;
55         }
56
57         /* キー入力されていれば録音再生の関数を呼び出す処理を記述する */
58         if(play_mode != NOSELECT){
59             sample_replay( play_mode );
60         }
61     }
62
63     return 1;
64 }
65
66 unsigned char menu(void)
67 /* LCD にメニューを書いて動作を選択するための関数 */
68 /* 戻り値は入力キー情報 */
69 {
70     /* キー入力取得のための変数を宣言する */
71     unsigned char cf, key_data;
72
73     lcd_cursor(0,0);
74     lcd_printstr(" Push * or # key");
75
76     /* キー入力判定用変数の初期化 */
77     cf = 0;
78     key_data = 0;
79
80     while (cf == 0){ /* キー入力するまでループする */
81         //key 0
82         PADR = 0x0e;
83         cf = P6DR;
84         cf = ~cf;
85         cf &= 0x07;
86         switch(cf) {
87             case 1 : key_data = '*'; break;
88             case 4 : key_data = '#'; break;
89         }
90     }
91 }

```

```

92  /* 入力されたキーの情報を返す */
93  return key_data;
94  }
95
96  void sample_replay(int mode)
97  /* 録音または再生を行う関数 */
98  /* mode: PLAY, SAMPLE */
99  {
100     lcd_cursor(0, 1);
101     lcd_printstr(" ");
102     if (mode == PLAY){ /* 再生モードの処理 */
103         /* ここにスピーカをスピーカとして使用する命令を記述する */
104         speaker_switch(SPEAKER);
105     }
106     if (mode == SAMPLE){ /* 録音モードの処理 */
107         /* ここにスピーカをマイクとして使用する命令を記述する */
108         speaker_switch(MIC);
109     }
110     bufptr = 0; /* バッファポインタを初期化 */
111     timer_start(0); /* サンプリングタイマチャネルのスタート(0) */
112     ENINT(); /* 割り込み許可CPU */
113
114     unsigned long temp;
115     while (bufptr < ((unsigned long)BUFSIZE * 1024)){
116         temp = bufptr;
117         lcd_cursor(temp/((BUFSIZE/5)*1024), 1);
118         lcd_printch(temp/((BUFSIZE/5)*1024) + 1 + '0');
119     }
120
121     lcd_cursor(0, 1);
122     lcd_printstr(" ");
123     /* バッファが一杯になるまで実行 */
124     speaker_switch(MIC); /* スピーカーオフ */
125     timer_stop(0); /* タイマのストップ */
126 }
127
128 #pragma interrupt
129 void int_imia0(void)
130 /* 録音・再生用のタイマ割り込みハンドラ */
131 /* プレイモードによってデータの格納か出力を行う */
132 {
133     if (play_mode == SAMPLE){
134         /* ここに録音のときの処理を記述する以下のコメントを参照のこと */
135         ad_start(0, 0); /* ◎A/D変換スタート */
136         while(ADSTATUS() == 0); /* A/変換終了まで待つD 約5us */
137         databuf[bufptr] = ADREAD(); /* ◎変換データを格納 */
138     }
139
140     if (play_mode == PLAY){
141         /* ここに再生のときの処理を記述する以下のコメントを参照のこと */
142         da_out(0, databuf[bufptr]); /* ◎D/Aにデータを出力 */
143     }
144
145     bufptr++; /* バッファポインタを +1 */
146     timer_intflag_reset(0); /* タイマの割り込みフラグをクリア */
147     ENINT(); /* を割り込み許可状態にCPU */
148 }

```

リスト 4: 課題 3 のソース

```

1  #include "h8-3052-iodef.h"
2  #include "h8-3052-int.h"
3  #include "lcd.h"
4  #include "ad.h"
5  #include "da.h"
6  #include "timer.h"
7
8  #define BUFSIZE 30 /* バッファの大きさ(kB) */
9  #define SAMPLINGTIME 100 /* 録音再生時のサンプリング周期(us) */
10 #define SAMPLE 0 /* 動作選択値録音 */
11 #define PLAY 1 /* 動作選択値再生 */
12 #define NOSELECT -1 /* 動作選択値未選択 */
13
14 volatile unsigned char databuf[(unsigned long)BUFSIZE * 1024];
15 volatile unsigned long bufptr;
16 volatile int play_mode;
17
18 int main(void);
19 unsigned char menu(void);
20 void sample_replay(int mode);
21 void int_imia0(void);
22

```



```

23 int main(void)
24 {
25     /* キー入力情報を取得するための変数を宣言する */
26     unsigned char key_data;
27
28     ROMEMU(); /* エミュレーションをROMON */
29     lcd_init(); /* LCD の初期化 */
30     ad_init(); /* A/D 変換器の初期化 */
31     da_init(); /* D/A 変換器の初期化 */
32     timer_init(); /* タイマの初期化 */
33
34     /* タイマチャネルの割り込み間隔(0) */
35     timer_set(0,SAMPLINGTIME);
36
37     /* ここにキー入力取得のためのポートの初期化を記述する */
38     P6DDR &= ~0x07; /* P60,1,2 入力 */
39     PADDR |= 0x0f; /* PA0,1,2,3 出力 */
40
41     while (1) {
42         play_mode = NOSELECT;
43         key_data = menu(); /* メニューを選ぶ */
44
45         /* 録音キーが押されたら、再生キーが押されたらSAMPLEPLAY */
46         /* をに格納する処理を記述するplay_mode */
47         switch( key_data ){
48             case '*':
49                 play_mode = SAMPLE;
50                 break;
51
52             case '#':
53                 play_mode = PLAY;
54                 break;
55         }
56
57         /* キー入力されていれば録音再生の関数を呼び出す処理を記述する/ */
58         if(play_mode != NOSELECT){
59             sample_replay( play_mode );
60         }
61     }
62
63     return 1;
64 }
65
66 unsigned char menu(void)
67 /* LCD にメニューを書いて動作を選択するための関数 */
68 /* 戻り値は入力キー情報 */
69 {
70     /* キー入力取得のための変数を宣言する */
71     unsigned char cf, key_data;
72
73     lcd_cursor(0,0);
74     lcd_printstr(" Push 1 or 2 key");
75
76     /* キー入力判定用変数の初期化 */
77     cf = 0;
78     key_data = 0;
79
80     while (cf == 0 ){ /* キー入力するまでループする */
81         //key 1,2,3
82         P6ADR = 0x07;
83         cf = P6DR;
84         cf = ~cf;
85         cf &= 0x07;
86         switch(cf) {
87             case 1 : key_data = '*'; break;
88             case 2 : key_data = '#'; break;
89         }
90     }
91
92     /* 入力されたキーの情報を返す */
93     return key_data;
94 }
95
96 void sample_replay(int mode)
97 /* 録音または再生を行う関数 */
98 /* mode: PLAY, SAMPLE */
99 {
100     //lcd_printstr(" ");
101     lcd_cursor(0, 1);
102     if (mode == PLAY){ /* 再生モードの処理 */
103         /* ここにスピーカをスピーカとして使用する命令を記述する */
104         lcd_printstr("Now Playing...");
105         speaker_switch(SPEAKER);
106     }

```

```

107 if (mode == SAMPLE){ /* 録音モードの処理 */
108     /* ここにスピーカをマイクとして使用する命令を記述する */
109     lcd_printstr("Now Sampling...");
110     speaker_switch(MIC);
111 }
112 bufptr = 0; /* バッファポインタを初期化 */
113 timer_start(0); /* サンプルングタイマチャネルのスタート(0) */
114 ENINT(); /* 割り込み許可CPU */
115
116 while (bufptr < ((unsigned long)BUFSIZE * 1024));
117
118 lcd_cursor(0, 1);
119 lcd_printstr(" ");
120 /* バッファが一杯になるまで実行 */
121 speaker_switch(MIC); /* スピーカーオフ */
122 timer_stop(0); /* タイマのストップ */
123 }
124
125 #pragma interrupt
126 void int_imia0(void)
127 /* 録音・再生用のタイマ割り込みハンドラ */
128 /* プレイモードによってデータの格納か出力を行う */
129 {
130     if (play_mode == SAMPLE){
131         /* ここに録音のときの処理を記述する以下のコメントを参照のこと */
132         ad_start(0, 0); /* ◎A/D変換スタート */
133         while(ADSTATUS() == 0); /* A変換終了まで待つD 約5us */
134         databuf[bufptr] = ADREAD(); /* ◎変換データを格納 */
135     }
136
137     if (play_mode == PLAY){
138         /* ここに再生のときの処理を記述する以下のコメントを参照のこと */
139         da_out(0, databuf[bufptr]); /* ◎D/Aにデータを出力 */
140     }
141
142     bufptr++; /* バッファポインタを +1 */
143     timer_intflag_reset(0); /* タイマの割り込みフラグをクリア0 */
144     ENINT(); /* を割り込み許可状態にCPU */
145 }

```

リスト 5: 課題 4 のソース

```

1 #include "h8-3052-iodef.h"
2 #include "h8-3052-int.h"
3 #include "lcd.h"
4 #include "ad.h"
5 #include "da.h"
6 #include "timer.h"
7
8 #define BUFSIZE 30 /* バッファの大きさ(kB) */
9 #define SAMPLINGTIME 100 /* 録音再生時のサンプルング周期(us) */
10 #define SAMPLE 0 /* 動作選択値録音 */
11 #define INVERSE 2
12 #define PLAY 1 /* 動作選択値再生 */
13 #define NOSELECT -1 /* 動作選択値未選択 */
14
15 #define TIME 120000 /* 録音再生時間 */
16
17 volatile unsigned char databuf[(unsigned long)BUFSIZE * 1024];
18 volatile unsigned long bufptr;
19 volatile int play_mode;
20
21 int main(void);
22 unsigned char menu(void);
23 void sample_replay(int mode);
24 void int_imia0(void);
25
26 int main(void)
27 {
28     /* キー入力情報を取得するための変数を宣言する */
29     unsigned char key_data;
30
31     ROMEMU(); /* エミュレーションをROMON */
32     lcd_init(); /* LCD の初期化 */
33     ad_init(); /* A/D 変換器の初期化 */
34     da_init(); /* D/A 変換器の初期化 */
35     timer_init(); /* タイマの初期化 */
36
37     /* タイマチャネルの割り込み間隔(0) */
38     timer_set(0, SAMPLINGTIME);
39
40     /* ここにキー入力取得のためのポートの初期化を記述する */

```

```

41 P6DDR &= ~0x07; /* P60,1,2 入力 */
42 PADDR |= 0x0f; /* PA0,1,2,3 出力 */
43
44 while (1) {
45     play_mode = NOSELECT;
46     key_data = menu(); /* メニューを選ぶ */
47
48     /* 録音キーが押されたら、再生キーが押されたらSAMPLEPLAY */
49     /* をに格納する処理を記述するplay_mode */
50     switch( key_data ){
51         case '*':
52             play_mode = SAMPLE;
53             break;
54
55         case '5':
56             play_mode = INVERSE;
57             break;
58     }
59
60     /* キー入力されていれば録音再生の関数を呼び出す処理を記述する/ */
61     if(play_mode != NOSELECT){
62         sample_replay( play_mode );
63     }
64 }
65
66 return 1;
67 }
68
69 unsigned char menu(void)
70 /* LCD にメニューを書いて動作を選択するための関数 */
71 /* 戻り値は入力キー情報 */
72 {
73     /* キー入力取得のための変数を宣言する */
74     unsigned char cf, key_data;
75
76     lcd_cursor(0,0);
77     lcd_printstr(" Push * or 5 key");
78
79     /* キー入力判定用変数の初期化 */
80     cf = 0;
81     key_data = 0;
82
83     while (cf == 0) { /* キー入力するまでループする */
84         PADR = 0x0b;
85         cf = P6DR;
86         cf = ~cf;
87         cf &= 0x07;
88         switch(cf) {
89             case 2 : key_data = '5'; break;
90         }
91         PADR = 0x0e;
92         cf = P6DR;
93         cf = ~cf;
94         cf &= 0x07;
95         switch(cf) {
96             case 1 : key_data = '*'; break;
97         }
98         if(key_data != 0)
99             break;
100     }
101
102     /* 入力されたキーの情報を返す */
103     return key_data;
104 }
105
106 void sample_replay(int mode)
107 /* 録音または再生を行う関数 */
108 /* mode: PLAY, SAMPLE */
109 {
110     lcd_cursor(0, 1);
111     if (mode == INVERSE){ /* 再生モードの処理 */
112         /* ここにスピーカをスピーカとして使用する命令を記述する */
113         lcd_printstr("Now Inverse...");
114         speaker_switch(SPEAKER);
115     }
116     if (mode == SAMPLE){ /* 録音モードの処理 */
117         /* ここにスピーカをマイクとして使用する命令を記述する */
118         lcd_printstr("Now Sampling...");
119         speaker_switch(MIC);
120     }
121     bufptr = 0; /* バッファポインタを初期化 */
122     timer_start(0); /* サンプリングタイマチャネルのスタート(0) */
123     ENINT(); /* 割り込み許可CPU */
124 }

```

```

125 while (bufptr < ((unsigned long)BUFSIZE * 1024));
126
127 lcd_cursor(0, 1);
128 lcd_printstr(" ");
129 /* バッファが一杯になるまで実行 */
130 speaker_switch(MIC); /* スピーカーオフ */
131 timer_stop(0); /* タイマのストップ */
132 }
133
134 #pragma interrupt
135 void int_imia0(void)
136 /* 録音・再生用のタイマ割り込みハンドラ */
137 /* プレイモードによってデータの格納か出力を行う */
138 {
139     if (play_mode == SAMPLE){
140         /* ここに録音のときの処理を記述する以下のコメントを参照のこと */
141         ad_start( 0, 0 ); /* ◎A/D変換スタート */
142         while(ADSTATUS() == 0); /* A/D変換終了まで待つD 約5us */
143         databuf[bufptr] = ADREAD(); /* ◎変換データを格納 */
144     }
145
146     if (play_mode == INVERSE){
147         /* ここに再生のときの処理を記述する以下のコメントを参照のこと */
148         da_out(0, databuf[BUFSIZE*1024-bufptr]); /* ◎D/Aにデータを出力 */
149     }
150
151     bufptr++; /* バッファポインタを +1 */
152     timer_intflag_reset(0); /* タイマの割り込みフラグをクリア */
153     ENINT(); /* を割り込み許可状態にCPU */
154 }

```

リスト 6: 課題 5 のソース

```

1 #include "h8-3052-iodef.h"
2 #include "h8-3052-int.h"
3 #include "lcd.h"
4 #include "ad.h"
5 #include "da.h"
6 #include "timer.h"
7
8 #define BUFSIZE 30 /* バッファの大きさ(kB) */
9 #define SAMPLINGTIME 100 /* 録音再生時のサンプリング周期(us) */
10 #define SAMPLE 0 /* 動作選択値録音 */
11 #define INVERSE 2
12 #define PLAY 1 /* 動作選択値再生 */
13 #define NOSELECT -1 /* 動作選択値未選択 */
14
15 #define TIME 120000 /* 録音再生時間 */
16
17 volatile unsigned char databuf[(unsigned long)BUFSIZE * 1024];
18 volatile unsigned long bufptr;
19 volatile int play_mode;
20
21 int main(void);
22 unsigned char menu(void);
23 void sample_replay(int mode);
24 void int_imia0(void);
25
26 int main(void)
27 {
28     /* キー入力情報を取得するための変数を宣言する */
29     unsigned char key_data;
30
31     ROMEMU(); /* エミュレーションをROMON */
32     lcd_init(); /* LCD の初期化 */
33     ad_init(); /* A/D 変換器の初期化 */
34     da_init(); /* D/A 変換器の初期化 */
35     timer_init(); /* タイマの初期化 */
36
37     /* タイマチャネルの割り込み間隔(0) */
38     timer_set(0, SAMPLINGTIME);
39
40     /* ここにキー入力取得のためのポートの初期化を記述する */
41     P6DDR &= ~0x07; /* P60,1,2 入力 */
42     PADDR |= 0x0f; /* PA0,1,2,3 出力 */
43
44     while (1) {
45         play_mode = NOSELECT;
46         key_data = menu(); /* メニューを選ぶ */
47
48         /* 録音キーが押されたら、再生キーが押されたらSAMPLEPLAY */
49         /* をに格納する処理を記述するplay_mode */

```

```

50     switch( key_data ){
51         case '*':
52             play_mode = SAMPLE;
53             break;
54
55         case '#':
56             play_mode = PLAY;
57             break;
58
59         case '5':
60             play_mode = INVERSE;
61             break;
62     }
63
64     /* キー入力されていれば録音再生の関数を呼び出す処理を記述する */
65     if(play_mode != NOSELECT){
66         sample_replay( play_mode );
67     }
68 }
69
70 return 1;
71 }
72
73 unsigned char menu(void)
74 /* LCD にメニューを書いて動作を選択するための関数 */
75 /* 戻り値は入力キー情報 */
76 {
77     /* キー入力取得のための変数を宣言する */
78     unsigned char cf, key_data;
79
80     lcd_cursor(0,0);
81     lcd_printstr("Push *, # or 5");
82
83     /* キー入力判定用変数の初期化 */
84     cf = 0;
85     key_data = 0;
86
87     while (cf == 0){ /* キー入力するまでループする */
88         PADR = 0x0b;
89         cf = P6DR;
90         cf = ~cf;
91         cf &= 0x07;
92         switch(cf) {
93             case 2 : key_data = '5'; break;
94         }
95
96         PADR = 0x0e;
97         cf = P6DR;
98         cf = ~cf;
99         cf &= 0x07;
100        switch(cf) {
101            case 1 : key_data = '*'; break;
102            case 4 : key_data = '#'; break;
103        }
104
105        if(key_data != 0)
106            break;
107    }
108
109    /* 入力されたキーの情報を返す */
110    return key_data;
111 }
112
113 void sample_replay(int mode)
114 /* 録音または再生を行う関数 */
115 /* mode: PLAY, SAMPLE */
116 {
117     lcd_cursor(0, 1);
118     if (mode == INVERSE){ /* 再生モードの処理 */
119         /* ここにスピーカをスピーカとして使用する命令を記述する */
120         lcd_printstr("Now Inverse...");
121         speaker_switch(SPEAKER);
122     }
123     if (mode == SAMPLE){ /* 録音モードの処理 */
124         /* ここにスピーカをマイクとして使用する命令を記述する */
125         lcd_printstr("Now Sampling...");
126         speaker_switch(MIC);
127     }
128     if(mode == PLAY){
129         lcd_printstr("Now Playing...");
130         speaker_switch(SPEAKER);
131     }
132     bufptr = 0; /* バッファポインタを初期化 */
133     timer_start(0); /* サンプリングタイマチャネルのスタート(0) */

```

```

134 ENINT(); /* 割り込み許可CPU */
135
136 while (bufptr < ((unsigned long)BUFSIZE * 1024));
137
138 lcd_cursor(0, 1);
139 lcd_printstr(" ");
140 /* バッファが一杯になるまで実行 */
141 speaker_switch(MIC); /* スピーカーオフ */
142 timer_stop(0); /* タイマのストップ */
143 }
144
145 #pragma interrupt
146 void int_imia0(void)
147 /* 録音・再生用のタイマ割り込みハンドラ */
148 /* プレイモードによってデータの格納か出力を行う */
149 {
150     if (play_mode == SAMPLE){
151         /* ここに録音のときの処理を記述する以下のコメントを参照のこと */
152         ad_start( 0, 0 ); /* ◎A/D変換スタート */
153         while(ADSTATUS() == 0); /* A/変換終了まで待つD 約5us */
154         databuf[bufptr] = ADREAD(); /* ◎変換データを格納 */
155     }
156
157     if (play_mode == INVERSE){
158         /* ここに再生のときの処理を記述する以下のコメントを参照のこと */
159         da_out(0, databuf[BUFSIZE*1024-bufptr]); /* ◎D/Aにデータを出力 */
160     }
161     if (play_mode == PLAY){
162         /* ここに再生のときの処理を記述する以下のコメントを参照のこと */
163         da_out(0, databuf[bufptr]); /* ◎D/Aにデータを出力 */
164     }
165
166     bufptr++; /* バッファポインタを +1 */
167     timer_intflag_reset(0); /* タイマの割り込みフラグをクリア0 */
168     ENINT(); /* を割り込み許可状態にCPU */
169 }

```