第 6 章　機上搭載機器について

成岡 優

2006 年 10 月 15 日

全般の説明

全体説明で示したとおり、室内飛行機の機体上にはマイコンを搭載した制御機器がある。制御機器の外観を図 1 に示す。図中で中央に見える部分が本プロジェクトで作成した制御機器である。

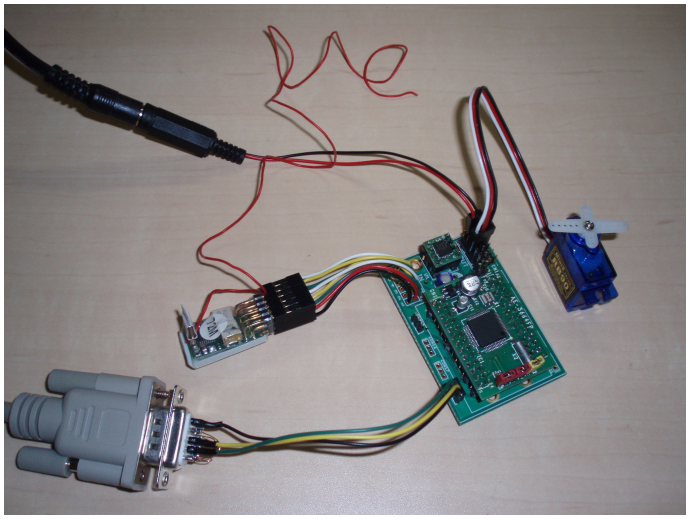

図 1　制御機器の外観

この制御機器では、まず、無線を通じてラジコン受信機に送られてきた制御情報や外界の情報 (機体の位置等) をデコードするとともに、加速度計や超音波センサを利用して機体の姿勢や詳細な位置を取得する。ラジコン受信機とマイコンの接続図を図 2 に、超音波センサとマイコンの接続図を図 3 にあげる。加速度計はマイコンに内蔵の A/D 変換の機能を利用して取り込んでいる。

そして、これらの情報を元にモーターやラダー、エレベータの制御量を計算し、それらを駆動するためのスピコンやサーボに対して適切なパルス信号を発行する。パルスはマイコンに内蔵のタイマの機能を利用して行っている。これらは全て同時に処理される。

以下、開発環境の構築、開発手順について説明をした後、作成したソフトウェア、ハードウェアについて記載する。

開発環境の構築

1. Cygwin をインストールする。この後 gcc や make、wget が必要になるので、パッケージの選択画面で忘れずにインストールすること (参考:コンパイラ導入手順)。
2. 開発環境を以下のコマンドにより構築する (参考:h8300-elf Makefile)。かかる時間は 2006 年 8 月現在の早いマシンにおいても 3 時間程度なので、暇なときに行うこと。

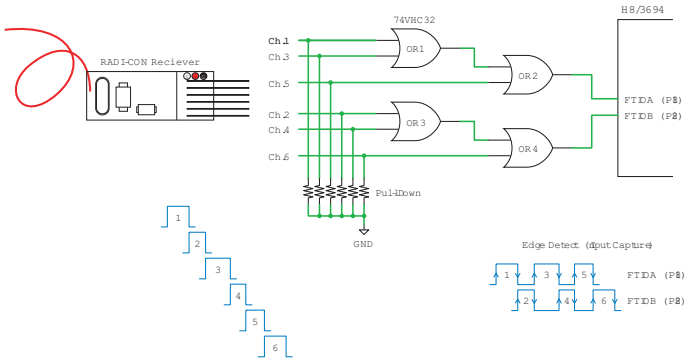

図 2　ラジコン受信機とマイコンの接続図

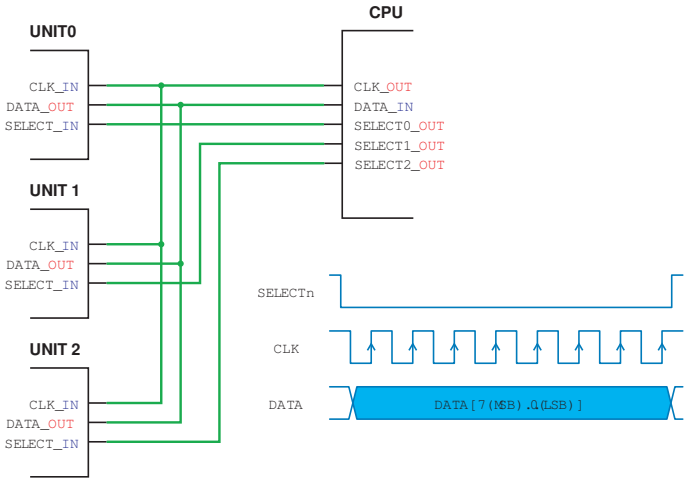

図 3　超音波センサとマイコンの接続図

```
$ if ![ −d /usr/local/src ]; then mkdir /usr/local/src; fi
$ cd /usr/local/src
$ wget http://fenrir.naruoka.org/download/embedded/h8devel.makefile
$ make −f h8devel.makefile
```

3. ソースコード一式を特定のディレクトリに展開する。

開発手順

1. プログラムを編集後、以下のコマンドを実行する

```
$ cd indoor
$ make Makefile
```

2. build_by_gcc に test.mot というファイルができるので、適当な書込みソフト (例えば StrawberryLinux の落合さんが開発された H8WriteTurbo など) を使ってプログラムを H8/3694 マイコンに書込む。書込みの際はジャンパピンの設定に注意すること。

ソフトウェア

機上搭載機器のためにプログラムを作成した。このプログラムは機体搭載機器の中の H8/3694 で動作するものである。以下に全ソースコードを掲載する。

```
./
|
|
|-- indoor/ ..................................................室内飛行機プロジェクトに特化したソース
|   |-- Makefile [1] ..........................................プロジェクト全体ビルド用ファイル
|   |-- common.h [2] ..........................................................共通ヘッダ
|   |-- test.c [3] ..........................................................メインプログラム
|   |-- ultra_sonic.h [4]
|   |-- ultra_sonic.c [5] ....................................超音波モジュールとのインターフェイス
|-- framework/ ...............................................他プロジェクトで再利用可能なソース
|   |-- dependency/ ..........................................................マイコンに依存
|   |   |-- AKI3694/ ..................................................秋月電子 H8/3694F 用ファイル
|   |   |   |-- Makefile [6]
|   |   |   |-- rom.x [7] ..................................................リンカスクリプト
|   |   |   |-- romcrt0.S [8] ..............................................スタートアップルーチン
|   |   |   |-- 3694.h [9] ..................................................I/O 定義
|   |   |   |-- AKI3694.h [10]
|   |   |   |-- dependency.c [11] ..........................................割り込み許可/拒否など
|   |   |   |-- common.h [12] ..........................................................共通ヘッダ
|   |   |   |-- ad.h [13]
|   |   |   |-- ad.c [14] ..................................................A/D 変換
|   |   |   |-- sci.h [15]
|   |   |   |-- sci.c [16] ..........................................シリアルインターフェイス (SCI)
|   |   |   |-- timer.h [17]
|   |   |   |-- timer.c [18] ..................................................タイマ
|   |   |   |-- wdt.h [19]
|   |   |   |-- wdt.c [20] ..................................ウォッチドックタイマ (WDT)
|   |   |-- util/ ..................................................マイコンに非依存
|   |   |   |-- Makefile [21]
|   |   |   |-- common.h [22] ..........................................................共通ヘッダ
|   |   |   |-- conv.h [23]
|   |   |   |-- conv.c [24] ..................................数文字変換、PRINTF の代用品
|   |   |   |-- fifo_char.h [25]
|   |   |   |-- fifo_char.c [26] ..........................................リングバッファ (CHAR 型)
|   |   |   |-- fifo_num.h [27]
|   |   |   |-- fifo_num.c [28] ..................................................リングバッファ (数値型)
```

ソースコード 1　indoor/Makefile

```
1 PRE = h8300-elf-
```

```
export CC = $(PRE)gcc
export OBJCPY = $(PRE)objcopy

export BUILD_DIR = $(CURDIR)/build_by_gcc
NEWLIB_DIR = /usr/local/h8300-elf/lib/h8300h
ROOT_DIR = $(CURDIR)
FRAMEWORK = $(ROOT_DIR)/../framework
DEPENDENCY = $(FRAMEWORK)/dependency/AKI3694
UTIL = $(FRAMEWORK)/util
OTHER_PACKAGE = $(UTIL) $(DEPENDENCY)
INCLUDES = \
  -I$(ROOT_DIR) $(patsubst %,-I%,$(OTHER_PACKAGE)) \
  -I$(FRAMEWORK) \
  -I/usr/local/h8300-elf/include

# ROM TARGET
STARTUP = $(DEPENDENCY)/romcrt0.S
LDSCRIPT = $(DEPENDENCY)/rom.x

export CFLAGS = \
  $(INCLUDES) \
  -Wall -mh -g -mrelax -gdwarf-2 -mint32 #-ansi -O2
export LDFLAGS = \
  -nostdlib -nostartfiles -nodefaultlibs -mrelax -L$(NEWLIB_DIR) -lc -lm -lgcc \
  -T $(LDSCRIPT) -Wl,-Map,$(MAP) -Wl,-static

TARGET = test
GOAL = $(TARGET).elf $(TARGET).mot $(TARGET).abs

SRCS = $(shell ls *.c)
MAP = $(BUILD_DIR)/$(TARGET).map

all : $(BUILD_DIR) $(patsubst %,$(BUILD_DIR)/%,$(GOAL))
  @echo COMPLETED.

$(BUILD_DIR) :
  mkdir $@

$(BUILD_DIR)/%.mot : $(BUILD_DIR)/%.elf
  $(OBJCPY) -O srec $< $@

$(BUILD_DIR)/%.abs : $(BUILD_DIR)/%.elf
  cp $< $@

$(BUILD_DIR)/%.elf : $(STARTUP) $(SRCS) $(BUILD_DIR)/*.o
  $(CC) $^ $(CFLAGS) $(LDFLAGS) -o $@

$(STARTUP) :

$(BUILD_DIR)/*.o : other_package

other_package :
  for package in $(OTHER_PACKAGE); do \
    (cd $$package && $(MAKE)); \
  done

clean:
  for package in $(OTHER_PACKAGE); do \
    (cd $$package && $(MAKE) clean); \
  done
  rm -f $(BUILD_DIR)/*

.PHONY: clean all other_package
```

ソースコード 2　indoor/common.h

```
#ifndef __COMMON_H__
#define __COMMON_H__

/*
 * 共通ヘッダ
 *
 * @author fenrir (M.Naruoka)
 * @since 04/06/03
 * @version 1.0
 * @see ad.c
 */

#define DEBUG 0

#define FIFO_CHAR_SIZE_T unsigned char

#define FIFO_NUM_SIZE_T unsigned char
#define FIFO_NUM_T unsigned short

#define AD_ONLY_4
#define AD_BUFFER_SIZE 16

#define WDT_TMWD_SETTING 0x0F

/* タイマA の設定
 *
 * インターバルタイマとして動作するように
 * Φ/512
 * 20MHz / (512 * 256) = 160Hz = 6.4ms で割り込みがかかるようにする
 */
#define TIMER_A_SETTING 0x03

#define SCI3_TX_BUFFER_SIZE 64
#define SCI3_RX_BUFFER_SIZE 16

/**
 * IRQ コントロール
 */
#ifdef __cplusplus
extern "C"
{
#endif

extern void enableIRQ();
extern void disableIRQ();
extern void sleep();
extern void nop();

#define sci_dump(str) sci3_write(str, strlen(str))
//#define sci_dump(str) printf(str)

#ifdef __cplusplus
};
#endif

/**
 * 定数
 */
#define TRUE 1
#define FALSE 0
#define EXIT 0
#define CONTINUE 1
```

```
63 #define VOID −1
64 #define PI 3.14159
65 #define INIT_STR "Initial."
66 #ifndef NULL
67   #define NULL 0
68 #endif
69
70 /* ユーティリティ */
71 #define min(a, b) ((a <= b) ? a : b)
72 #define max(a, b) ((a >= b) ? a : b)
73 #define pow2(x) (x * x)
74
75 #endif /* __COMMON_H__ */
```

ソースコード 3　indoor/test.c

```
1  #include <stdio.h>
2  #include <string.h>
3
4  /* 非依存ヘッダ */
5  #include "common.h"
6  #include "ultra_sonic.h"
7
8  /* ユーティリティヘッダ */
9  #include <fifo_num.h>
10 #include <conv.h>
11
12 /* 依存ヘッダ */
13 #include <AKI3694.h>
14 #include <sci.h>
15 #include <timer.h>
16 #include <ad.h>
17 #include <wdt.h>
18
19 // サーボ(スピコン含む)チャンネル数
20 #define SERVO_CH 6
21 unsigned short output_pulse[SERVO_CH];
22 unsigned short input_pulse[SERVO_CH];
23
24 // 超音波チャンネル数
25 #define ULTRA_SONIC_CH 1
26
27 // 重力加速度、単位はmm/s^2
28 #define SCALE_1G 9806
29
30 // サーボに与えるパルス幅の設定
31 #define MAX_PULSE_WIDTH 22000
32 #define MIN_PULSE_WIDTH 8000
33 #define MIDDLE_PULSE_WIDTH 15000
34
35 // サーボのぴくつきを防止するか (1=する、0=しない)
36 #define ANTI_SHAKING 1
37 #if ANTI_SHAKING
38 #define SERVO_MEAN_NUMBER 16
39 unsigned short elevator_history[SERVO_MEAN_NUMBER];
40 unsigned short throttle_history[SERVO_MEAN_NUMBER];
41 unsigned short ludder_history[SERVO_MEAN_NUMBER];
42 #endif
43
44
45 /* 処理ルーチン */
46
47 /**
48  * その他の初期化
```

```
 *
 */
void misc_init(){

  int i;

  /* サーボパルス幅の初期化 */
  for(i = 0; i < SERVO_CH; i++){
    output_pulse[i] = MIDDLE_PULSE_WIDTH;
    input_pulse[i] = MIDDLE_PULSE_WIDTH;
  }

  /* サーボ履歴の初期化 */
#if ANTI_SHAKING
  for(i = 0; i < SERVO_MEAN_NUMBER; i++){
    elevator_history[i] = MIDDLE_PULSE_WIDTH;
    throttle_history[i] = MIDDLE_PULSE_WIDTH;
    ludder_history[i] = MIDDLE_PULSE_WIDTH;
  }
#endif

  /* LED の有効化 */
  //IO.PCR8 = 0x03;
  IO.PCR8 = 0x01; // P81(LED2)はタイマW(FTIOA)で使用!!

  /* PORT50〜55 はサーボにつながる */
  IO.PCR5 = 0x3F;

  /* タイマW の有効化
   *
   * クロックはφ/2(0.1 us)
   * A => インプットキャプチャ (割り込み)
   * B => インプットキャプチャ (割り込み)
   * C => アウトプットコンペア
   * D => アウトプットコンペア
   */
  TW.TCRW.BYTE = 0x10;
  TW.TIOR0.BYTE = 0x47; // B => 立ち上がりエッジのみ検出
  TW.TIOR1.BYTE = 0x00;
  TW.TIERW.BYTE = 0x03;

  /* カウントスタート */
  TW.TMRW.BIT.CTS = 1;
}

#define servo_high(ch) (IO.PDR5.BYTE |= (0x01 << ch))
#define servo_low(ch) (IO.PDR5.BYTE &= ~(0x01 << ch))

unsigned char out_target_ch;

inline void start_pulse_timerw(unsigned char ch);

inline void stop_pulse_timerw(){

  /* ポートLow */
  servo_low(out_target_ch);

  /* 次のチャンネルへ */
  start_pulse_timerw(out_target_ch + 1);
}

inline void start_pulse_timerw(unsigned char ch){

  /* パラメータの設定 */
```

```
113   if((out_target_ch = ch) >= SERVO_CH){
114     /* 割り込み禁止 */
115     TW.TIERW.BIT.IMIEC = 0;
116     return;
117   }
118
119   TW.GRC = (TW.TCNT + output_pulse[ch]);
120
121   /* ポートHigh */
122   servo_high(ch);
123
124   /* コンペアマッチC で割り込みがかかるようにする */
125   if(ch == 0){
126     TW.TSRW.BIT.IMFC = 0;
127     TW.TIERW.BIT.IMIEC = 1;
128   }
129 }
130
131 #define BIT_IMFA 0x01
132 #define BIT_IMFB 0x02
133 #define BIT_IMFC 0x04
134 #define BIT_IMFD 0x08
135
136 inline void check_pulse(unsigned char flag){
137   static unsigned char target_ch = SERVO_CH;
138   static unsigned short before_count;
139
140   if(target_ch < SERVO_CH){
141     // 奇数 (1, 3, ...)
142     if(target_ch & 0x01){
143       if(flag & BIT_IMFA){
144         unsigned short pulse_width
145           = (TW.GRA > before_count) ? (TW.GRA - before_count) : ~(before_count - TW.GRA);
146         if(pulse_width < MAX_PULSE_WIDTH && pulse_width > MIN_PULSE_WIDTH){
147           input_pulse[target_ch] = pulse_width;
148           if(flag & BIT_IMFB){
149             before_count = TW.GRB;
150             target_ch++;
151             return;
152           }
153         }
154       }
155     // 偶数 (0, 2, ...)
156     }else{
157       if(flag & BIT_IMFB){
158         unsigned short pulse_width
159           = (TW.GRB > before_count) ? (TW.GRB - before_count) : ~(before_count - TW.GRB);
160         if(pulse_width < MAX_PULSE_WIDTH && pulse_width > MIN_PULSE_WIDTH){
161           input_pulse[target_ch] = pulse_width;
162           if(flag & BIT_IMFA){
163             before_count = TW.GRA;
164             target_ch++;
165             return;
166           }
167         }
168       }
169     }
170     target_ch = SERVO_CH;
171     TW.TIOR0.BYTE &= ~(0x30); // B => 立ち上がりエッジのみ検出
172   }else{
173     if((flag & (BIT_IMFA | BIT_IMFB)) == BIT_IMFB){
174       before_count = TW.GRB;
175       target_ch = 0;
176       TW.TIOR0.BYTE |= 0x30; // B => 両エッジ検出
```

```
177       }
178     }
179   }
180
181   #pragma interrupt
182   void int_timerw(){
183
184     unsigned char flag = TW.TSRW.BYTE;
185     TW.TSRW.BYTE = 0x00;
186
187     /* IMFC */
188     if(flag & BIT_IMFC){
189       stop_pulse_timerw();
190     }
191
192     /* IMFA, IMFB */
193     if(flag & (BIT_IMFA | BIT_IMFB)){
194       check_pulse(flag);
195     }
196   }
197
198   #define CALC_START 0x01
199   #define COMMAND_INIT 0x02
200   unsigned char status = 0;
201
202   /**
203    * タイマA インターバルによる呼び出しルーチン
204    * 6.4ms ごとに呼び出される
205    */
206   void call_every_6m4s(){
207     static unsigned char loop = 0;
208     switch(++loop){
209       case 2:
210         start_pulse_timerw(0);
211         break;
212       case 3:
213         status |= CALC_START;
214         //IO.PDR8.BIT.B1 = (IO.PDR8.BIT.B1 ? 0 : 1); // P81(LED2)はタイマW(FTIOA)で使用!!
215         loop = 0;
216     }
217   }
218
219   #define pulse_in_range(command) (max(min(command, MAX_PULSE_WIDTH), MIN_PULSE_WIDTH))
220   #define pulse_in_range2(command, range) (max(min(command, MIDDLE_PULSE_WIDTH + range),
          MIDDLE_PULSE_WIDTH − range))
221   #define pulse_in_range3(command, range) (max(min(command, range), − range))
222
223   /**
224    * 指令値の計算
225    *
226    * @param accel[]
227    *  3軸の加速度 0,1,2の順にX,Y,Z 軸、単位は mm/s^2、
228    * 例えば静止状態では
229    * accel[0],accel[1]には約 0(mm/s^2)
230    * accel[2]には約 9806(mm/s^2)
231    * がセットされている
232    *
233    * @param ultra_sonic[]
234    * 超音波センサから得られた距離 (未実装なので使わないこと)
235    */
236   inline void calc_command(short accel[3], unsigned char ultra_sonic[1]){
237
238     /*
239       * 補足説明
```

```
240     *
241     * 1.定義済み変数について
242     *
243     * 関数の外部で次の変数が定義されています。
244     * input_pulse 受信機より得られたパルス幅 (0〜5)
245     * output_pulse 実際に送信するパルス幅 (0〜5)
246     *
247     * パルス幅の単位は [0.1us]です。
248     * 例えば､双葉のサーボなら
249     * 15200[0.1us]が中心位置 (0度)で
250     * 20200[0.1us]が +90度
251     * 10200[0.1us]が−90度
252     *
253     * 詳しくは、
254     * http://berry.sakura.ne.jp/technics/servo_control_p3.html
255     * 等に情報があります。
256     *
257     * 例えば
258     * int i;
259     * for(i = 0; i < 6; i++){
260     * output_pulse[i] = input_pulse[i];
261     * }
262     * はマイコンが間に挿入されていないのと同じ状態、
263     * つまり受信機とサーボやスピコンを直繋ぎしたのと同じ状態
264     * を実現します。
265     *
266     * 2.コーディング上の注意
267     *
268     * 残念ながらマイコンの計算能力が高くないので、
269     * 整数演算を使用するようにしてください。
270     * 小数点演算や三角関数が入ると非常にスピードが落ちます。
271     * 例えば、0.25をかける場合は
272     * >> 2 (ビットシフト演算) や
273     * / 4 (整数演算)
274     * としてください。
275     * どうしても三角関数等を使用したい場合は相談してください。
276     *
277     * 3.この関数の呼び出し速度
278     *
279     * この関数は約 50Hz で呼び出されます。
280     */
281   static unsigned char calc_invoked = 0;
282   calc_invoked++;
283
284 #if ANTI_SHAKING
285   //ぴくつき防止用
286   static int elevator_sum = MIDDLE_PULSE_WIDTH * SERVO_MEAN_NUMBER;
287   static int throttle_sum = MIDDLE_PULSE_WIDTH * SERVO_MEAN_NUMBER;
288   static int ludder_sum = MIDDLE_PULSE_WIDTH * SERVO_MEAN_NUMBER;
289
290   unsigned char history_index = calc_invoked % SERVO_MEAN_NUMBER;
291 #endif
292
293   unsigned short elevator_command;
294   unsigned short ludder_command;
295   unsigned short throttle_command;
296
297   if(input_pulse[4] > MIDDLE_PULSE_WIDTH){
298     //自動
299     //エレベータ
300     {
301       //const short KE_1 = 1;
302       //const short KE_2 = 1;
303       /*
```

```
304         elevator_command = MIDDLE_PULSE_WIDTH − 500;
305         elevator_command += KE_1 ∗ accel[0];
306         elevator_command += KE_2 ∗ (accel[2] − SCALE_1G); //エレベータのパルス幅 [0.1us]
307         elevator_command = pulse_in_range2(elevator_command, 2000);
308         ∗/
309         elevator_command = input_pulse[1];
310       }
311
312       //スロットル
313       {
314
315         const short TARGET_HIGHT = 300; // 目標高度 [∗10 mm]
316         const short KT = 10; // スロットルゲイン
317         short h = ((20000 − input_pulse[2]) / 1000) ∗ 100; // プロポから送られてきた高度
318         throttle_command = pulse_in_range(18500 + KT ∗ ( TARGET_HIGHT − h )); // スロットルのパルス幅 [0.1us]
319
320         //throttle_command = input_pulse[2];
321       }
322
323       //ラダー
324       {
325         const short KR_1 = 2;
326         //const short KR_2 = 2;
327         ludder_command = MIDDLE_PULSE_WIDTH − 1000;
328         ludder_command −= accel[1] / KR_1;
329         //ludder_command = MIDDLE_PULSE_WIDTH;
330         //ludder_command += accel[1] / KR_1;  //ラダーのパルス幅 [0.1us]
331         //ludder_command += KR_2 ∗ (input_pulse[3] − MIDDLE_PULSE_WIDTH); //ラダーのパルス幅 [0.1us]
332         //ludder_command = pulse_in_range2(ludder_command, 3500);
333       }
334
335       output_pulse[0] = input_pulse[2];
336 #if ANTI_SHAKING
337       //ぴくつき防止用
338       output_pulse[1] = (elevator_sum / SERVO_MEAN_NUMBER);
339       output_pulse[2] = (throttle_sum / SERVO_MEAN_NUMBER);
340       output_pulse[3] = (ludder_sum / SERVO_MEAN_NUMBER);
341 #else
342       //こっちだとぴくつく
343       output_pulse[1] = elevator_command;
344       output_pulse[2] = throttle_command;
345       output_pulse[3] = ludder_command;
346 #endif
347       output_pulse[4] = input_pulse[0];
348       output_pulse[5] = input_pulse[1];
349     }else{
350       //手動
351       int i;
352       for(i = 0; i < 6; i++){
353         output_pulse[i] = input_pulse[i];
354       }
355
356       elevator_command = input_pulse[1];
357       throttle_command = input_pulse[2];
358       ludder_command = input_pulse[3];
359     }
360
361 #if ANTI_SHAKING
362     //ぴくつき防止用
363     elevator_sum −= elevator_history[history_index];
364     elevator_sum += (elevator_history[history_index] = elevator_command);
365     throttle_sum −= throttle_history[history_index];
366     throttle_sum += (throttle_history[history_index] = throttle_command);
367     ludder_sum −= ludder_history[history_index];
```

```
368   ludder_sum += (ludder_history[history_index] = ludder_command);
369
370 #define DUMP_COMMAND 0
371 #if DUMP_COMMAND
372   if(calc_invoked % 8 == 0){
373     char buf[32];
374     char *buf_index = buf;
375     buf_index = ushort2string(elevator_command, buf_index);
376     *(buf_index++) = ' ';
377     buf_index = ushort2string(throttle_command, buf_index);
378     *(buf_index++) = ' ';
379     buf_index = ushort2string(ludder_command, buf_index);
380     null_char(crlf(buf_index));
381     sci_dump(buf);
382   }
383 #endif
384 #endif
385 }
386
387 // 加速度計のゼロ点調整はこの値を変更すること
388 #define ACCEL_X_0G 32760
389 #define ACCEL_Y_0G 32500
390 #define ACCEL_Z_1G 22575
391
392 inline void polling(){
393   while(!(status & CALC_START));
394   status &= ~(CALC_START);
395
396   static unsigned char invoked_polling = 0;
397   invoked_polling++;
398
399   unsigned char i, j;
400
401   num_t accel_raw[3];
402
403   // AD 変換値の計算(16回して平均をとる)
404   {
405     num_t num_buf;
406     accel_raw[0] = accel_raw[1] = accel_raw[2] = 0;
407     for(i = 0; i < 16; i++){
408       ad_start();
409       for(j = 0; j < 3; j++){
410         while(!fifo_num_read(&(fifo_ad[j]), &num_buf));
411         accel_raw[j] += (num_buf >> 4);
412       }
413     }
414   }
415
416   short accel[3];
417
418   // 変換値の加工 (値がmm/s^2になるようにする)
419   {
420     /*
421      * ACCEL_1G 13107 // = (1 << 16) / 5
422      * EARTH_1G 9.80665
423      * ACCEL_1mm_s2 (4 / 3) // = (ACCEL_1G / (EARTH_1G * 1000)) = 1.33654204
424      * AD_RAW / ACCEL_1mm_s2 = accel [mm/s^2]
425      */
426     // X 軸
427     accel[0] = ((short)(accel_raw[0] >> 2) - (ACCEL_X_0G >> 2)) * 3;
428     // Y 軸
429     accel[1] = ((accel_raw[1] >> 2) - (short)(ACCEL_Y_0G >> 2)) * 3;
430     // Z 軸
431     accel[2] = (((ACCEL_Z_1G >> 2) - (short)(accel_raw[2] >> 2)) * 3) + SCALE_1G;
```

```
  }


#if DEBUG
  // 加速度計の値出力
  if(invoked_polling % 8 == 0){
    char buf[32];
    char *buf_index = buf;
    //null_char(ushort2string((unsigned short)invoked_polling, buf));
    for(i = 0; i < 3; i++){
      *(buf_index++) = i + '0';
      *(buf_index++) = ':';
      //buf_index = ushort2string(accel_raw[i], buf_index);
      buf_index = short2string(accel[i], buf_index);
      *(buf_index++) = ' ';
    }
    null_char(crlf(buf_index));
    sci_dump(buf);
  }
#endif

  // 超音波センサの値を読み込む
  unsigned char us[ULTRA_SONIC_CH];
  for(i = 0; i < ULTRA_SONIC_CH; i++){
    us[i] = us_read(i);
  }

#if DEBUG
  // パススルー
  for(i = 0; i < SERVO_CH; i++){
    output_pulse[i] = input_pulse[i];
  }
#else
  // ここに制御を埋め込むこと
  calc_command(accel, us);
#endif
}

/**
 * メイン関数
 *
 */
int main(){

  // Wait
  //int i;
  //for(i = 0; i < 100000; i++);

  // 初期設定
  misc_init();
  us_init();
  timer_init(call_every_6m4s);
  ad_init();
  sci3_init();
  wdt_init();

  enableIRQ(); /* 割り込み有効 */

  sci_dump("Test Program started!\r\n");

  wdt_start();

  while(1){
    polling();
```

```
496
497       wdt_reset(12); // WDT を 100ms で動作 20E6 / 8192 * 100 * 1E-3 = 244.14
498
499       IO.PDR8.BIT.B0 = (IO.PDR8.BIT.B0 ? 0 : 1);
500     }
501
502     disableIRQ(); /* 割り込み無効 */
503
504     return 0;
505   }
```

ソースコード 4　indoor/ultra_sonic.h

```
1  #ifndef __ULTRA_SONIC_H__
2  #define __ULTRA_SONIC_H__
3
4  /**
5   * 超音波センサユニットとの接続部の初期化
6   *
7   */
8  void us_init();
9
10 /**
11  * 超音波センサユニットから値を読み取る
12  *
13  */
14 unsigned char us_read(unsigned char ch);
15
16 #endif /* __ULTRA_SONIC_H__ */
```

ソースコード 5　indoor/ultra_sonic.c

```
1  #include <stdio.h>
2  #include <string.h>
3
4  /* 非依存ヘッダ */
5  #include "common.h"
6  #include "ultra_sonic.h"
7
8  /* 依存ヘッダ */
9  #include <AKI3694.h>
10
11 #define US_CHANNEL 3
12
13 /* 処理ルーチン */
14 #define select_low(ch) (IO.PDR1.BYTE &= ~(0x40 >> ch))
15 #define select_high(ch) (IO.PDR1.BYTE |= (0x40 >> ch))
16 #define clk_low() (IO.PDR5.BIT.B7 = 0)
17 #define clk_high() (IO.PDR5.BIT.B7 = 1)
18 #define is_data_high() (IO.PDR5.BIT.B6)
19
20 /**
21  * 超音波センサユニットとの接続部の初期化
22  *
23  */
24 void us_init(){
25
26   /*
27    * 接続は 3チャンネル分で
28    *
29    * P16〜P14 => SELECT(OUT)
30    * P56/SDA => DATA(IN)
31    * P57/SCL => CLK(OUT)
32    */
```

```
33
34    /* P14〜P16 の処理 */
35    IO.PCR1 |= 0x70;
36
37    /* P56,P57 の処理 */
38    IO.PCR5 |= 0x80;
39    IO.PCR5 &= ~(0x40);
40
41    /* SELECT と CLK は High にしておく */
42    IO.PDR1.BYTE |= 0x70;
43    IO.PDR5.BYTE |= 0x80;
44  }
45
46  /**
47   * 超音波センサユニットから値を読み取る
48   *
49   */
50  unsigned char us_read(unsigned char ch){
51
52    unsigned char mask = 0x80;
53    unsigned char result = 0x00;
54
55    select_low(ch);
56    do{
57      clk_low();
58      clk_high();
59      if(is_data_high()){result |= mask;}
60    }while(mask >>= 1);
61    select_high(ch);
62
63    return result;
64  }
```

ソースコード 6　framework/dependency/AKI3694/Makefile

```
1  CC = h8300-elf-gcc
2  NEWLIB_DIR = /usr/local/h8300-elf/lib/h8300h
3
4  CFLAGS ?= \
5    -I../.. -I. -I/usr/local/h8300-elf/include \
6    -Wall -mh -g -mint32 -mrelax -gdwarf-2
7
8  BUILD_DIR ?= build_by_gcc
9
10 SRCS_C = \
11   $(shell ls *.c)
12
13 OBJS = $(SRCS_C:.c=.o)
14
15 all : $(BUILD_DIR) $(patsubst %,$(BUILD_DIR)/%,$(OBJS))
16
17 $(BUILD_DIR)/%.o : %.c
18   $(CC) -c $(CFLAGS) -o $@ $<
19
20 $(BUILD_DIR) :
21   mkdir $@
22
23 clean :
24   cd $(BUILD_DIR); rm -f $(OBJS)
25
26 run : all
27
28 .PHONY : clean all
```

ソースコード 7　framework/dependency/AKI3694/rom.x

```
/*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−*/
/* rom.x
/* 秋月H8−3694用リンカスクリプト
/* @since 9July2006 fenrir
/*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−*/
OUTPUT_FORMAT("elf32-h8300")
OUTPUT_ARCH(h8300:h8300h)
ENTRY("_start")

/*−−−−− メモリマップ −−−−−*/
MEMORY
{
    vectors(r) : o = 0x0000, l = 0x34
    rom(r) : o = 0x0034, l = 32k − 0x34
    ram(rwx) : o = 0xf780, l = 2k
    stack(rwx) : o = 0xff7c, l = 0x0004
}

/*−−−−− 各セクション定義 −−−−−*/
SECTIONS
{

  /*リセットベクタと割り込みベクタ等の設定*/
  .vectors : {

  SHORT(ABSOLUTE(_start)) /* 0 リセット */
  SHORT(ABSOLUTE(_start)) /* 1 システム予約 1 */
  SHORT(ABSOLUTE(_start)) /* 2 システム予約 2 */
  SHORT(ABSOLUTE(_start)) /* 3 システム予約 3 */
  SHORT(ABSOLUTE(_start)) /* 4 システム予約 4 */
  SHORT(ABSOLUTE(_start)) /* 5 システム予約 5 */
  SHORT(ABSOLUTE(_start)) /* 6 システム予約 6 */
  SHORT(DEFINED(_int_nmi) ? ABSOLUTE(_int_nmi) : ABSOLUTE(_start)) /* 7 NMI(外部割込み) */
  SHORT(DEFINED(_int_trap0) ? ABSOLUTE(_int_trap0) : ABSOLUTE(_start)) /* 8 TRAPA#1 */
  SHORT(DEFINED(_int_trap1) ? ABSOLUTE(_int_trap1) : ABSOLUTE(_start)) /* 9 TRAPA#2 */
  SHORT(DEFINED(_int_trap2) ? ABSOLUTE(_int_trap2) : ABSOLUTE(_start)) /* 10 TRAPA#3 */
  SHORT(DEFINED(_int_trap3) ? ABSOLUTE(_int_trap3) : ABSOLUTE(_start)) /* 11 TRAPA#4 */
  SHORT(ABSOLUTE(_start)) /* 12 ブレーク条件成立 */
  SHORT(ABSOLUTE(_start)) /* 13 スリープ命令の実行による直接遷移 */
  SHORT(DEFINED(_int_irq0) ? ABSOLUTE(_int_irq0) : ABSOLUTE(_start)) /* 14 IRQ0 */
  SHORT(DEFINED(_int_irq1) ? ABSOLUTE(_int_irq1) : ABSOLUTE(_start)) /* 15 IRQ1 */
  SHORT(DEFINED(_int_irq2) ? ABSOLUTE(_int_irq2) : ABSOLUTE(_start)) /* 16 IRQ2 */
  SHORT(DEFINED(_int_irq3) ? ABSOLUTE(_int_irq3) : ABSOLUTE(_start)) /* 17 IRQ3 */
  SHORT(DEFINED(_int_wkp) ? ABSOLUTE(_int_wkp) : ABSOLUTE(_start)) /* 18 WKP */
  SHORT(DEFINED(_int_timera) ? ABSOLUTE(_int_timera) : ABSOLUTE(_start)) /* 19 タイマA */
  SHORT(ABSOLUTE(_start)) /* 20 システム予約 7 */
  SHORT(DEFINED(_int_timerw) ? ABSOLUTE(_int_timerw) : ABSOLUTE(_start)) /* 21 タイマW */
  SHORT(DEFINED(_int_timerv) ? ABSOLUTE(_int_timerv) : ABSOLUTE(_start)) /* 22 タイマV */
  SHORT(DEFINED(_int_sci3) ? ABSOLUTE(_int_sci3) : ABSOLUTE(_start)) /* 23 SCI3 */
  SHORT(DEFINED(_int_iic2) ? ABSOLUTE(_int_iic2) : ABSOLUTE(_start)) /* 24 IIC2 */
  SHORT(DEFINED(_int_ad) ? ABSOLUTE(_int_ad) : ABSOLUTE(_start)) /* 25 A/D 変換終了 */

  } > vectors

  /* プログラム領域＆参照のみのデータ領域 */
  .text : {
    _text_begin = . ;
    *(.text) /* プログラム領域 */
    *(.rodata) /* コンスタント領域 (read only) */
    *(.rodata.*)
    *(.gnu.linkonce.r.*)
    *(.gnu.linkonce.t.*)
```

```
63      _text_end = . ;
64    } > rom
65
66    .tors : {
67      _ctors_begin = . ;
68      *(.ctors)
69      _ctors_end = . ;
70      _dtors_begin = . ;
71      *(.dtors)
72      _dtors_end = . ;
73    } > rom
74
75    /* 初期値を持つグローバル変数, スタティック変数領域 */
76    .data : AT(ADDR(.tors) + SIZEOF(.tors)){
77          _data_begin = . ;
78          *(.strings) /* 文字列 */
79          *(.data)
80          *(.tiny)
81          _data_end = . ;
82    } > ram
83
84    /* 初期値を持たないグローバル変数, スタティック変数領域 */
85    .bss : {
86      _bss_begin = . ;
87      *(.bss)
88      *(COMMON)
89      _bss_end = . ;
90      _end = . ;
91    } > ram
92
93    /* スタック領域 */
94    .stack : {
95      _initial_stack_point = . +4 ;
96      _stack = . ;
97      *(.stack)
98    } > stack
99
100   .eight : {
101     *(.eight)
102   } > ram
103
104   /* デバックセクション */
105   /* Stabs */
106   .stab 0 (NOLOAD) : { *(.stab) }
107   .stabstr 0 (NOLOAD) : { *(.stabstr) }
108   .stab.excl 0 (NOLOAD) : { *(.stab.excl) }
109   .stab.exclstr 0 (NOLOAD) : { *(.stab.exclstr) }
110   .stab.index 0 (NOLOAD) : { *(.stab.index) }
111   .stab.indexstr 0 (NOLOAD) : { *(.stab.indexstr) }
112   .comment 0 (NOLOAD) : { *(.comment) }
113   /* DWARF 1 */
114   .debug 0 (NOLOAD) : { *(.debug) }
115   .line 0 (NOLOAD) : { *(.line) }
116   /* GNU DWARF 1 拡張 */
117   .debug_srcinfo 0 (NOLOAD) : { *(.debug_srcinfo) }
118   .debug_sfnames 0 (NOLOAD) : { *(.debug_sfnames) }
119   /* DWARF 1.1 & DWARF 2 */
120   .debug_aranges 0 (NOLOAD) : { *(.debug_aranges) }
121   .debug_pubnames 0 (NOLOAD) : { *(.debug_pubnames) }
122   /* DWARF 2 */
123   .debug_info 0 (NOLOAD) : { *(.debug_info .gnu.linkonce.wi.*) }
124   .debug_abbrev 0 (NOLOAD) : { *(.debug_abbrev) }
125   .debug_line 0 (NOLOAD) : { *(.debug_line) }
126   .debug_frame 0 (NOLOAD) : { *(.debug_frame) }
```

```
  .debug_str 0 (NOLOAD) : { *(.debug_str) }
  .debug_loc 0 (NOLOAD) : { *(.debug_loc) }
}
```

ソースコード 8　framework/dependency/AKI3694/romcrt0.S

```
  .h8300h
  .section .text
  .global _start

_start:

  /* スタックポインタ設定 */
  mov.l #_initial_stack_point,sp

  /* 初期値 0の領域 (.bss)の 0クリア */
  mov.l #_bss_begin,er0
  mov.l #_bss_end,er1
  sub.w r2,r2
.loop:
  mov.w r2,@er0
  adds #2,er0
  cmp.l er1,er0
  blo .loop

  /* 初期値付データ領域 (.data)の書き込み */
  mov.l #_dtors_end,er0
  mov.l #_data_begin,er1
  mov.l #_data_end,er2
.loop2:
  mov.w @er0,r3
  mov.w r3,@er1
  adds #2,er0
  adds #2,er1
  cmp.l er2,er1
  blo .loop2

/* main()コール */
  jsr @_main
  rts
```

ソースコード 9　framework/dependency/AKI3694/3694.h

```
/************************************************************************/
/* H8/3694 Series Include File Ver 2.1 */
/* from Renesas */
/************************************************************************/

#ifndef __3694_H__
#define __3694_H__

typedef union{
  struct{
    unsigned char high;
    unsigned char low;
  } u8;
  unsigned short u16;
} uc16_t;

struct st_lvd { /* struct LVD */
        union { /* LVDCR */
                unsigned char BYTE; /* Byte Access */
                struct { /* Bit Access */
                        unsigned char LVDE :1; /* LVDE */
```

```
                                unsigned char :3; /* */
                                unsigned char LVDSEL:1; /* LVDSEL */
                                unsigned char LVDRE :1; /* LVDRE */
                                unsigned char LVDDE :1; /* LVDDE */
                                unsigned char LVDUE :1; /* LVDUE */
                                } BIT; /* */
                        } CR; /* */
                union { /* LVDSR */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char :6; /* */
                                unsigned char LVDDF:1; /* LVDDF */
                                unsigned char LVDUF:1; /* LVDUF */
                                } BIT; /* */
                        } SR; /* */
};/* */
struct st_iic2 { /* struct IIC2 */
                union { /* ICCR1 */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char ICE :1; /* ICE */
                                unsigned char RCVD:1; /* RCVD */
                                unsigned char MST :1; /* MST */
                                unsigned char TRS :1; /* TRS */
                                unsigned char CKS :4; /* CKS */
                                } BIT; /* */
                        } ICCR1; /* */
                union { /* ICCR2 */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char BBSY :1; /* BBSY */
                                unsigned char SCP :1; /* SCP */
                                unsigned char SDAO :1; /* SDAO */
                                unsigned char SDAOP :1; /* SDAOP */
                                unsigned char SCLO :1; /* SCLO */
                                unsigned char :1; /* */
                                unsigned char IICRST:1; /* IICRST */
                                } BIT; /* */
                        } ICCR2; /* */
                union { /* ICMR */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char MLS :1; /* MLS */
                                unsigned char WAIT:1; /* WAIT */
                                unsigned char :2; /* */
                                unsigned char BCWP:1; /* BCWP */
                                unsigned char BC :3; /* BC */
                                } BIT; /* */
                        } ICMR; /* */
                union { /* ICIER */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char TIE :1; /* TIE */
                                unsigned char TEIE :1; /* TEIE */
                                unsigned char RIE :1; /* RIE */
                                unsigned char NAKIE:1; /* NAKIE */
                                unsigned char STIE :1; /* STIE */
                                unsigned char ACKE :1; /* ACKE */
                                unsigned char ACKBR:1; /* ACKBR */
                                unsigned char ACKBT:1; /* ACKBT */
                                } BIT; /* */
                        } ICIER; /* */
                union { /* ICSR */
                        unsigned char BYTE; /* Byte Access */
```

```
                        struct { /* Bit Access */
                                unsigned char TDRE :1; /* TDRE */
                                unsigned char TEND :1; /* TEND */
                                unsigned char RDRF :1; /* RDRF */
                                unsigned char NACKF:1; /* NACKF */
                                unsigned char STOP :1; /* STOP */
                                unsigned char ALOVE:1; /* ALOVE */
                                unsigned char AAS :1; /* AAS */
                                unsigned char ADZ :1; /* ADZ */
                                } BIT; /* */
                        } ICSR; /* */
                union { /* SAR */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char SVA:7; /* SVA */
                                unsigned char FS :1; /* FS */
                                } BIT; /* */
                        } SAR; /* */
                unsigned char ICDRT; /* ICDRT */
                unsigned char ICDRR; /* ICDRR */
};/* */
struct st_tw { /* struct TW */
                union { /* TMRW */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char CTS :1; /* CTS */
                                unsigned char :1; /* */
                                unsigned char BUFEB:1; /* BUFEB */
                                unsigned char BUFEA:1; /* BUFEA */
                                unsigned char :1; /* */
                                unsigned char PWMD :1; /* PWMD */
                                unsigned char PWMC :1; /* PWMC */
                                unsigned char PWMB :1; /* PWMB */
                                } BIT; /* */
                        } TMRW; /* */
                union { /* TCRW */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char CCLR:1; /* CCLR */
                                unsigned char CKS :3; /* CKS */
                                unsigned char TOD :1; /* TOD */
                                unsigned char TOC :1; /* TOC */
                                unsigned char TOB :1; /* TOB */
                                unsigned char TOA :1; /* TOA */
                                } BIT; /* */
                        } TCRW; /* */
                union { /* TIERW */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char OVIE :1; /* OVIE */
                                unsigned char :3; /* */
                                unsigned char IMIED:1; /* IMIED */
                                unsigned char IMIEC:1; /* IMIEC */
                                unsigned char IMIEB:1; /* IMIEB */
                                unsigned char IMIEA:1; /* IMIEA */
                                } BIT; /* */
                        } TIERW; /* */
                union { /* TSRW */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char OVF :1; /* OVF */
                                unsigned char :3; /* */
                                unsigned char IMFD:1; /* IMFD */
                                unsigned char IMFC:1; /* IMFC */
```

```
150                             unsigned char IMFB:1; /* IMFB */
151                             unsigned char IMFA:1; /* IMFA */
152                             } BIT; /* */
153                     } TSRW; /* */
154             union { /* TIOR0 */
155                     unsigned char BYTE; /* Byte Access */
156                     struct { /* Bit Access */
157                             unsigned char :1; /* */
158                             unsigned char IOB:3; /* IOB */
159                             unsigned char :1; /* */
160                             unsigned char IOA:3; /* IOA */
161                             } BIT; /* */
162                     } TIOR0; /* */
163             union { /* TIOR1 */
164                     unsigned char BYTE; /* Byte Access */
165                     struct { /* Bit Access */
166                             unsigned char :1; /* */
167                             unsigned char IOD:3; /* IOD */
168                             unsigned char :1; /* */
169                             unsigned char IOC:3; /* IOC */
170                             } BIT; /* */
171                     } TIOR1; /* */
172             /*unsigned int TCNT;(-mint32) *//* TCNT */
173             unsigned short TCNT;
174             /*unsigned int GRA;(-mint32) *//* GRA */
175             unsigned short GRA;
176             /*unsigned int GRB;(-mint32) *//* GRB */
177             unsigned short GRB;
178             /*unsigned int GRC;(-mint32) *//* GRC */
179             unsigned short GRC;
180             /*unsigned int GRD;(-mint32) *//* GRD */
181             unsigned short GRD;
182 }; /* */
183 struct st_flash { /* struct FLASH */
184                     union { /* FLMCR1 */
185                             unsigned char BYTE; /* Byte Access */
186                             struct { /* Bit Access */
187                                     unsigned char :1; /* */
188                                     unsigned char SWE:1; /* SWE */
189                                     unsigned char ESU:1; /* ESU */
190                                     unsigned char PSU:1; /* PSU */
191                                     unsigned char EV :1; /* EV */
192                                     unsigned char PV :1; /* PV */
193                                     unsigned char E :1; /* E */
194                                     unsigned char P :1; /* P */
195                                     } BIT; /* */
196                             } FLMCR1; /* */
197                     union { /* FLMCR2 */
198                             unsigned char BYTE; /* Byte Access */
199                             struct { /* Bit Access */
200                                     unsigned char FLER:1; /* FLER */
201                                     } BIT; /* */
202                             } FLMCR2; /* */
203                     union { /* FLPWCR */
204                             unsigned char BYTE; /* Byte Access */
205                             struct { /* Bit Access */
206                                     unsigned char PDWND:1; /* PDWND */
207                                     } BIT; /* */
208                             } FLPWCR; /* */
209                     union { /* EBR1 */
210                             unsigned char BYTE; /* Byte Access */
211                             struct { /* Bit Access */
212                                     unsigned char :3; /* */
213                                     unsigned char EB4:1; /* EB4 */
```

```
                                unsigned char EB3:1; /* EB3 */
                                unsigned char EB2:1; /* EB2 */
                                unsigned char EB1:1; /* EB1 */
                                unsigned char EB0:1; /* EB0 */
                                } BIT; /* */
                        } EBR1; /* */
                char wk[7]; /* */
                union { /* FENR */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char FLSHE:1; /* FLSHE */
                                } BIT; /* */
                        } FENR; /* */
};/* */
struct st_tv { /* struct TV */
                union { /* TCRV0 */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char CMIEB:1; /* CMIEB */
                                unsigned char CMIEA:1; /* CMIEA */
                                unsigned char OVIE :1; /* OVIE */
                                unsigned char CCLR :2; /* CCLR */
                                unsigned char CKS :3; /* CKS */
                                } BIT; /* */
                        } TCRV0; /* */
                union { /* TCSRV */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char CMFB:1; /* CMFB */
                                unsigned char CMFA:1; /* CMFA */
                                unsigned char OVF :1; /* OVF */
                                unsigned char :1; /* */
                                unsigned char OS :4; /* OS */
                                } BIT; /* */
                        } TCSRV; /* */
                unsigned char TCORA; /* TCORA */
                unsigned char TCORB; /* TCORB */
                unsigned char TCNTV; /* TCNT */
                union { /* TCRV1 */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char :3; /* */
                                unsigned char TVEG:2; /* TVEG */
                                unsigned char TRGE:1; /* TRGE */
                                unsigned char :1; /* */
                                unsigned char ICKS:1; /* ICKS */
                                } BIT; /* */
                        } TCRV1; /* */
};/* */
struct st_ta { /* struct TA */
                union { /* TMA */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
                                unsigned char CKSO:3; /* CKSO */
                                unsigned char :1; /* */
                                unsigned char CKSI:4; /* CKSI */
                                } BIT; /* */
                        } TMA; /* */
                unsigned char TCA; /* TCA */
};/* */
struct st_sci3 { /* struct SCI3 */
                union { /* SMR */
                        unsigned char BYTE; /* Byte Access */
                        struct { /* Bit Access */
```

```
278                                     unsigned char COM :1; /* COM */
279                                     unsigned char CHR :1; /* CHR */
280                                     unsigned char PE :1; /* PE */
281                                     unsigned char PM :1; /* PM */
282                                     unsigned char STOP:1; /* STOP */
283                                     unsigned char MP :1; /* MP */
284                                     unsigned char CKS :2; /* CKS */
285                                     } BIT; /* */
286                             } SMR; /* */
287                     unsigned char BRR; /* BRR */
288                     union { /* SCR3 */
289                             unsigned char BYTE; /* Byte Access */
290                             struct { /* Bit Access */
291                                     unsigned char TIE :1; /* TIE */
292                                     unsigned char RIE :1; /* RIE */
293                                     unsigned char TE :1; /* TE */
294                                     unsigned char RE :1; /* RE */
295                                     unsigned char MPIE:1; /* MPIE */
296                                     unsigned char TEIE:1; /* TEIE */
297                                     unsigned char CKE :2; /* CKE */
298                                     } BIT; /* */
299                             } SCR3; /* */
300                     unsigned char TDR; /* TDR */
301                     union { /* SSR */
302                             unsigned char BYTE; /* Byte Access */
303                             struct { /* Bit Access */
304                                     unsigned char TDRE:1; /* TDRE */
305                                     unsigned char RDRF:1; /* RDRF */
306                                     unsigned char OER :1; /* OER */
307                                     unsigned char FER :1; /* FER */
308                                     unsigned char PER :1; /* PER */
309                                     unsigned char TEND:1; /* TEND */
310                                     unsigned char MPBR:1; /* MPBR */
311                                     unsigned char MPBT:1; /* MPBT */
312                                     } BIT; /* */
313                             } SSR; /* */
314                     unsigned char RDR; /* RDR */
315 }; /* */
316 struct st_ad { /* struct A/D */
317                     /*unsigned int ADDRA;(-mint32) *//* ADDRA */
318                     uc16_t ADDRA;
319                     /*unsigned int ADDRB;(-mint32) *//* ADDRB */
320                     uc16_t ADDRB;
321                     /*unsigned int ADDRC;(-mint32) *//* ADDRC */
322                     uc16_t ADDRC;
323                     /*unsigned int ADDRD;(-mint32) *//* ADDRD */
324                     uc16_t ADDRD;
325                     union { /* ADCSR */
326                             unsigned char BYTE; /* Byte Access */
327                             struct { /* Bit Access */
328                                     unsigned char ADF :1; /* ADF */
329                                     unsigned char ADIE:1; /* ADIE */
330                                     unsigned char ADST:1; /* ADST */
331                                     unsigned char SCAN:1; /* SCAN */
332                                     unsigned char CKS :1; /* CKS */
333                                     unsigned char CH :3; /* CH */
334                                     } BIT; /* */
335                             } ADCSR; /* */
336                     union { /* ADCR */
337                             unsigned char BYTE; /* Byte Access */
338                             struct { /* Bit Access */
339                                     unsigned char TRGE:1; /* TRGE */
340                                     } BIT; /* */
341                             } ADCR; /* */
```

```
342 }; /* */
343 struct st_wdt { /* struct WDT */
344                     union { /* TCSRWD */
345                             unsigned char BYTE; /* Byte Access */
346                             struct { /* Bit Access */
347                                     unsigned char B6WI :1; /* B6WI */
348                                     unsigned char TCWE :1; /* TCWE */
349                                     unsigned char B4WI :1; /* B4WI */
350                                     unsigned char TCSRWE:1; /* TCSRWE */
351                                     unsigned char B2WI :1; /* B2WI */
352                                     unsigned char WDON :1; /* WDON */
353                                     unsigned char B0WI :1; /* B0WI */
354                                     unsigned char WRST :1; /* WRST */
355                                     } BIT; /* */
356                             } TCSRWD; /* */
357                     unsigned char TCWD; /* TCWD */
358                     union { /* TMWD */
359                             unsigned char BYTE; /* Byte Access */
360                             struct { /* Bit Access */
361                                     unsigned char :4; /* */
362                                     unsigned char CKS:4; /* CKS */
363                                     } BIT; /* */
364                             } TMWD; /* */
365 }; /* */
366 struct st_abrk { /* struct ABRK */
367                       union { /* ABRKCR */
368                               unsigned char BYTE; /* Byte Access */
369                               struct { /* Bit Access */
370                                       unsigned char RTINTE:1; /* RTINTE */
371                                       unsigned char CSEL :2; /* CSEL */
372                                       unsigned char ACMP :3; /* ACMP */
373                                       unsigned char DCMP :2; /* DCMP */
374                                       } BIT; /* */
375                               } CR; /* */
376                       union { /* ABRKSR */
377                               unsigned char BYTE; /* Byte Access */
378                               struct { /* Bit Access */
379                                       unsigned char ABIF:1; /* ABIF */
380                                       unsigned char ABIE:1; /* ABIE */
381                                       } BIT; /* */
382                               } SR; /* */
383                       void *BAR; /* BAR */
384                       /*unsigned int BDR;(-mint32) *//* BDR */
385                       unsigned char BDR_H;
386                       unsigned char BDR_L;
387 }; /* */
388 struct st_io { /* struct IO */
389                     union { /* PUCR1 */
390                             unsigned char BYTE; /* Byte Access */
391                             struct { /* Bit Access */
392                                     unsigned char B7:1; /* Bit 7 */
393                                     unsigned char B6:1; /* Bit 6 */
394                                     unsigned char B5:1; /* Bit 5 */
395                                     unsigned char B4:1; /* Bit 4 */
396                                     unsigned char :1; /* Bit 3 */
397                                     unsigned char B2:1; /* Bit 2 */
398                                     unsigned char B1:1; /* Bit 1 */
399                                     unsigned char B0:1; /* Bit 0 */
400                                     } BIT; /* */
401                             } PUCR1; /* */
402                     union { /* PUCR5 */
403                             unsigned char BYTE; /* Byte Access */
404                             struct { /* Bit Access */
405                                     unsigned char :2; /* Bit 7,6 */
```

```
                        unsigned char B5:1; /* Bit 5 */
                        unsigned char B4:1; /* Bit 4 */
                        unsigned char B3:1; /* Bit 3 */
                        unsigned char B2:1; /* Bit 2 */
                        unsigned char B1:1; /* Bit 1 */
                        unsigned char B0:1; /* Bit 0 */
                        } BIT; /* */
                } PUCR5; /* */
        char wk1[2]; /* */
        union { /* PDR1 */
                unsigned char BYTE; /* Byte Access */
                struct { /* Bit Access */
                        unsigned char B7:1; /* Bit 7 */
                        unsigned char B6:1; /* Bit 6 */
                        unsigned char B5:1; /* Bit 5 */
                        unsigned char B4:1; /* Bit 4 */
                        unsigned char :1; /* Bit 3 */
                        unsigned char B2:1; /* Bit 2 */
                        unsigned char B1:1; /* Bit 1 */
                        unsigned char B0:1; /* Bit 0 */
                        } BIT; /* */
                } PDR1; /* */
        union { /* PDR2 */
                unsigned char BYTE; /* Byte Access */
                struct { /* Bit Access */
                        unsigned char :5; /* Bit 7-3 */
                        unsigned char B2:1; /* Bit 2 */
                        unsigned char B1:1; /* Bit 1 */
                        unsigned char B0:1; /* Bit 0 */
                        } BIT; /* */
                } PDR2; /* */
        char wk2[2]; /* */
        union { /* PDR5 */
                unsigned char BYTE; /* Byte Access */
                struct { /* Bit Access */
                        unsigned char B7:1; /* Bit 7 */
                        unsigned char B6:1; /* Bit 6 */
                        unsigned char B5:1; /* Bit 5 */
                        unsigned char B4:1; /* Bit 4 */
                        unsigned char B3:1; /* Bit 3 */
                        unsigned char B2:1; /* Bit 2 */
                        unsigned char B1:1; /* Bit 1 */
                        unsigned char B0:1; /* Bit 0 */
                        } BIT; /* */
                } PDR5; /* */
        char wk3; /* */
        union { /* PDR7 */
                unsigned char BYTE; /* Byte Access */
                struct { /* Bit Access */
                        unsigned char :1; /* Bit 7 */
                        unsigned char B6:1; /* Bit 6 */
                        unsigned char B5:1; /* Bit 5 */
                        unsigned char B4:1; /* Bit 4 */
                        } BIT; /* */
                } PDR7; /* */
        union { /* PDR8 */
                unsigned char BYTE; /* Byte Access */
                struct { /* Bit Access */
                        unsigned char B7:1; /* Bit 7 */
                        unsigned char B6:1; /* Bit 6 */
                        unsigned char B5:1; /* Bit 5 */
                        unsigned char B4:1; /* Bit 4 */
                        unsigned char B3:1; /* Bit 3 */
                        unsigned char B2:1; /* Bit 2 */
```

```
470                             unsigned char B1:1; /* Bit 1 */
471                             unsigned char B0:1; /* Bit 0 */
472                             } BIT; /* */
473                     } PDR8; /* */
474             char wk4; /* */
475             union { /* PDRB */
476                     unsigned char BYTE; /* Byte Access */
477                     struct { /* Bit Access */
478                             unsigned char B7:1; /* Bit 7 */
479                             unsigned char B6:1; /* Bit 6 */
480                             unsigned char B5:1; /* Bit 5 */
481                             unsigned char B4:1; /* Bit 4 */
482                             unsigned char B3:1; /* Bit 3 */
483                             unsigned char B2:1; /* Bit 2 */
484                             unsigned char B1:1; /* Bit 1 */
485                             unsigned char B0:1; /* Bit 0 */
486                             } BIT; /* */
487                     } PDRB; /* */
488             char wk5[2]; /* */
489             union { /* PMR1 */
490                     unsigned char BYTE; /* Byte Access */
491                     struct { /* Bit Access */
492                             unsigned char IRQ3:1; /* IRQ3 */
493                             unsigned char IRQ2:1; /* IRQ2 */
494                             unsigned char IRQ1:1; /* IRQ1 */
495                             unsigned char IRQ0:1; /* IRQ0 */
496                             unsigned char :2; /* */
497                             unsigned char TXD :1; /* TXD */
498                             unsigned char TMOW:1; /* TMOW */
499                             } BIT; /* */
500                     } PMR1; /* */
501             union { /* PMR5 */
502                     unsigned char BYTE; /* Byte Access */
503                     struct { /* Bit Access */
504                             unsigned char :2; /* */
505                             unsigned char WKP5:1; /* WKP5 */
506                             unsigned char WKP4:1; /* WKP4 */
507                             unsigned char WKP3:1; /* WKP3 */
508                             unsigned char WKP2:1; /* WKP2 */
509                             unsigned char WKP1:1; /* WKP1 */
510                             unsigned char WKP0:1; /* WKP0 */
511                             } BIT; /* */
512                     } PMR5; /* */
513             char wk6[2]; /* */
514             unsigned char PCR1; /* PCR1 */
515             unsigned char PCR2; /* PCR2 */
516             char wk7[2]; /* */
517             unsigned char PCR5; /* PCR5 */
518             char wk8; /* */
519             unsigned char PCR7; /* PCR7 */
520             unsigned char PCR8; /* PCR8 */
521 }; /* */
522 union un_syscr1 { /* union SYSCR1 */
523                     unsigned char BYTE; /* Byte Access */
524                     struct { /* Bit Access */
525                             unsigned char SSBY :1; /* SSBY */
526                             unsigned char STS :3; /* STS */
527                             unsigned char NESEL:1; /* NESEL */
528                             } BIT; /* */
529 }; /* */
530 union un_syscr2 { /* union SYSCR2 */
531                     unsigned char BYTE; /* Byte Access */
532                     struct { /* Bit Access */
533                             unsigned char SMSEL:1; /* SMSEL */
```

```
534                             unsigned char LSON :1; /* LSON */
535                             unsigned char DTON :1; /* DTON */
536                             unsigned char MA :3; /* MA */
537                             unsigned char SA :2; /* SA */
538                             } BIT; /* */
539 }; /* */
540 union un_iegr1 { /* union IEGR1 */
541                     unsigned char BYTE; /* Byte Access */
542                     struct { /* Bit Access */
543                             unsigned char NMIEG:1; /* NMIEG */
544                             unsigned char :3; /* */
545                             unsigned char IEG3 :1; /* IEG3 */
546                             unsigned char IEG2 :1; /* IEG2 */
547                             unsigned char IEG1 :1; /* IEG1 */
548                             unsigned char IEG0 :1; /* IEG0 */
549                             } BIT; /* */
550 }; /* */
551 union un_iegr2 { /* union IEGR2 */
552                     unsigned char BYTE; /* Byte Access */
553                     struct { /* Bit Access */
554                             unsigned char :2; /* */
555                             unsigned char WPEG5:1; /* WPEG5 */
556                             unsigned char WPEG4:1; /* WPEG4 */
557                             unsigned char WPEG3:1; /* WPEG3 */
558                             unsigned char WPEG2:1; /* WPEG2 */
559                             unsigned char WPEG1:1; /* WPEG1 */
560                             unsigned char WPEG0:1; /* WPEG0 */
561                             } BIT; /* */
562 }; /* */
563 union un_ienr1 { /* union IENR1 */
564                     unsigned char BYTE; /* Byte Access */
565                     struct { /* Bit Access */
566                             unsigned char IENDT:1; /* IENDT */
567                             unsigned char IENTA:1; /* IENTA */
568                             unsigned char IENWP:1; /* IENWP */
569                             unsigned char :1; /* */
570                             unsigned char IEN3 :1; /* IEN3 */
571                             unsigned char IEN2 :1; /* IEN2 */
572                             unsigned char IEN1 :1; /* IEN1 */
573                             unsigned char IEN0 :1; /* IEN0 */
574                             } BIT; /* */
575 }; /* */
576 union un_irr1 { /* union IRR1 */
577                     unsigned char BYTE; /* Byte Access */
578                     struct { /* Bit Access */
579                             unsigned char IRRDT:1; /* IRRDT */
580                             unsigned char IRRTA:1; /* IRRTA */
581                             unsigned char :2; /* */
582                             unsigned char IRRI3:1; /* IRRI3 */
583                             unsigned char IRRI2:1; /* IRRI2 */
584                             unsigned char IRRI1:1; /* IRRI1 */
585                             unsigned char IRRI0:1; /* IRRI0 */
586                             } BIT; /* */
587 }; /* */
588 union un_iwpr { /* union IWPR */
589                     unsigned char BYTE; /* Byte Access */
590                     struct { /* Bit Access */
591                             unsigned char :2; /* */
592                             unsigned char IWPF5:1; /* IWPF5 */
593                             unsigned char IWPF4:1; /* IWPF4 */
594                             unsigned char IWPF3:1; /* IWPF3 */
595                             unsigned char IWPF2:1; /* IWPF2 */
596                             unsigned char IWPF1:1; /* IWPF1 */
597                             unsigned char IWPF0:1; /* IWPF0 */
```

```
598                             } BIT; /* */
599 }; /* */
600 union un_mstcr1 { /* union MSTCR1 */
601                         unsigned char BYTE; /* Byte Access */
602                         struct { /* Bit Access */
603                                 unsigned char :1; /* */
604                                 unsigned char MSTIIC:1; /* MSTIIC */
605                                 unsigned char MSTS3 :1; /* MSTS3 */
606                                 unsigned char MSTAD :1; /* MSTAD */
607                                 unsigned char MSTWD :1; /* MSTWD */
608                                 unsigned char MSTTW :1; /* MSTTW */
609                                 unsigned char MSTTV :1; /* MSTTV */
610                                 unsigned char MSTTA :1; /* MSTTA */
611                                 } BIT; /* */
612 }; /* */
613 #define LVD (*(volatile struct st_lvd *)0xF730) /* LVD Address*/
614 #define IIC2 (*(volatile struct st_iic2 *)0xF748) /* IIC2 Address*/
615 #define TW (*(volatile struct st_tw *)0xFF80) /* TW Address*/
616 #define FLASH (*(volatile struct st_flash *)0xFF90) /* FLASH Address*/
617 #define TV (*(volatile struct st_tv *)0xFFA0) /* TV Address*/
618 #define TA (*(volatile struct st_ta *)0xFFA6) /* TA Address*/
619 #define SCI3 (*(volatile struct st_sci3 *)0xFFA8) /* SCI3 Address*/
620 #define AD (*(volatile struct st_ad *)0xFFB0) /* A/D Address*/
621 #define WDT (*(volatile struct st_wdt *)0xFFC0) /* WDT Address*/
622 #define ABRK (*(volatile struct st_abrk *)0xFFC8) /* ABRK Address*/
623 #define IO (*(volatile struct st_io *)0xFFD0) /* IO Address*/
624 #define SYSCR1 (*(volatile union un_syscr1*)0xFFF0) /* SYSCR1Address*/
625 #define SYSCR2 (*(volatile union un_syscr2*)0xFFF1) /* SYSCR2Address*/
626 #define IEGR1 (*(volatile union un_iegr1 *)0xFFF2) /* IEGR1 Address*/
627 #define IEGR2 (*(volatile union un_iegr2 *)0xFFF3) /* IEGR2 Address*/
628 #define IENR1 (*(volatile union un_ienr1 *)0xFFF4) /* IENR1 Address*/
629 #define IRR1 (*(volatile union un_irr1 *)0xFFF6) /* IRR1 Address*/
630 #define IWPR (*(volatile union un_iwpr *)0xFFF8) /* IWPR Address*/
631 #define MSTCR1 (*(volatile union un_mstcr1*)0xFFF9) /* MSTCR1Address*/
632
633 #endif /*__3694_H__*/
```

ソースコード 10　framework/dependency/AKI3694/AKI3694.h

```
1  #ifndef __AKI3694_H__
2  #define __AKI3694_H__
3
4  #include "3694.h"
5  //#define printf ((int (*)(const char *,...))0x00002bb0)
6  //#define scanf ((int (*)(const char *,...))0x00002c02)
7
8  #define AD_MAX 0xffc0
9
10 #define PORT0 (unsigned char)0x01
11 #define PORT1 (unsigned char)0x02
12 #define PORT2 (unsigned char)0x04
13 #define PORT3 (unsigned char)0x08
14 #define PORT4 (unsigned char)0x10
15 #define PORT5 (unsigned char)0x20
16 #define PORT6 (unsigned char)0x40
17 #define PORT7 (unsigned char)0x80
18
19 #endif /* __AKI3694_H__ */
```

ソースコード 11　framework/dependency/AKI3694/dependency.c

```
1 /*
2  * CPU レジスタの操作など、チップの固有性に依存するルーチン
3  *
```

```
 * @author fenrir (M.Naruoka)
 * @since 04/06/04
 * @version 1.0
 */

inline void enableIRQ(){
  asm volatile ("andc.b #0x7f,ccr");
}

inline void disableIRQ(){
  asm volatile ("orc.b #0x80,ccr");
}

inline void sleep(){
  asm volatile ("sleep");
}

inline void nop(){
  asm volatile("nop");
}
```

ソースコード 12　framework/dependency/AKI3694/common.h

```
#ifndef __COMMON_H__
#define __COMMON_H__

/* common.h がユーザ側で定義されていない場合のひな方 */

//#define AD_ONLY_4
//#define DISABLE_SCI3

//#define SCI3_TX_BUFFER_SIZE 32
//#define SCI3_RX_BUFFER_SIZE 16

//#define WDT_TMWD_SETTING 0x0F

/* タイマA の設定
 *
 * インターバルタイマとして動作するように
 * Φ/32
 * 20MHz / (32 * 256) = 2.5KHz = 0.4ms で割り込みがかかるようにする
 */
#define TIMER_A_SETTING 0x06

#endif
```

ソースコード 13　framework/dependency/AKI3694/ad.h

```
#ifndef __AD_H__
#define __AD_H__

/*
 * A/D 変換を行うルーチン
 *
 * @author fenrir (M.Naruoka)
 * @since 04/06/03
 * @version 1.0
 * @see ad.c
 */

#include <util/fifo_num.h>

#ifndef AD_ONLY_4
  #define AD_CH 8
```

```
#else
  #define AD_CH 4
#endif

#ifndef AD_BUFFER_SIZE
  #define AD_BUFFER_SIZE 8 /* バッファのサイズ */
#endif

#ifdef __cplusplus
extern "C"
{
#endif

void ad_init();
void ad_start();
void ad_end();

extern fifo_num_t fifo_ad[AD_CH]; /* A/D 変換後の結果用の FIFO、全部で 8 チャンネル */

#ifdef __cplusplus
}
#endif

#endif /* __AD_H__ */
```

ソースコード 14　framework/dependency/AKI3694/ad.c

```
/*
 * A/D 変換を行うルーチン
 *
 * @author fenrir (M.Naruoka)
 * @since 04/06/02
 * @version 1.0
 */
#include "AKI3694.h"
#include <common.h>
#include <util/fifo_num.h>
#include "ad.h"

static num_t buffer_ad[(AD_BUFFER_SIZE + 1) * AD_CH];
fifo_num_t fifo_ad[AD_CH];

/**
 * A/D 変換の初期設定をする関数
 *
 */
void ad_init(){

  /* A/D 変換後用の FIFO の初期化 */
  int i;
  for(i = 0; i < AD_CH; i++){
    fifo_num_init(&(fifo_ad[i]), &(buffer_ad[(AD_BUFFER_SIZE + 1) * i]), (AD_BUFFER_SIZE + 1));
  }

  /* A/D 変換停止 */
  AD.ADCSR.BIT.ADST = 0;

  /* スキャンモード
   * 高速変換
   * 割り込み可
   * はじめはグループ 0のAD0～AD3 を変換
   * 8チャンネル時はさらにその後グループ 1のAD4～AD7 を変換
   */
  AD.ADCSR.BYTE = 0x5b; //5
```

```
38 }
39
40 /**
41  * A/D 変換の開始をする関数
42  * インターバルタイマの割り込みによって、
43  * またグループ 0のAD 変換終了後に呼び出してもらう。
44  *
45  * @see ad_end();
46  */
47 void ad_start(){
48    AD.ADCSR.BIT.ADST = 1; /* A/D 変換スタート */
49 }
50
51 /**
52  * A/D 変換の終了後に起動される関数
53  * 割り込みで呼び出してもらう
54  *
55  * @see int_ad();
56  */
57 void ad_end(){
58
59    AD.ADCSR.BYTE &= 0x5f; /* 終了フラグをクリア & AD 変換を完全停止*/
60
61 #if AD_CH > 4
62    /* どっちのグループを変換していたか */
63    if((AD.ADCSR.BYTE & 0x04) == 0){
64 #endif
65
66       /* グループ 0(AD0～AD3)を変換していた */
67
68 #if DEBUG > 2
69       fifo_num_write(&fifo_ad[0], 230);
70       fifo_num_write(&fifo_ad[1], 340);
71       fifo_num_write(&fifo_ad[2], 560);
72       fifo_num_write(&fifo_ad[3], 780);
73 #else
74       /*fifo_num_write(&fifo_ad[0], ((AD.ADDRA_H << 8) | AD.ADDRA_L)); 旧方式 */
75       fifo_num_write(&fifo_ad[0], AD.ADDRA.u16);
76       fifo_num_write(&fifo_ad[1], AD.ADDRB.u16);
77       fifo_num_write(&fifo_ad[2], AD.ADDRC.u16);
78       fifo_num_write(&fifo_ad[3], AD.ADDRD.u16);
79 #endif
80
81 #if AD_CH > 4
82       AD.ADCSR.BYTE |= 0x04; /* グループ 1に切り替えて */
83       ad_start(); /* 再度AD 変換 */
84    }else{
85
86       /* グループ 1(AD4～AD7)を変換していた */
87
88 #if DEBUG > 2
89       fifo_num_write(&fifo_ad[4], 230);
90       fifo_num_write(&fifo_ad[5], 340);
91       fifo_num_write(&fifo_ad[6], 560);
92       fifo_num_write(&fifo_ad[7], 780);
93 #else
94       fifo_num_write(&fifo_ad[4], AD.ADDRA.u16);
95       fifo_num_write(&fifo_ad[5], AD.ADDRB.u16);
96       fifo_num_write(&fifo_ad[6], AD.ADDRC.u16);
97       fifo_num_write(&fifo_ad[7], AD.ADDRD.u16);
98 #endif
99
100      AD.ADCSR.BYTE &= ~(0x04); /* グループ 0に切り替え */
101   }
```

```
#endif
}

/**
 * A/D 変換の終了によって呼びだされる割り込み関数
 *
 * @see ad_end
 */
#pragma interrupt
void int_ad(){
  if(AD.ADCSR.BIT.ADF){AD.ADCSR.BIT.ADF = 0;}
  ad_end();
}
```

ソースコード 15　framework/dependency/AKI3694/sci.h

```
#ifndef __SCI_H__
#define __SCI_H__

/*
 * SCI をコントロールするルーチン
 *
 * @author fenrir (M.Naruoka)
 * @since 04/06/04
 * @version 1.0
 */

#ifdef __cplusplus
extern "C"
{
#endif

#include <util/fifo_char.h>

/* SCI3 */
#ifndef SCI3_DISABLE

#ifndef SCI3_TX_BUFFER_SIZE
  #define SCI3_TX_BUFFER_SIZE 31
#endif
#ifndef SCI3_RX_BUFFER_SIZE
  #define SCI3_RX_BUFFER_SIZE 15
#endif

  void sci3_init();
  fifo_char_size_t sci3_write(char *data, fifo_char_size_t size);
  fifo_char_size_t sci3_read(char *c, fifo_char_size_t size);
  fifo_char_size_t sci3_tx_size();
  fifo_char_size_t sci3_rx_size();

#endif /* SCI3_DISABLE */

#ifdef __cplusplus
};
#endif

#endif /* __SCI_H__ */
```

ソースコード 16　framework/dependency/AKI3694/sci.c

```
/*
 * リングバッファ +割り込みを利用したシリアル通信
 *
 * @author fenrir (M.Naruoka)
```

```
 * @since 04/05/30
 * @version 1.0
 */
#include <stdio.h>
#include "AKI3694.h"
#include <common.h>
#include "sci.h"

/* 送信 */
static fifo_char_size_t sci_write(fifo_char_t *, char *, fifo_char_size_t);
static void sci_send(volatile struct st_sci3 *, fifo_char_t *);

/* 受信 */
static fifo_char_size_t sci_read(fifo_char_t *, char *, fifo_char_size_t);
static void sci_recieve(volatile struct st_sci3 *, fifo_char_t *);

/* バッファサイズ */
static fifo_char_size_t sci_fifo_size(fifo_char_t *);

/* エラー処理 */
static void sci_error_handler(volatile struct st_sci3 *);


/**********/
/* SCI3 */
/**********/
#ifndef SCI3_DISABLE

static fifo_char_t fifo_tx3;
static fifo_char_t fifo_rx3;
static char buffer_tx3[SCI3_TX_BUFFER_SIZE + 1];
static char buffer_rx3[SCI3_RX_BUFFER_SIZE + 1];

/**
 * SCI3 を初期化します。
 *
 */
void sci3_init(){
  fifo_char_init(&fifo_tx3, buffer_tx3, SCI3_TX_BUFFER_SIZE + 1);
  fifo_char_init(&fifo_rx3, buffer_rx3, SCI3_RX_BUFFER_SIZE + 1);

  /* SCI3 のレジスタ設定はここに書くこと */

  /* 通信 全二重通信
   * 通信モード 調歩同期式モード
   * クロック 内部クロック
   * データ長 8bit
   * パリティ 無
   * ストップビット長 1bit
   * ビットレート 9600bps(BRR=64@20MHz)
   * 割り込み　TXI(送信データエンプティ)、RXI(受信データフル)、ERI(受信エラー)
   */
  SCI3.SCR3.BYTE = 0x00;
  SCI3.SMR.BYTE = 0x00;
  SCI3.BRR = 64;
  int i; for(i = 0; i < 10000; i++); // 1bit 期間待機
  SCI3.SSR.BYTE &= 0x80; // エラーフラグクリア
  SCI3.SCR3.BYTE |= 0xf0; // 割り込み許可
  IO.PMR1.BIT.TXD = 1;
}

/**
 * SCI3 から送信するデータを登録します。
 *
```

```
 * @param data 送信データ
 * @param size 送信データサイズ
 * @return (int) 送信バッファに登録されたデータサイズ
 */
fifo_char_size_t sci3_write(char *data, fifo_char_size_t size){
  fifo_char_size_t _size = sci_write(&fifo_tx3, data, size);
  /*
   * バッファが空の状態のときTXI は一時的に無効になっているので、
   * それを再度有効にする
   */
  if(_size > 0 && (SCI3.SCR3.BIT.TIE == 0)){SCI3.SCR3.BIT.TIE = 1;}
  return _size;
}

/**
 * SCI3 の送信バッファにたまっているデータ量を返します。
 *
 * @return (int) SCI3 の送信バッファにたまっているデータ量
 */
fifo_char_size_t sci3_tx_size(){return sci_fifo_size(&fifo_tx3);}

/**
 * SCI3 から受信したデータをもらいます。
 *
 * @param c 受信したデータを格納するバッファ
 * @param size 受信したデータを格納するバッファのサイズ
 * @return (int) 実際にバッファから読み込んだサイズ
 */
fifo_char_size_t sci3_read(char *c, fifo_char_size_t size){return sci_read(&fifo_rx3, c, size);}

/**
 * SCI3 の受信バッファにたまっているデータ量を返します。
 *
 * @return (int) SCI3 の受信バッファにたまっているデータ量
 */
fifo_char_size_t sci3_rx_size(){return sci_fifo_size(&fifo_rx3);}

/* 割り込み処理 */
#pragma interrupt
void int_sci3(){
  /* エラー処理 */
  if(SCI3.SSR.BYTE & 0x38){
    /* フラグクリア */
    SCI3.SSR.BYTE &= ~(0x38);
    sci_error_handler(&SCI3);
  }
  /* 送信データエンプティ */
  if(SCI3.SSR.BIT.TDRE){
    sci_send(&SCI3, &fifo_tx3);
  }
  /* 受信データフル */
  if(SCI3.SSR.BIT.RDRF){
    sci_recieve(&SCI3, &fifo_rx3);
  }
}

#endif /* SCI3_DISABLE */

/************/
/* 汎用 */
/************/

/* 送信 */
static fifo_char_size_t sci_write(fifo_char_t *fifo, char *data, fifo_char_size_t size){
```

```
133   return fifo_char_write(fifo, data, size); /* リングバッファに書き込み */
134 }
135
136 static void sci_send(volatile struct st_sci3 *sci, fifo_char_t *fifo){
137
138   /* 書き込むデータがあるか確認 */
139   if(fifo_char_read(fifo, (char *)&(sci->TDR), 1) == 0){
140
141     /* 書き込むデータがないので、一時的にTXI 割り込みがかからないようにする */
142     sci->SCR3.BIT.TIE = 0;
143   }
144 }
145
146 /* 受信 */
147 static fifo_char_size_t sci_read(fifo_char_t *fifo, char *c, fifo_char_size_t size){
148   return fifo_char_read(fifo, c, size); /* リングバッファから読み込み */
149 }
150
151 static void sci_recieve(volatile struct st_sci3 *sci, fifo_char_t *fifo){
152   fifo_char_write(fifo, (char *)&(sci->RDR), 1); /* リングバッファに 1バイト書き出し */
153 }
154
155 /* バッファサイズ */
156 static fifo_char_size_t sci_fifo_size(fifo_char_t *fifo){
157   return fifo_char_size(fifo);
158 }
159
160 /* エラー処理 */
161 static void sci_error_handler(volatile struct st_sci3 *sci){
162
163 }
```

ソースコード 17　framework/dependency/AKI3694/timer.h

```
1  #ifndef __ITU_H__
2  #define __ITU_H__
3
4  /*
5   * タイマをコントロールするルーチン
6   *
7   * @author fenrir (M.Naruoka)
8   * @since 04/06/04
9   * @version 1.0
10  */
11
12 #ifndef TIMER_A_SETTING
13 #define TIMER_A_SETTING 0x06
14 #endif
15
16 #ifdef __cplusplus
17 extern "C"
18 {
19 #endif
20
21 void timer_init(void (*)());
22
23 #ifdef __cplusplus
24 }
25 #endif
26
27 #endif /* __ITU_H__ */
```

ソースコード 18　framework/dependency/AKI3694/timer.c

```
/*
 * タイマをコントロールするルーチン
 *
 * @author fenrir (M.Naruoka)
 * @since 04/06/04
 * @version 1.0
 */
#include "AKI3694.h"
#include <common.h>
#include "timer.h"

//インターバルタイマによって呼び出される関数
static void (*timer_a_callback)();

/**
 * タイマA を初期化するルーチン
 *
 * @param f タイマA(インターバル)割り込みによってコールバックされる関数
 */
void timer_init(void (*f)()){

  /* タイマA の設定
   *
   */
  TA.TMA.BYTE = TIMER_A_SETTING;

  /* タイマA コールバック関数の登録 */
  timer_a_callback = f;

  /* タイマA 割り込み許可 */
  IENR1.BIT.IENTA = 1;
}

/**
 * タイマ割り込みによって呼び出されるルーチン
 *
 */
#pragma interrupt
void int_timera(){
  IRR1.BIT.IRRTA = 0;
  timer_a_callback();
}
```

ソースコード 19　framework/dependency/AKI3694/wdt.h

```
#ifndef __WDT_H__
#define __WDT_H__

/*
 * ウォッチドックタイマをコントロールするルーチン
 *
 * @author fenrir (M.Naruoka)
 * @since 04/06/04
 * @version 1.0
 */

#ifndef WDT_TMWD_SETTING
#define WDT_TMWD_SETTING 0x0F
#endif

#ifdef __cplusplus
extern "C"
{
#endif
```

```
20
21 void wdt_init();
22 void wdt_start();
23 void wdt_stop();
24 void wdt_reset(unsigned char tcwd);
25 unsigned char wdt_reseted();
26
27 #ifdef __cplusplus
28 }
29 #endif
30
31 #endif /* __ITU_H__ */
```

ソースコード 20　framework/dependency/AKI3694/wdt.c

```
1  /*
2   * タイマをコントロールするルーチン
3   *
4   * @author fenrir (M.Naruoka)
5   * @since 04/06/04
6   * @version 1.0
7   */
8  #include "AKI3694.h"
9  #include <common.h>
10 #include "wdt.h"
11
12 /**
13  * ウォッチドックタイマを初期化するルーチン
14  */
15 void wdt_init(){
16   /* WDT の設定
17     *
18     */
19   WDT.TMWD.BYTE = WDT_TMWD_SETTING;
20 }
21
22
23 /**
24  * ウォッチドックタイマを開始するルーチン
25  */
26 void wdt_start(){
27   WDT.TCSRWD.BYTE = 0x9A;
28   WDT.TCSRWD.BYTE = 0xA6;
29 }
30
31 /**
32  * ウォッチドックタイマをリセットするルーチン
33  *
34  */
35 void wdt_reset(unsigned char tcwd){
36   WDT.TCSRWD.BYTE = 0x6A;
37   WDT.TCWD = tcwd;
38 }
39
40 /**
41  * ウォッチドックタイマをストップするルーチン
42  *
43  */
44 void wdt_stop(){
45   WDT.TCSRWD.BYTE = 0x9A;
46   WDT.TCSRWD.BYTE = 0xA2;
47 }
48
49 /**
```

```
 * ウォッチドックタイマによるリセットかを検出する
 */
unsigned char wdt_reseted(){
  return WDT.TCSRWD.BIT.WRST == 1;
}
```

ソースコード 21　framework/util/Makefile

```
CC = h8300-elf-gcc
NEWLIB_DIR = /usr/local/h8300-elf/lib/h8300h

CFLAGS ?= \
  -I../.. -I. -I/usr/local/h8300-elf/include \
  -Wall -mh -g -mint32 -mrelax -gdwarf-2

BUILD_DIR ?= build_by_gcc

SRCS_C = \
  $(shell ls *.c)

OBJS = $(SRCS_C:.c=.o)

all : $(BUILD_DIR) $(patsubst %,$(BUILD_DIR)/%,$(OBJS))

$(BUILD_DIR)/%.o : %.c
  $(CC) -c $(CFLAGS) -o $@ $<

$(BUILD_DIR) :
  mkdir $@

clean :
  cd $(BUILD_DIR); rm -f $(OBJS)

run : all

.PHONY : clean all
```

ソースコード 22　framework/util/common.h

```
#ifndef __COMMON_H__
#define __COMMON_H__

/* common.h がユーザ側で定義されていない場合のひな方 */

/**
 * 定数
 */
#define TRUE 1
#define FALSE 0
#define EXIT 0
#define CONTINUE 1
#define VOID -1
#define PI 3.14159
#define INIT_STR "Initial."
#ifndef NULL
  #define NULL 0
#endif

/* ユーティリティ */
#define min(a, b) (a <= b ? a : b)
#define max(a, b) (a >= b ? a : b)
#define pow2(x) (x * x)

#endif
```

ソースコード 23　framework/util/conv.h

```
#ifndef __CONV_H__
#define __CONV_H__

/* 文字列処理 */
char *uint2string(unsigned int i, char *buffer);
char *int2string(int i, char *buffer);
char *ushort2string(unsigned short i, char *buffer);
char *short2string(short i, char *buffer);
char *long2string(long l, char *buffer);
inline char *crlf(char *buffer);
inline char *null_char(char *buffer);

#endif /* __CONV_H__ */
```

ソースコード 24　framework/util/conv.c

```
#include <common.h>
#include "conv.h"

/* 文字列処理、printf の代用品 */
char *uint2string(unsigned int i, char *buffer){
  unsigned int digit;
  unsigned int power;
  unsigned char hit = FALSE;
  for(power = 10000000; power > 1; power /= 10){
    digit = i / power;
    if(digit > 0){hit = TRUE;}
    if(hit){*(buffer++) = digit + '0';}
    i -= digit * power;
  }
  *(buffer++) = i + '0';
  return buffer;
}

char *int2string(int i, char *buffer){
  if(i < 0){
    *(buffer++) = '-';
    i *= -1;
  }
  return uint2string((unsigned int)i, buffer);
}

char *ushort2string(unsigned short i, char *buffer){
  unsigned short digit;
  unsigned short power;
  unsigned char hit = FALSE;
  for(power = 10000; power > 1; power /= 10){
    digit = i / power;
    if(digit > 0){hit = TRUE;}
    if(hit){*(buffer++) = digit + '0';}
    i -= digit * power;
  }
  *(buffer++) = i + '0';
  return buffer;
}

char *short2string(short i, char *buffer){
  if(i < 0){
    *(buffer++) = '-';
    i *= -1;
  }
  return ushort2string((unsigned short)i, buffer);
}
```

```
48
49 char *long2string(long l, char *buffer){
50
51   if(l < 0){*(buffer++) = '-'; l *= -1;}
52
53   int digit;
54   long power;
55   unsigned char hit = FALSE;
56   for(power = 1000000000; power > 1; power /= 10){
57     digit = l / power;
58     if(digit > 0){hit = TRUE;}
59     if(hit){*(buffer++) = digit + '0';}
60     l -= digit * power;
61   }
62   *(buffer++) = l + '0';
63   return buffer;
64 }
65
66 inline char *crlf(char *buffer){
67   *(buffer++) = '\r';
68   *(buffer++) = '\n';
69   return buffer;
70 }
71
72 inline char *null_char(char *buffer){
73   *(buffer++) = '\0';
74   return buffer;
75 }
```

ソースコード 25　framework/util/fifo_char.h

```
1  /*
2   * Ring Buffer という FIFO を表現するヘッダファイル
3   * ring_buffer.c がメインソース
4   *
5   * @author fenrir (M.Naruoka)
6   * @since 04/05/30
7   * @version 1.0
8   */
9
10 #ifndef __FIFO_CHAR_H__
11 #define __FIFO_CHAR_H__
12
13 #ifdef __cplusplus
14 extern "C"
15 {
16 #endif
17
18 #ifndef FIFO_CHAR_SIZE_T
19   #define FIFO_CHAR_SIZE_T int
20 #endif
21
22 typedef FIFO_CHAR_SIZE_T fifo_char_size_t;
23
24 typedef struct rbuffer_char_t{
25   char *buffer;
26   fifo_char_size_t size;
27   char *prius;
28   char *follower;
29 } fifo_char_t;
30 /* リングバッファルール
31  * follower(追っかけ)はprius(先行者)に追いつくことはできないし、追い抜くこともできない
32  * prius(先行者)はfollower(追っかけ)に追いつくことはできるが、追い抜くことはできない
33  */
```

```
34
35 fifo_char_t *fifo_char_init(fifo_char_t *buffer, char *c, fifo_char_size_t size);
36 fifo_char_size_t fifo_char_write(fifo_char_t *buffer, char *data, fifo_char_size_t size);
37 fifo_char_size_t fifo_char_read(fifo_char_t *buffer, char *c, fifo_char_size_t size);
38 fifo_char_size_t fifo_char_size(fifo_char_t *buffer);
39
40 #ifdef __cplusplus
41 }
42 #endif
43
44 #endif /* __FIFO_CHAR_H__ */
```

ソースコード 26　framework/util/fifo_char.c

```
1  /*
2   * リングバッファ (FIFO)を表現するC ソース
3   * これは文字列 (char)用。
4   *
5   * @author fenrir (M.Naruoka)
6   * @since 04/05/30
7   * @version 1.0
8   */
9
10 #include <stdio.h>
11 #include <common.h>
12 #include "fifo_char.h"
13
14 /**
15  * リングバッファを初期化します。
16  *
17  * @param buffer 初期化するリングバッファ
18  * @param c 実際にバッファとして機能する領域
19  * @param size 実際のバッファサイズ、c の大きさ
20  * @return (fifo_char_t) 初期化されたリングバッファ
21  */
22 fifo_char_t *fifo_char_init(fifo_char_t *buffer, char *c, fifo_char_size_t size){
23
24   if(size > 0){
25
26     buffer->buffer = c;
27     buffer->size = size;
28     buffer->prius = buffer->buffer;
29     buffer->follower = buffer->buffer + size - 1;
30     return buffer;
31   }else{return NULL;}
32 }
33
34 /**
35  * リングバッファにデータを書き込みます。
36  *
37  * @param buffer 書き込む対象リングバッファ
38  * @param data 書き込むデータ
39  * @param size 書き込むデータのサイズ (byte)
40  * @return int 正常に書き込まれたデータのサイズ (byte)
41  */
42 fifo_char_size_t fifo_char_write(fifo_char_t *buffer, char *data, fifo_char_size_t size){
43
44   fifo_char_size_t _size = 0;
45
46   if(data != NULL){
47
48     /* バッファに書き込み */
49     while(_size < size){
50
```

```
51          if(buffer->prius == buffer->buffer + buffer->size){buffer->prius = buffer->buffer;}
52          if(buffer->prius == buffer->follower){break;}
53          *((buffer->prius)++) = *(data + (_size++));
54        }
55      }
56
57      return _size;
58  }
59
60  /**
61   * リングバッファからデータを読み取ります。
62   *
63   * @param buffer 読み込み元となるリングバッファ
64   * @param c 読み取ったデータを格納するバッファ
65   * @param size 最大読み込みサイズ (byte)
66   * @return int 読み取ったデータの大きさ (byte)
67   */
68  fifo_char_size_t fifo_char_read(fifo_char_t *buffer, char *c, fifo_char_size_t size){
69
70      fifo_char_size_t _size = 0;
71
72      if(c != NULL){
73
74          /* 読込先に書き込み */
75          while(_size < size){
76
77              if((buffer->follower + 1 - buffer->prius) % buffer->size == 0){break;}
78              if(++buffer->follower == buffer->buffer + buffer->size){buffer->follower = buffer->buffer;}
79              *(c + (_size++)) = *(buffer->follower);
80          }
81      }
82
83      return _size;
84  }
85
86  /**
87   * リングバッファ上にあるデータサイズを求めます。
88   *
89   * @param buffer リングバッファ
90   * @return int バッファ上のデータの大きさ (byte)
91   */
92  fifo_char_size_t fifo_char_size(fifo_char_t *buffer){
93
94      return (buffer->prius > buffer->follower ?
95                  buffer->prius - buffer->follower - 1:
96                  buffer->prius + buffer->size - buffer->follower - 1);
97  }
```

ソースコード 27　framework/util/fifo_num.h

```
1   /*
2    * 数値用Ring Buffer(FIFO)のヘッダファイル
3    * fifo_num.c がメインソース
4    *
5    * @author fenrir (M.Naruoka)
6    * @since 04/05/30
7    * @version 1.0
8    * @see fifo_num.c
9    * @see common.h
10   */
11
12  #ifndef __FIFO_NUM_H__
13  #define __FIFO_NUM_H__
14
```

```
15 #ifdef __cplusplus
16 extern "C"
17 {
18 #endif
19
20 #ifndef FIFO_NUM_SIZE_T
21 #define FIFO_NUM_SIZE_T int
22 #endif
23
24 typedef FIFO_NUM_SIZE_T fifo_num_size_t;
25
26 #ifndef FIFO_NUM_T
27 #define FIFO_NUM_T int
28 #endif
29
30 typedef FIFO_NUM_T num_t;
31
32
33 typedef struct{
34   num_t *buffer;
35   fifo_num_size_t size;
36   num_t *prius;
37   num_t *follower;
38 } fifo_num_t;
39 /* リングバッファルール
40  * follower(追っかけ)はprius(先行者)に追いつくことはできないし、追い抜くこともできない
41  * prius(先行者)はfollower(追っかけ)に追いつくことはできるが、追い抜くことはできない
42  */
43
44 fifo_num_t *fifo_num_init(fifo_num_t *buffer, num_t *array, fifo_num_size_t size);
45 num_t *fifo_num_write(fifo_num_t *buffer, num_t data);
46 num_t *fifo_num_read(fifo_num_t *buffer, num_t *data);
47 fifo_num_size_t fifo_num_size(fifo_num_t *buffer);
48
49 #ifdef __cplusplus
50 };
51 #endif
52
53 #endif /* __FIFO_NUM_H__ */
```

ソースコード 28　framework/util/fifo_num.c

```
1  /*
2   * リングバッファ (FIFO)を表現するC ソース
3   * これは数値用。
4   * num_t を定義して使用のこと
5   *
6   * @author fenrir (M.Naruoka)
7   * @since 04/05/30
8   * @version 1.0
9   */
10
11 #include <stdio.h>
12 #include <common.h>
13 #include "fifo_num.h"
14
15 /**
16  * FIFO を初期化します。
17  *
18  * @param buffer 初期化するFIFO
19  * @param array 実際にバッファとして機能する領域
20  * @param size 実際のバッファサイズ、array の大きさ
21  * @return (fifo_num_t) 初期化されたリングバッファ
22  */
```

```
23 fifo_num_t *fifo_num_init(fifo_num_t *buffer, num_t *array, fifo_num_size_t size){
24
25   if(size > 0){
26
27     buffer->buffer = array;
28     buffer->size = size;
29     buffer->prius = buffer->buffer;
30     buffer->follower = buffer->buffer + size - 1;
31     return buffer;
32   }else{return NULL;}
33 }
34
35 /**
36  * FIFO に数値データを書き込みます。
37  *
38  * @param buffer 書き込む対象FIFO
39  * @param data 書き込む数値データ
40  * @return (accuracy *) 正常に書き込まれたかどうか、NULL は異常
41  */
42 num_t *fifo_num_write(fifo_num_t *buffer, num_t data){
43
44   /* 空いているかチェック */
45   if(buffer->prius != buffer->follower){
46     if(buffer->prius == buffer->buffer + buffer->size){buffer->prius = buffer->buffer;}
47     *(buffer->prius++) = data;
48     return buffer->prius;
49   }
50
51   return NULL;
52 }
53
54 /**
55  * FIFO から数値データを読み取ります。
56  *
57  * @param buffer 読み込み元となるFIFO
58  * @param data 読み取ったデータを格納するバッファ
59  * @return (accuracy *) 読み取ったデータを格納するバッファ、データを読み込まなかった場合、NULL
60  */
61 num_t *fifo_num_read(fifo_num_t *buffer, num_t *data){
62
63   if(data != NULL){
64
65     /* リングバッファ上にデータがあるかチェック */
66     if(((buffer->follower + 1) - buffer->prius) % buffer->size != 0){
67
68       /* 読込先に書き込み */
69       if(++buffer->follower == buffer->buffer + buffer->size){buffer->follower = buffer->buffer;}
70       *data = *(buffer->follower);
71       return data;
72     }
73   }
74
75   return NULL;
76 }
77
78 /**
79  * FIFO 上にあるデータの個数を取得します。
80  *
81  * @param buffer FIFO
82  * @return (int) FIFO 上にあるデータの個数
83  */
84 fifo_num_size_t fifo_num_size(fifo_num_t *buffer){
85
86   return (buffer->prius > buffer->follower ?
```

```
87              buffer->prius - buffer->follower - 1:
88              buffer->prius + buffer->size - buffer->follower - 1);
89  }
```

ハードウェア

機体搭載機器として以下のハードウェアを作成した。回路図を図 4 に示す。また、この回路図にそって基板を作成した。基板のレイアウトを図 5 に記載する。使用した部品を表 1 にあげる。

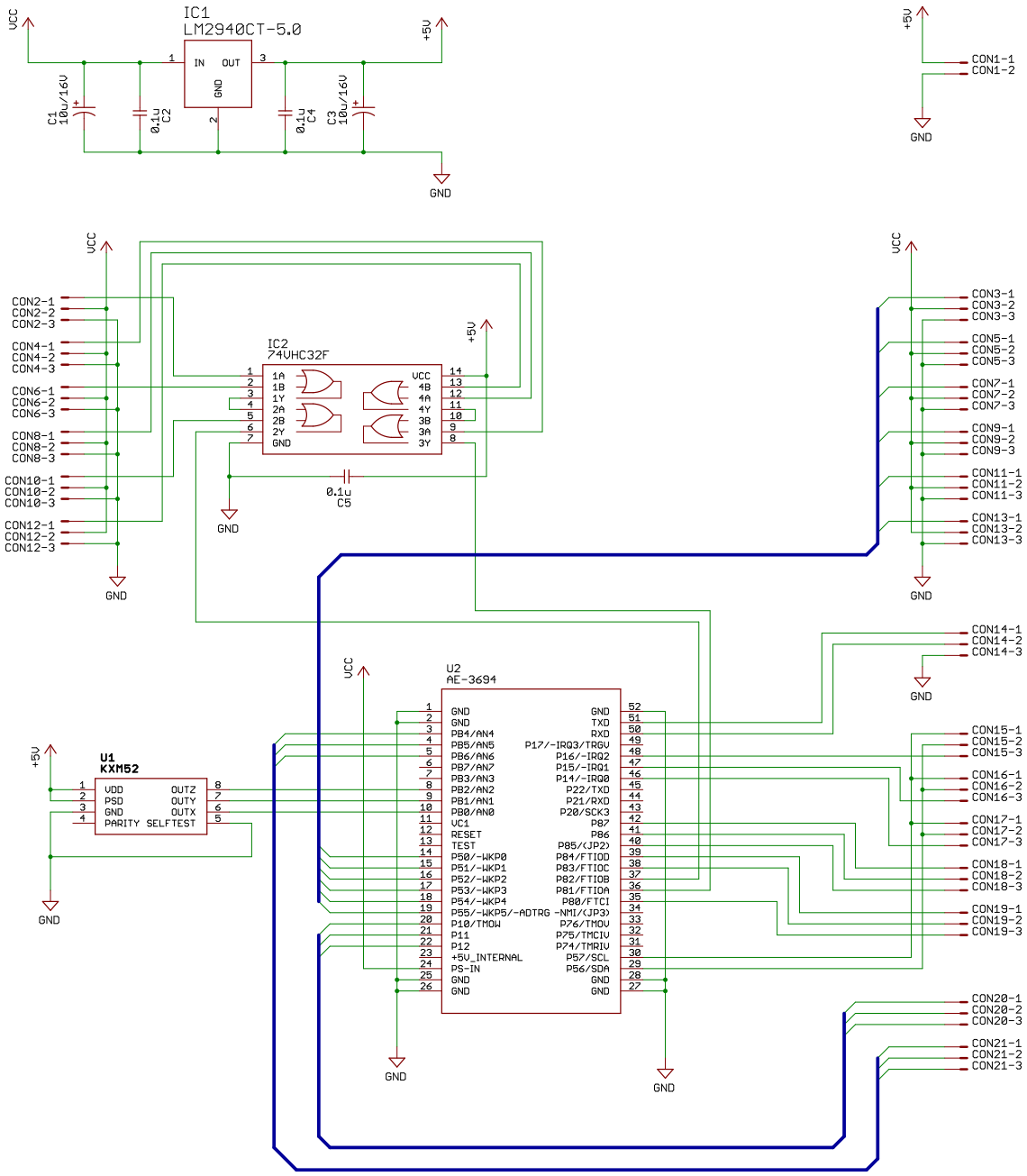

図 4　機体搭載機器 回路図

補足を行うと、スピコンから供給される電圧が 5V であったため、作成した基板のレギュレータを使用する必要はなかった。レギュレータを使用しない設定の場合、秋月 H8/3694F のジャンパ線を一箇所切断する必要がある (秋月のマ

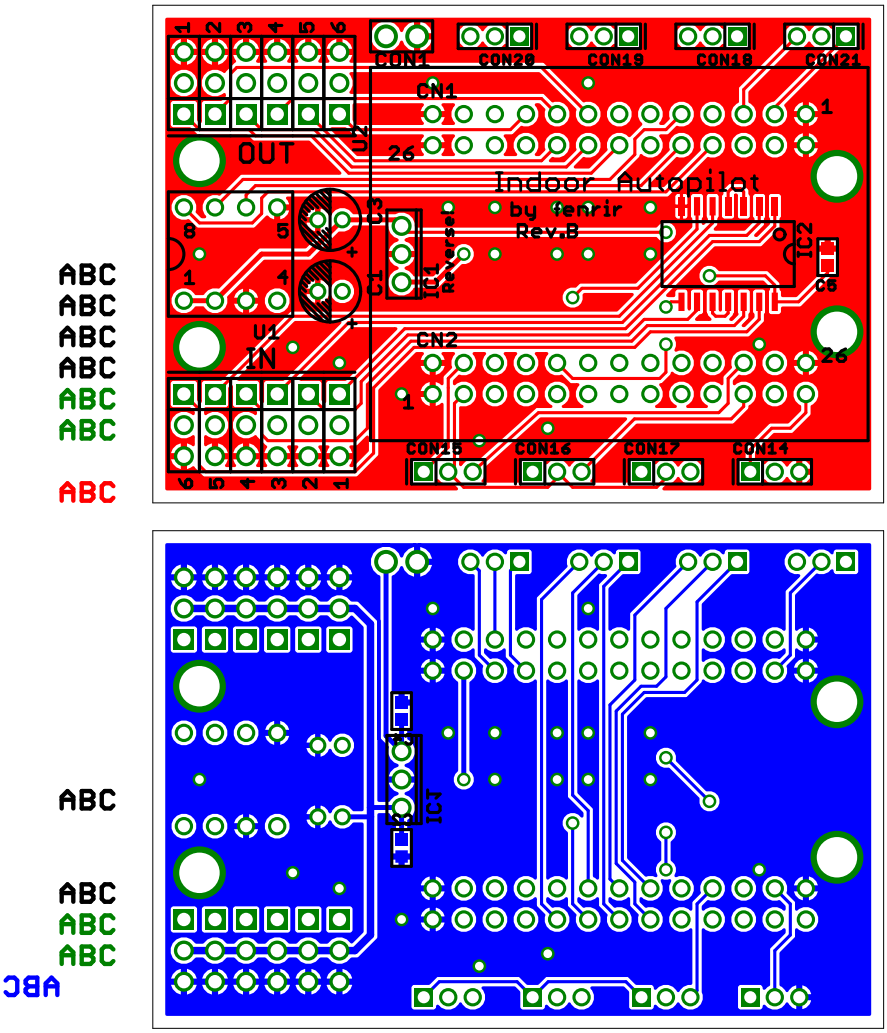

図 5　機体搭載基板 レイアウト (表・裏)

ニュアルを参照) ので注意すること。これを怠ると秋月基板上のレギュレータに逆電圧がかかり損傷する恐れがある。

最後に、ハードウェアの参考リソースとして以下のものをあげておく。

- 秋月電子通商 キット取扱い説明書回路図集
- 同 H8/3664FP フラットマイコンモジュール・キット
  今回使用したモジュールの取説。
- ルネサス H8/3694 グループ
  マイコンのマニュアル等がダウンロードできる。

表 1　機体搭載基板 部品表

名前	値	説明
C1	10μ/16V	OS コン、千石地下 1 階
C2	0.1μ	1608 パッケージ
C3	10μ/16V	OS コン、千石地下 1 階
C4	0.1μ	1608 パッケージ
C5	0.1μ	1608 パッケージ
CON1		フルピッチ (2.54 mm ピッチ) 3 穴
CON2		フルピッチ 3 穴
CON3		フルピッチ 3 穴
CON4		フルピッチ 3 穴
CON5		フルピッチ 3 穴
CON6		フルピッチ 3 穴
CON7		フルピッチ 3 穴
CON8		フルピッチ 3 穴
CON9		フルピッチ 3 穴
CON10		フルピッチ 3 穴
CON11		フルピッチ 3 穴
CON12		フルピッチ 3 穴
CON13		フルピッチ 3 穴
CON14		2.0 mm ピッチ 3 穴
CON15		2.0 mm ピッチ 3 穴
CON16		2.0 mm ピッチ 3 穴
CON17		2.0 mm ピッチ 3 穴
CON18		2.0 mm ピッチ 3 穴
CON19		2.0 mm ピッチ 3 穴
CON20		2.0 mm ピッチ 3 穴
CON21		2.0 mm ピッチ 3 穴
IC1	LM2940CT-5.0	3 端子レギュレータ、秋月
IC2	74VHC32F	SOP14 パッケージ、千石
U1	KXM52	DIP8 パッケージ、秋月 3 軸加速度計モジュール
U2	AE-3694	秋月 H8/3664FP フラットマイコンモジュール・キット