```
/*-------------------------------------------------*/
/* ものづくりコンテスト山形県大会08 ( 指定関数 )      */
/*-------------------------------------------------*/
/*-------------------------------------------------------------------------------*/
/* 指定関数(1)                                                                   */
/*  モジュール名 get_ad                                                          */
/*  処理概要     アナログ入力の関数                                              */
/*               距離センサーのアナログ出力値をＡＤ変換し、変換後のデータを返す関数。*/
/*               ただし、ＡＤ変換のデータは、本来分解能１０ビット（０～１０２３）  */
/*               であるが、そのうちの上位8ビットを有効なデータとする。また、      */
/*               データのばらつきをなくすために、複数回（5回程度）読み込んだデータの */
/*               平均値を返すものとする。                                        */
/*                                                                               */
/*               上位10bitにデータが格納されるので、下位8bitにシフトし(右に8bit)   */
/*               変換データを確定する。（下位2bitゴミとしてカット）              */
/*  引数         なし                                                            */
/*  戻り値       8bit data                                                       */
/*               1111 1111 11-- ---- : 変換後データ                              */
/*               ---- ---- 1111 1111 : 8bitシフト後データ                        */
/*                                                                               */
/*               ---- ---- 0000 0000 : 0x00      0                              */
/*                    :                 :                                      */
/*               ---- ---- 1111 1111 : 0xff     255  0～255 の 256通りのデータ   */
/*-------------------------------------------------------------------------------*/
unsigned int get_ad( void )
{
    unsigned char sensor;       /* unsigned intで 0 ～ 255 まで扱える */

    unsigned char indata1, indata2, indata3, indata4, indata5;


       AD.CSR.BYTE = 0x00;            /*(Init_AD) adch0(AN0 (P70) ) 単一モード */


      /* 1回目 */
       AD.CSR.BIT.ADST = 1;           /*(Start_AD)                AD変換開始 */
       while( AD.CSR.BIT.ADF != 1 ){} /*(End_AD  ) AD変換終了チェック */
       indata1 = AD.DRA >>8;          /* 上位10bitを右に8bitシフトする。下位2bit無視 */


      /* 2回目 */
       AD.CSR.BIT.ADST = 1;           /*(Start_AD)                AD変換開始 */
       while( AD.CSR.BIT.ADF != 1 ){} /*(End_AD  ) AD変換終了チェック */
       indata2 = AD.DRA >>8;          /* 上位10bitを右に8bitシフトする。下位2bit無視 */


      /* 3回目 */
       AD.CSR.BIT.ADST = 1;           /*(Start_AD)                AD変換開始 */
       while( AD.CSR.BIT.ADF != 1 ){} /*(End_AD  ) AD変換終了チェック */
       indata3 = AD.DRA >>8;          /* 上位10bitを右に8bitシフトする。下位2bit無視 */


      /* 4回目 */
       AD.CSR.BIT.ADST = 1;           /*(Start_AD)                AD変換開始 */
       while( AD.CSR.BIT.ADF != 1 ){} /*(End_AD  ) AD変換終了チェック */
       indata4 = AD.DRA >>8;          /* 上位10bitを右に8bitシフトする。下位2bit無視 */


      /* 5回目 */
       AD.CSR.BIT.ADST = 1;           /*(Start_AD)                AD変換開始 */
       while( AD.CSR.BIT.ADF != 1 ){} /*(End_AD  ) AD変換終了チェック */
       indata5 = AD.DRA >>8;          /* 上位10bitを右に8bitシフトする。下位2bit無視 */


      /* 5回の平均値 */
       sensor = ( indata1 + indata2 + indata3 + indata4 + indata5 ) / 5;


      /* 値を返す */
       return sensor;
}


/*-------------------------------------------------------------------------------*/
/* 指定関数(2)                                                                   */
/*  モジュール名 disp_led                                                        */
/*  処理概要     ＬＥＤへデータを表示させるための関数                            */
/*               ＬＥＤ０～ＬＥＤ７に与えられたデータ（８ビット）を表示するための関数。*/
/*               表示データは、上位ビットをＬＥＤ７に、下位ビットをＬＥＤ０に表示し、 */
/*               '１'で点灯、'０'で消灯とする。                                  */
/*  引数         表示データ                                                      */
/*  戻り値       なし                                                            */
/*-------------------------------------------------------------------------------*/
void disp_led( unsigned int a)
{
      /* 表示はLED */
```

```
 87          SEG_SELECT = LED;
 88
 89        /* データ出力 */
 90           KIBAN081 = ~a;
 91 }
 92
 93
 94 /*-------------------------------------------------------------------------------*/
 95 /* 指定関数(3)                                                                   */
 96 /*  モジュール名 disp_7seg                                                       */
 97 /*  処理概要     ７セグメントＬＥＤへデータを表示させるための関数               */
 98 /*               ７セグメントＬＥＤに０～９の数字を表示させるための関数。       */
 99 /*               数字の形状は任意とする。                                        */
100 /*  引数         表示データ                                                      */
101 /*  戻り値       なし                                                            */
102 /*-------------------------------------------------------------------------------*/
103 void disp_7seg( unsigned int a)
104 {
105       /* 表示は7セグメントLED */
106          SEG_SELECT = T7SEG;
107
108        /* データ出力 */
109           KIBAN081 = tendo[ a ];
110 }
111
112
113 /*-----------------------------------------------------------------------------------------*/
114 /* 指定関数(4)                                                                             */
115 /*  モジュール名 wait                                                                      */
116 /*  処理概要     ソフトウェアタイマー                                                      */
117 /*               およそ１[msec]のソフトウェアによるタイマーの関数。引数を繰返し回数とする。*/
118 /*               今回の山形県大会では厳密な時間を必要としないため、                        */
119 /*               割込みによるタイマーはなくても構いません。                                */
120 /*  引数         タイマー値    1:  1[ms]                                                   */
121 /*                            10: 10[ms] = 0.1[s]                                         */
122 /*                            50: 50[ms] = 0.5[s]                                         */
123 /*                           100:100[ms] = 1.0[s]                                         */
124 /*  戻り値       なし                                                                      */
125 /*-----------------------------------------------------------------------------------------*/
126 void wait( int iTimer )
127 {
128     int i;
129          while( iTimer ) {
130               for( i = 0; i < 5000; i++ );    /* H8/3048f */
131                    iTimer--;
132          }
133 }
134
135
136 /*-------------------------------------------------------------------------------*/
137 /* 指定関数(5)                                                                   */
138 /*  モジュール名 handan                                                          */
139 /*  処理概要     物体の位置を判別するための関数                                  */
140 /*               第８回ものづくりコンテスト全国大会電子回路組立部門課題（案）における、*/
141 /*               課題２の「測定位置」を「範囲（目安）」に従って判定し、          */
142 /*               その「測定位置」を返す関数。                                    */
143 /*               戻り値は、２桁（１０，２０，・・）でも、１桁（１，２，・・）の  */
144 /*               どちらでも可とする。                                            */
145 /*  引数         psdデータ値                                                     */
146 /*  戻り値       なし                                                            */
147 /*-------------------------------------------------------------------------------*/
148 unsigned char handan( unsigned int psd_data )
149 {
150
151     unsigned char distance;
152     unsigned char position;
153
154
155       /* センサデータを判別する */
156          if( psd_data >= 176 ) {
157               position = 1;  /* 物体の位置は 10cm ～ 15cm */
158               led_position = 0x01;
159
160          } else if( psd_data >= 118 ) {
161               position = 2;  /* 物体の位置は 15cm ～ 25cm */
162               led_position = 0x02;
163
164          } else if( psd_data >=  85 ) {
165               position = 3;  /* 物体の位置は 25cm ～ 35cm */
166               led_position = 0x04;
167
168          } else if( psd_data >=  76 ) {
169               position = 4;  /* 物体の位置は 35cm ～ 45cm */
170               led_position = 0x08;
171
172          } else if( psd_data >=  63 ) {
```

```
173              position = 5;  /* 物体の位置は 45cm ～ 55cm */
174              led_position = 0x10;
175
176          } else if( psd_data >=  55 ) {
177              position = 6;  /* 物体の位置は 55cm ～ 65cm */
178              led_position = 0x20;
179
180          } else if( psd_data >=  49 ) {
181              position = 7;  /* 物体の位置は 65cm ～ 75cm */
182              led_position = 0x40;
183
184          } else if( psd_data >=  40 ) {
185              position = 8;  /* 物体の位置は 75cm ～ 80cm */
186              led_position = 0x80;
187
188          } else {
189              position = 0;  /* 物体の位置は 80cm ～      */
190              led_position = 0x00;
191          }
192
193       /* 値を返す */
194          return position;
195 }
196
197
198
199 /*-------------------------------------------------------*/
200 /* ものづくりコンテスト山形県大会08 山形電波工高csc関数 */
201 /*-------------------------------------------------------*/
202 /*-------------------------------------------------------------------------------*/
203 /* その他関数(1)                                                                 */
204 /*  モジュール名  openning                                                       */
205 /*  処理概要      オープニング(山形電波工高固有)                                 */
206 /*  引数          なし                                                           */
207 /*  戻り値        なし                                                           */
208 /*-------------------------------------------------------------------------------*/
209 void openning08( void )
210 {
211     signed char i;                  /* -128 ～ +127 */
212
213          SEG_SELECT = T7SEG;         /* 表示は7セグメントLED */
214          for( i = 3; i >= 0; i-- ) {
215              KIBAN081= tendo[ i ];   /* 表示 */
216                 wait( 50 );
217              KIBAN081= 0xff;         /* KIBAN081 = ~0x00; */
218            /*seg_cls();*/          /* 消去 */
219                 wait( 50 );
220          }
221 }
222
223
224 /*-------------------------------------------------------------------------------*/
225 /* その他関数(2)                                                                 */
226 /*  モジュール名 Init_Port                                                       */
227 /*  処理概要     H8 ポート初期化                                                 */
228 /*  引数         なし                                                           */
229 /*  戻り値       なし                                                           */
230 /*-------------------------------------------------------------------------------*/
231 void Init_Port( void )
232 {
233       /* I/Oポート設定  0:入力  1:出力 */
234          P6.DDR = 0x00;           /* on bord ディップスイッチ        */
235                                   /* 0x00 = 0000 0000  全bit入力設定 */
236
237          PA.DDR = 0xff;           /* 事前配布基板(1) LED & 7segLED          */
238                                   /* 0xff = 1111 1111  全bit出力設定        */
239          PA.DR.BYTE = 0x00;
240
241          P5.DDR = 0xff;           /* 事前配布基板(1) LED or 7segLED指示端子 */
242                                   /* 0xff = 1111 1111  全bit出力設定        */
243          P5.DR.BYTE = 0x00;
244
245
246          PB.DDR = 0xff;           /*                                        */
247                                   /* 0xff = 1111 1111  全bit出力設定        */
248          PB.DR.BYTE = 0x00;
249
250                                   /* 当日作成基板はP7ポート(P7は入力専用なので設定なし) */
251 }
252
253
254
255 /*-------------------------------------------------------------------------------*/
256 /* その他関数(3)                                                                 */
257 /*  モジュール名 Init_H8                                                         */
258 /*  処理概要     H8 タイマー初期化                                               */
```

```
259 /*  引数        なし                                                          */
260 /*  戻り値      なし                                                          */
261 /*--------------------------------------------------------------------------*/
262 /* 参考 H8/3048f(旧cpu) 14.7854MHz      */
263 /*       16ms   29490                   */
264 /*       10ms   18432                   */
265 /*        8ms   14745                   */
266 /*        5ms    9216                   */
267 /*        1ms    1843                   */
268 /*---------------------------------*/
269 void Init_H8( void )
270 {
271     EI; /* 割り込み許可 */
272
273       /* タイマ設定      */
274          ITU.TMDR.BIT.PWM0 = 0;    /* ITU0:通常動作       */
275          ITU.TMDR.BIT.PWM1 = 0;    /* ITU1:通常動作       */
276          ITU.TMDR.BIT.PWM2 = 0;    /* ITU2:通常動作       */
277          ITU.TMDR.BIT.PWM3 = 0;    /* ITU3:通常動作       */
278          ITU.TMDR.BIT.PWM4 = 0;    /* ITU4:通常動作       */
279
280
281       /* タイマ設定 */
282          ITU4.TCR.BIT.CCLR = 1;    /* カウンタクリア要因                          */
283          ITU4.TCR.BIT.CKEG = 0;    /* クロックエッジ                              */
284          ITU4.TCR.BIT.TPSC = 3;    /* タイマプリスケーラ 1.8432MHz                */
285          ITU4.GRA = 18432;         /* 割り込みインターバル 1.8432MHz/18432=100Hz */
286          ITU4.TIER.BIT.IMIEA = 1;  /* IMFAフラグによる割り込み許可                */
287
288
289       /* タイマスタート */
290          ITU.TSTR.BIT.STR0 = 0;    /* まだスタートしない */
291          ITU.TSTR.BIT.STR1 = 0;    /* まだスタートしない */
292          ITU.TSTR.BIT.STR2 = 0;    /* まだスタートしない */
293          ITU.TSTR.BIT.STR3 = 0;    /* まだスタートしない */
294          ITU.TSTR.BIT.STR4 = 0;    /* まだスタートしない */
295 }
296
297
298
299 /*--------------------------------------------------------------------------*/
300 /* その他関数(4)                                                            */
301 /*  モジュール名  start_warikomi                                            */
302 /*  処理概要      タイマーitu4のスタート                                    */
303 /*  引数          なし                                                      */
304 /*  戻り値        なし                                                      */
305 /*--------------------------------------------------------------------------*/
306 void start_warikomi( void )
307 {
308         ITU.TSTR.BIT.STR4 = 1; /* itu4スタート */
309 }
310
311
312 /*--------------------------------------------------------------------------*/
313 /* その他関数(5)                                                            */
314 /*  モジュール名  end_warikomi                                              */
315 /*  処理概要      タイマーitu4のストップ                                    */
316 /*  引数          なし                                                      */
317 /*  戻り値        なし                                                      */
318 /*--------------------------------------------------------------------------*/
319 void end_warikomi( void )
320 {
321         ITU.TSTR.BIT.STR4 = 0; /* itu4ストップ */
322 }
323
324
325 /*--------------------------------------------------------------------------*/
326 /* その他関数(6)                                                            */
327 /*  モジュール名 課題20用 int_imia4                                         */
328 /*  処理概要     LED出力処理                                                */
329 /*  引数         なし                                                       */
330 /*  戻り値       なし                                                       */
331 /*--------------------------------------------------------------------------*/
332 void int_imia420( void )
333 {
334
335       /*--------------------------*/
336       /* 通常点灯                 */
337       /*--------------------------*/
338          disp_led( led_position );   /* handan()でLED点灯位置が入る */
339
340       /*-------------------------*/
341       /* reset interrupt request */
342       /*-------------------------*/
343          ITU4.TSR.BIT.IMFA = 0;
344 }
```

```
345
346
347 /*-------------------------------------------------------------------------*/
348 /* その他関数(7)                                                           */
349 /*  モジュール名 課題2用 int_imia4                                         */
350 /*  処理概要     ブリンク処理                                              */
351 /*  引数         なし                                                      */
352 /*  戻り値       なし                                                      */
353 /*-------------------------------------------------------------------------*/
354 void int_imia42( void )
355 {
356
357         if ( brink_flag == 0 ) {
358           /*--------------------------*/
359           /* 通常点灯                 */
360           /*--------------------------*/
361
362                     disp_led( led_position );   /* handan()でLED点灯位置が入る */
363
364                     hiyoji_flag = ON;           /* 次は点灯。点滅処理になったらまず消灯 */
365
366
367         } else {
368             /*--------------------------*/
369             /* 点滅点灯                 */
370             /*--------------------------*/
371
372                     if( hiyoji_flag == ON ) {       /* 前回は表示した。∴今回は消灯 */
373                      /*-----------------------*/
374                      /* 点滅処理    消灯      */
375                      /*-----------------------*/
376                         KIBAN081 = ~0x00;
377
378
379                             itu4_count++;           /* 消灯を30ms続ける */
380                             if( itu4_count == 3 ) {
381                                 hiyoji_flag = OFF;  /* 点滅処理 今回は消灯。次は点灯 */
382                                 itu4_count = 0;
383                             }
384
385
386                     } else {                    /* 前回は消灯した。∴今回は点灯 */
387                      /*-----------------------*/
388                      /* 点滅処理     点灯      */
389                      /*-----------------------*/
390
391                             disp_led( led_position );       /* handan()でLED点灯位置が入る */
392
393                             itu4_count++;           /* 点灯を30ms続ける。 */
394                             if( itu4_count == 3 ) {
395                                 hiyoji_flag = ON;  /* 点滅処理 今回点灯。次は消灯 */
396                                 itu4_count = 0;
397                             }
398
399
400                     }
401
402
403           }
404
405       /*-------------------------*/
406       /* reset interrupt request */
407       /*-------------------------*/
408          ITU4.TSR.BIT.IMFA = 0;
409 }
410
411
412 /*-------------------------------------------------------------------------*/
413 /* その他関数(8)                                                           */
414 /*  モジュール名 課題3用 int_imia4                                         */
415 /*  処理概要     7セグメントLED出力処理                                    */
416 /*  引数         なし                                                      */
417 /*  戻り値       なし                                                      */
418 /*-------------------------------------------------------------------------*/
419 void int_imia43( void )
420 {
421
422       /*--------------------------*/
423       /* 通常点灯                 */
424       /*--------------------------*/
425          disp_7seg( distance );   /* handan()でLED点灯位置が入る */
426
427
428       /*-------------------------*/
429       /* reset interrupt request */
430       /*-------------------------*/
```

```
431           ITU4.TSR.BIT.IMFA = 0;
432 }
433
434
435 /*------------------------------------------------------------------------------*/
436 /* その他関数(9)                                                                */
437 /*  モジュール名 課題4用 int_imia4                                              */
438 /*  処理概要                                                                    */
439 /*  引数         なし                                                           */
440 /*  戻り値       なし                                                           */
441 /*------------------------------------------------------------------------------*/
442 void int_imia44( void )
443 {
444
445               if( ledor7seg_flag == HIKARI_ON ) {
446                     /*---------------------------*/
447                     /* LED出力                   */
448                     /*---------------------------*/
449                        disp_led( led_position );   /* handan()でled_position更新 */
450
451                } else {
452                     /*---------------------------*/
453                     /* 7segLED出力               */
454                     /*---------------------------*/
455                       disp_7seg( distance );       /* handan()でdistance更新 */
456
457                }
458
459         /*-------------------------*/
460         /* reset interrupt request */
461         /*-------------------------*/
462            ITU4.TSR.BIT.IMFA = 0;
463 }
464
465
466 /*------------------------------------------------------------------------------*/
467 /* その他関数(10)                                                               */
468 /*  モジュール名 課題6用 int_imia4                                              */
469 /*  処理概要                                                                    */
470 /*  引数         なし                                                           */
471 /*  戻り値       なし                                                           */
472 /*------------------------------------------------------------------------------*/
473 void int_imia4( void )
474 {
475           itu4_count++;
476           if( itu4_count == 1 ) {    /* 10ms毎切り替える */
477
478
479               if( ledor7seg_flag == LED ) {
480
481                      /*---------------------------*/
482                      /* 7セグメントLED出力        */
483                      /*---------------------------*/
484                        disp_7seg( distance );
485                        ledor7seg_flag = T7SEG;  /* 今7segLEDに出力したよ */
486
487               } else {
488                     /*---------------------------*/
489                     /* LED出力                   */
490                     /*---------------------------*/
491                        disp_led( led_position );
492                        ledor7seg_flag = LED;  /* 今LEDに出力したよ */
493
494               }
495
496             itu4_count = 0;
497
498           }
499
500         /*-------------------------*/
501         /* reset interrupt request */
502         /*-------------------------*/
503            ITU4.TSR.BIT.IMFA = 0;
504 }
505
506
507 /*------------------------------------------------------------------------------*/
508 /* その他関数(11)                                                               */
509 /*  モジュール名  seg_cls                                                       */
510 /*  処理概要      7セグメントLED消去                                            */
511 /*  引数          なし                                                          */
512 /*  戻り値        なし                                                          */
513 /*------------------------------------------------------------------------------*/
514 void seg_cls( void )
515 {
516         /* KIBAN081= 0xff;*/ /* KIBAN081 = ~0x00; */
```

```
517
518 }
```