```c
// *************************
// *** Ｈ８ボード動作確認 ***
// *************************
#include <mes2.h>

/* ************ */
/* *** main *** */
/* ************ */
int
main(void)
{
        printf("¥n Hello, H8-3069 world! ¥n");
}
```

```
1  // ***************************************
2  // *** リセットスイッチ動作確認          ***
3  // ***    produced by Y. Mori            ***
4  // ***       special thanks to A. Ruike ***
5  // ***************************************
6  #include <mes2.h>
7  #include "r3069.h"
8
9  static int  End_flag;
10
11
12 // ************************
13 // *** 割り込み処理関数 ***
14 // *** リセット用       ***
15 // ************************
16 #pragma interrupt
17 void
18 prg_end(void)
19 {
20     load_segment( 7 );
21     End_flag= 1;
22 }
23
24
25 /* ************ */
26 /* *** main *** */
27 /* ************ */
28 int
29 main(void)
30 {
31     End_flag = 0;
32
33     // NMI 割り込みの設定（リセット用）
34     // SYSCRレジスタのNMIEGビットで立下りで割り込み要求を発生させる
35     // 1 //
36     set_handler( 7, prg_end );
37
38     while(1) {
39         printf(" Now program is running ... ¥r");
40         if( End_flag == 1){
41             exit(0);
42         }
43     }
44 }
```

```c
// ******************************************
// *** タイマ割り込み の動作テストプログラム ***
// ***   8bit バージョン                    ***
// ***                    produced by Y. Mori ***
// ***   special thanks to A. Ruike, S. Kido ***
// ******************************************
// tftp timer.elf 192.168.1.1
// timer.elf

#include "r3069.h"
#include <mes2.h>

static int  Count_feed;  // 割り込み回数，外部変数とする
static int  Count_sec;   // 1[s]間で feed()に入る回数

static void  init_settings( void );


// ***************
// *** feed () ***
// ***************
#pragma interrupt
void
feed(void )
{
    load_segment(42);  // 24，ベクタ番号（大域変数を使うときに必要），pp.143

    Count_feed ++;

    // TSCR:タイマコントロールステータスレジスタ
    // CMFA:TCORA のコンペアマッチの発生を示すステータスフラグ
    OCT_ITU3.TCSR.BIT.CMFA = 0;
}


// ***************
// *** main () ***
// ***************
int
main(void)
{
    int i;

    init_settings();

    while (1){
        sleep(10);  // 複数回表示されるのを防ぐため

        if(         // 01 //            ){   // 1 秒ごとに表示
            printf(" %d [s]¥r", (int)(Count_feed/Count_sec) );
        }
    }
}

// -------------------------
// --- H8 の初期設定関数 ---
// -------------------------
static void
init_settings( void )
{       // 8 ビットタイマの ITU3 を用いる
    OCT_ITU3.TCNT  = 0;       // カウンタ，TCORA と 8TCNT の値は常に比較されている
        // 02 //              // 周期 20ms = ? Hz になるように回数を設定
    OCT_ITU3.TCSR.BYTE = 0x00;
        // 03 //              // カウンタクリア要因：コンペアマッチ A によりクリア
        // 04 //              // CMFA による割り込み許可
        // 05 //              // クロックセレクト：φ/8192

    set_handler( 42, feed );

    Count_feed= 0;
    Count_sec=  // 06 // ;       // 1[s]間で feed()に入る回数
}
```

```
1   // ***************************************
2   // *** motor の動作テストプログラム      ***
3   // ***    produced by Y. Mori            ***
4   // ***       special thanks to A. Ruike ***
5   // ***************************************
6   // tftp motor.elf 192.168.1.1
7   // motor.elf
8   #include "r3069.h"
9   #include <mes2.h>
10
11  // --- motor ---
12  #define  MOTOR_0_CW     P4.DR.BIT.B0= 1; P4.DR.BIT.B2= 0;
13  #define  MOTOR_0_CCW    P4.DR.BIT.B0= 0; P4.DR.BIT.B2= 1;
14  #define  MOTOR_0_RUN    P4.DR.BIT.B0= 0; P4.DR.BIT.B2= 0;
15  #define  MOTOR_0_BREAK  P4.DR.BIT.B0= 1; P4.DR.BIT.B2= 1;
16  #define  MOTOR_0_DUTY   OCT_ITU0.TCORB
17
18  #define  Max_duty    250
19  #define  Limit_duty  250
20  #define  Ratio_duty (Max_duty/Limit_duty)
21
22  // --- Joint number ---
23  #define  LW  0  //  Left  wheel
24
25  // --------------
26  // --- 関数群 ---
27  // --------------
28  static void  init_settings();
29  static void  motor( int no, int duty );
30
31  // ***************
32  // *** main () ***
33  // ***************
34  int
35  main( void )
36  {
37      int jnt;
38      int duty;
39      int num;
40      int confirm= 1;
41      int i;
42
43          // 1 //    // H8 の初期設定
44
45      num = LW;
46
47      motor( num, 0 );
48
49      // motor check
50      printf("¥n duty(%d, %d)= ", -Max_duty, Max_duty);
51      scanf( "%d", &duty );
52
53      motor( num, duty );
54
55      while( confirm != 'q' ){
56          printf(" q key: quit      ¥r");
57          confirm= getchar();
58      }
59
60      motor( num, 0 );
61
62      exit(0);
63  }
64
65
66  // ------------------------
67  // --- H8 の初期設定関数 ---
68  // ------------------------
69  static void
70  init_settings( void )
71  {
72      // --- ポート入出力設定 ---
73      P4.DDR= 0xff;  // output
74
```

```
75      // ***************
76      // *** for Left ***
77      // ***************       (8 ビットタイマの ITU0 を用いる)
78      // --- 8bit timer pwm settings 3069 マニュアル pp.482(上のページ) 参照 ---
79      // TCR: タイマコントロールレジスタ
80          // 02 //       // [φ/64 ] に設定
81
82      // TCNT をコンペアマッチ A でクリア: CCLR0=1, CCLR1=0
83          // 03//
84
85      // TCSR: タイマコントロール/ステータスレジスタ
86      // コンペアマッチ B で 0 出力: OIS3 と OIS2 を 01 に設定
87          // 04 //
88
89      // コンペアマッチ A で 1 出力: OS1 と OS0 を 10 に設定
90          // 05 //
91
92      // TCRB: タイムコンスタントレジスタ B
93      // Duty        Duty 比の初期値を 0 に
94      OCT_ITU0.TCORB= 0;
95
96      // TCRB: タイムコンスタントレジスタ A
97      // 周波数: 20MHz(Clock)/64(φ)/250(TCORA)= 1.25kHz
98      OCT_ITU0.TCORA= Max_duty;
99
100     // スタート
101     MOTOR_0_RUN;
102 }
103
104 // ----------------------
105 // --- モータ駆動関数 ---
106 // ----------------------
107 static void
108 motor( int no, int duty )
109 {
110     int  real_duty= 0;
111
112     switch( no ){
113
114     // for Left
115     case LW:
116     if( duty > 0 ){
117         real_duty= duty*Ratio_duty;
118
119         if( real_duty >= Max_duty ){
120             MOTOR_0_CW;
121             MOTOR_0_DUTY= Max_duty;
122         }
123         else{
124             MOTOR_0_CW;
125             MOTOR_0_DUTY= real_duty;
126         }
127     }
128     else if(duty == 0){
129         MOTOR_0_DUTY= Max_duty;   // 0:ｽﾄｯﾌﾟ（ゆっくり止まる）, Max_duty:ﾌﾞﾚｰｷ（急に止まる）
130         MOTOR_0_BREAK;
131     }
132     else{
133         real_duty= - duty*Ratio_duty;
134
135         if( real_duty >= Max_duty ){
136             MOTOR_0_CCW;
137             MOTOR_0_DUTY= Max_duty;
138         }
139         else{
140             MOTOR_0_CCW;
141             MOTOR_0_DUTY= real_duty;
142         }
143     }
144     break;
145
146     default: break;
147     }
148 }
```

```
1   // ****************************************
2   // *** encoder 読み込みテストプログラム    ***
3   // ***                  produced by Y. Mori ***
4   // *** special thanks to A. Ruike, S. Kido ***
5   // ****************************************
6   // tftp enco.elf 192.168.1.1
7   // enco.elf
8
9   #include "r3069.h"
10  #include <mes2.h>
11  #include "enc_common.h"
12
13  ct_sharedType  Ct;
14
15  // --------------
16  // --- 関数群 ---
17  // --------------
18  static void  init_settings();
19  static void  initialize_para( void );
20  static int   enco( int no );  // この H8 ﾎﾞｰﾄﾞでは，int=long の扱い
21
22
23  // ***************
24  // *** main () ***
25  // ***************
26  int
27  main( void )
28  {
29      // 01 //   // H8 の初期設定
30      // 02 //   // パラメータの初期化設定
31
32      while(1){
33          printf(" %d  %d ¥r", enco(LW), enco(RW) );
34      }
35
36      exit(0);
37  }
38
39
40  // -------------------------
41  // --- H8 の初期設定関数 ---
42  // -------------------------
43  static void
44  init_settings( void )
45  {
46      // *** for LW ***
47      // --- ITU2 位相係数モード設定 ---
48          // 03 //   // 位相係数モードを設定
49          // 04 //   // オーバフローのみを検知
50      HEX_ITU2.TCNT= 0;
51          // 05 //   // タイマスタート（カウント動作）
52  }
53
54
55  // --------------------------
56  // --- パラメータの初期化 ---
57  // --------------------------
58  static void
59  initialize_para( void )
60  {
61      int  jnt;
62
63      // ----------------------
64      // --- encoder の設定 ---
65      // ----------------------
66      Ct.enco[LW].present.d= 0; // 位置の初期化
67      Ct.tmp_enco_val[LW]  = 0; // エンコーダのカウンタの初期化
68  }
69
70
71
72
73
74
```

```
75  // ------------------------------
76  // --- エンコーダ値の読込関数 ---
77  // ------------------------------
78  static int
79  enco(int no)
80  {
81      // エンコーダの位置(d)の初期値を取得
82      // 関数を呼び出したときの現在位置を初期値として取得
83          // 06 //
84
85      if( no==LW ){
86      // エンコーダの現在の位置を取得
87          Ct.enco[no].present.d= - HEX_ITU2.TCNT; // モーターLW は 4 逓倍
88      }
89
90      if( Ct.enco[no].present.d > 32767 ){
91          Ct.enco[no].present.d -= 65536;
92      }
93
94      // エンコーダの現在値と初期値の差(delta)を計算
95          // 07 //
96
97      // エンコーダの差分値が下限を超えたら
98          // 08 //           {
99          //   最大値を足す
100             // 09 //
101     }
102
103     // エンコーダの差分値が上限を超えたら
104         // 10 //           {
105         //   最大値を引く
106             // 11 //
107     }
108
109     // エンコーダの出力値に現在の差分値を足して新たな出力値とする
110         // 12 //
111
112     return( Ct.tmp_enco_val[no] );
113 }
```

```
// ****************************************
// *** enc_common.h (共通変数，共通定数) ***
// ***            produced by Y. Mori     ***
// ****************************************
#ifndef __ENC_COMMON_H
#define __ENC_COMMON_H

// ************************
// *** Common Parameter ***
// ************************

// --- Joint number ---
#define  Jnum  2
#define  LW  0  //  Left wheel
#define  RW  1  // Right wheel


// ***********************
// *** For Controller ***
// ***********************

// --- encoder count type ---
typedef struct{
        int  d, delta;  // pos, delta

} ct_eCountType;


// --- encoder type ---
typedef struct{
        ct_eCountType  present, last;

} ct_encoType;


// --- control shared type ---
typedef struct{
        ct_encoType   enco[Jnum];

        int    enco_dir[Jnum];      // エンコーダの回転方向
        int    tmp_enco_val[Jnum];  // エンコーダカウント用仮変数

} ct_sharedType;

#endif




/* ct_sharedType +
                  |.enco_dir[Jnum]
                  |.tmp_enco_val[Jnum]
                  |.enco[Jnum] +
                                |.present +          (現在値)
                                |         |.d          (位置)
                                |         |.delta      (差分値)
                                |.last +             (過去値 または 初期値)
                                        |.d            (位置)
                                        |.delta        (差分値)
*/
```

```
1   // ****************************************
2   // *** Port 読み込みテストプログラム       ***
3   // ***                  produced by Y. Mori ***
4   // *** special thanks to A. Ruike, S. Kido ***
5   // ****************************************
6   #include "r3069.h"
7   #include <mes2.h>
8
9   // ***************
10  // *** main () ***
11  // ***************
12  int
13  main(void){
14
15      // ポートの 5 番を全て入力ポートに初期化する(値は 2 桁の 16 進数表示)
16      // 01 //
17
18      while(1){
19          // 02 には DR レジスタの B2 を，03 には B3 の値を出力する
20          printf(" P5: No.2 = %d    No.3 = %d ¥r", // 02 //, // 03 // );
21      }
22  }
```

```
1   // ***************************************
2   // ***  ＡＤポートの動作確認              ***
3   // ***     produced by Y. Mori            ***
4   // ***        special thanks to A. Ruike ***
5   // ***************************************
6   // ● H8 ボードの JP2, JP3 を配線すること
7
8   #include "r3069.h"
9   #include <mes2.h>
10
11  /* ************ */
12  /* *** main *** */
13  /* ************ */
14  int
15  main(void)
16  {
17      int  ad_data, ad_volt;
18
19      // AD ポートの初期化
20
21      // ADCSR レジスタを"高速モード(70 ステート)"を選択するように BYTE を用いて設定（以下参照）
22          // 01 //
23      // 00001000(8 桁の 2 進数)   //  8 桁の 2 進数→2 桁の 16 進数表現にして値を設定
24      // ||||||||_AN0
25      // |||||_  変換時間 高速モード
26      // ||||_   単一モード
27      // |||_    A/D 変換停止
28      // ||_     A/D 変換割り込み禁止
29      // |_      A/D 変換終了フラグ
30
31      while (1){
32          // ADCSR レジスタの ADST ビットにより，AD 変換スタート
33              // 02 //
34          while (AD.ADCSR.BIT.ADF == 0);   // 変換終了まで待つ
35          ad_data = AD.ADDRA >> 6;  // ビットシフト
36          AD.ADCSR.BIT.ADF = 0;
37
38          ad_volt = (int)((5.13f/1024.0f)*1000.0f*ad_data + 0.5f);  //5.13[V]は入力電圧(実測)
39
40          printf(" %4d   %4d [mV]¥r", ad_data, ad_volt);
41      }
42  }
```

```
1   // ***************************************
2   // *** ファイル入出力の動作確認         ***
3   // ***    produced by Y. Mori           ***
4   // ***       special thanks to A. Ruike ***
5   // ***************************************
6   // ・tftp - 192.168.1.1 data*.txt (*は，file_number) で，
7   //   PC のプログラムのあるフォルダにファイルが転送される．
8   //
9   // ・dir で，H8 の中のファイルが見える．
10  //
11  // ・type data*.txt で，data*.txt の内容を参照できる．
12
13  #include <mes2.h>
14
15  /* ************ */
16  /* *** main *** */
17  /* ************ */
18  int
19  main(void)
20  {
21      static int   fd;
22      static char  buf[256];  // static をつけないと，サイズが大きいときにｺﾝﾊﾟｲﾙｴﾗｰが起きる
23      int  i;
24
25      fd= open("data.txt", OptWrite);
26
27      if( fd == -1 ){
28          printf(" File open error! ¥n");
29          exit(-1);
30      }
31
32      for( i=0; i<=10; i++ ){
33          sprintf(buf, "%3d %3d ¥r¥n", i, i*10);
34          write( fd, buf, strlen(buf) );
35      }
36
37      close(fd);
38  }
```