```
/************************************************************************/
/************************************************************************/
/***                                                                  ***/
/***                        RX220マイコンの初期設定例                 ***/
/***                              hwsetup.c                           ***/
/***                                                                  ***/
/************************************************************************/
/************************************************************************/
/* ※ルネサスエレクトロニクスのアプリケーションノート「RX220初期設定例」と    */
/*   そのプログラム例を参考にしています。                               */
/* ※ルネサスエレクトロニクスの「RX220 CubeSuite+用ルネサススタータキットの   */
/*   サンプルコード」を参考にしています。                               */

#include <stdlib.h>
#include "iodefine.h"
#include "typedefine.h"

#include "NonExistPort.h"
#include "InitSystemClock.h"

/************************************************************************/
/*                            マクロ定義                                */
/************************************************************************/

#define MAIN_CLOCK_CYCLE (1000000000L/MAIN_CLOCK_Hz)
#define SUB_CLOCK_CYCLE  (1000000000L/SUB_CLOCK_Hz)

/* 関数cmt0_waitの待機時間指定について ***************************************/
/* 注意 : CMT0は、カウントソースPCKが低速オンチップオシレータLOCOになっている */
/*      : 状態で使用する。                                              */
/*      : LOCOの周波数 = min112.5kHz, typ125kHz, max137.5kHz            */
/*      : CMT0のクロック選択でPCLK/32を使用するため、CMT0の1カウントの時間は  */
/*      : 約232,727nS（max137.5kHzを想定しておくこと）となる。          */

#define FOR_CMT0_TIME    (232727)       /* CMT0の1カウントの時間は約232,727ns */

/************************************************************************/
/*                          CMT0 によるソフトウェアウェイト             */
/************************************************************************/
/* 注意 : プロテクトレジスタ(PRCR.PRC1=1)をプロテクト解除状態にしておくこと。 */
/*      : そうしないと、モジュールストップコントロールレジスタA(MSTPCRA)への  */
/*      : 書き込みができない。                                          */
/* 注意 : システムクロックがLOCO(125kHz)になっている状態で使用すること。       */
/* 機能 : 引数で指定した時間（CMT0のカウント単位）だけ待機する。           */
/* 引数 : dwWtTime = 待機時間                                           */
/*      :              (CMT0のカウント単位＝FOR_CMT0_TIME単位＝約232,727ns)  */

static void cmt0_wait(DWORD dwWtTime)
{
    /* コンペアマッチタイマ（CMT0,CMT1）のモジュールストップ解除 */
    MSTP(CMT0) = 0;                     /* SYSTEM.MSTPCRA.BIT.MSTPA15 = 0 （CMT0,CMT1共通） */

    /* CMT0カウンタのカウントを停止する */
    CMT.CMSTR0.BIT.STR0 = 0;            /* CMT0.CMCNTカウンタのカウント動作停止 */

    /* CMT0のクロック選択、コンペアマッチ割り込み許可 */
    CMT0.CMCR.WORD = 0x00C1;            /* クロック選択=PCLK/32、コンペアマッチ割り込み（CMIn）を許可 */
    /* CMT0タイマカウンタをクリア */
    CMT0.CMCNT = 0x0000;

    /* 指定待ち時間(dwWtTime)が０でなければ-1する */
    if (0 != dwWtTime){
        dwWtTime = dwWtTime - 1;
    }

    /* 指定待ち時間をCMT0コンペアマッチレジスタにセット */
    CMT0.CMCOR = dwWtTime;

    /* CMT0.CMI0割り込みが発生中の場合はクリアする */
    while (1 == IR(CMT0,CMI0)){
        IR(CMT0,CMI0) = 0;              /* CMT0.CMI0割り込み要求をクリアする */
    }

    /* CMT0カウンタのカウントを開始する */
    CMT.CMSTR0.BIT.STR0 = 1;            /* CMT0.CMCNTカウンタのカウント開始 */

    /* CMT0.CMI0割り込みの発生を待つ */
    while (0 == IR(CMT0,CMI0)){
    }

    /* CMT0カウンタのカウントを停止する */
    CMT.CMSTR0.BIT.STR0 = 0;

    /* CMT0コンペアマッチ割り込み禁止（初期値に戻す） */
    CMT0.CMCR.WORD = 0x0080;

    /* CMT0タイマカウンタをクリア */
```

```
88      CMT0.CMCNT = 0x0000;
89
90      /* CMT0コンペアマッチレジスタをクリア */
91      CMT0.CMCOR = 0x0000;
92
93      /* CMT0.CMI0割り込み要求をクリア */
94      IR(CMT0,CMI0) = 0;
95
96      /* コンペアマッチタイマ（CMT0,CMT1）をモジュールストップ */
97      MSTP(CMT0) = 1;                     /* SYSTEM.MSTPCRA.BIT.MSTPA15 = 0 （CMT0,CMT1共通） */
98
99  }
100
101
102 /************************************************************************/
103 /*                              サブクロック設定                        */
104 /************************************************************************/
105
106 /* サブクロックの発振 ***************************************************/
107
108 static void oscillation_subclk(void)
109 {
110
111     int i;
112     volatile BYTE dummy;
113
114 #if (SEL_SUB == B_USE) && (SEL_RTC == B_USE)
115     /* クロック発振の安定待ち */
116     cmt0_wait((WAIT_TIME_FOR_SUB_OSCILLATION/FOR_CMT0_TIME)+1); /* 安定時間４秒 */
117 #endif
118
119     /* サブクロック発振器の動作停止１ */
120     SYSTEM.SOSCCR.BYTE = 0x01;
121     /* 設定が正しく書き込まれたことを確認する */
122     while (0x01 != SYSTEM.SOSCCR.BYTE){
123     }
124     /* サブクロック発振器の動作停止２ */
125     RTC.RCR3.BIT.RTCEN = 0;
126
127     /* RTC.RCR3.RTCENのダミーリードを３回 */
128     for (i = 0; i < 3; i++){
129         dummy = RTC.RCR3.BIT.RTCEN;
130     }
131     /* RTC.RCR3.RTCENが書かれたことを確認する */
132     while (0 != RTC.RCR3.BIT.RTCEN){
133     }
134
135
136     /* サブクロックで５サイクル分を待つ */
137     cmt0_wait((SUB_CLOCK_CYCLE*5/FOR_CMT0_TIME)+1);
138
139     /* サブクロック発振器ドライブ能力設定 */
140     RTC.RCR3.BYTE = REG_RCR3;           /* 'Low CL'または'Standard CL' */
141
142     /* RTC.RCR3のダミーリードを３回 */
143     for (i = 0; i < 3; i++){
144         dummy = RTC.RCR3.BYTE;
145     }
146
147     /* RTC.RCR3が書かれたことを確認する */
148     while (REG_RCR3 != RTC.RCR3.BYTE){
149     }
150
151     /* サブクロック発振器の発振安定待機時間を選択 */
152     SYSTEM.SOSCWTCR.BYTE = REG_SOSCWTCR;/* サブロック発振器ウェイト時間設定 */
153
154     /* サブクロック発振器の動作開始 */
155     SYSTEM.SOSCCR.BYTE = 0x00;
156     /* 設定が正しく書き込まれたことを確認する */
157     while (0x00 != SYSTEM.SOSCCR.BYTE){
158     }
159
160     /* サブクロック発振の安定待ち */
161     cmt0_wait(WAIT_TIME_FOR_SUB_OSCILLATION/FOR_CMT0_TIME+1);   /* 安定時間４秒 */
162
163 }
164
165 /* RTC を使用する場合の初期化 *******************************************************/
166
167 static void enable_RTC(void)
168 {
169
170 #if (SEL_RTC == B_USE)
171
172     int i;
173     volatile BYTE dummy;
174
```

```
175     /* サブクロック発振器動作開始 */
176     RTC.RCR3.BIT.RTCEN = 1;             /* サブクロック発振器動作 */
177
178     /* RTC.RCR3.RTCENを３回ダミーリード */
179     for (i = 0; i < 3; i++){
180         dummy = RTC.RCR3.BIT.RTCEN;
181     }
182     /* 設定が正しく書き込まれたことを確認する */
183     while (1 != RTC.RCR3.BIT.RTCEN){
184     }
185
186     /* サブクロックで６サイクルを待つ */
187     cmt0_wait((SUB_CLOCK_CYCLE*6/FOR_CMT0_TIME)+1);
188
189     /* RTCのプリスケーラおよびカウンタ（時計）の停止 */
190     RTC.RCR2.BIT.START = 0;             /* 年、月、曜日、日、時、分、秒、64Hzカウンタおよびプリスケーラは停止 *
    /
191
192     /* 設定が正しく書き込まれたことを確認する */
193     while (0 != RTC.RCR2.BIT.START){
194     }
195
196     /* RTCのカレンダカウントモードをセット */
197     RTC.RCR2.BIT.CNTMD = SEL_CNTMD;     /* RTC コントロールレジスタ2（RCR2） */
198     /* 設定が正しく書き込まれたことを確認する */
199     while (SEL_CNTMD != RTC.RCR2.BIT.CNTMD){
200     }
201
202     /* ---- RTC Software Reset ---- */
203     RTC.RCR2.BIT.RESET = 1;             /* プリスケーラおよびRTCソフトウェアリセット対象レジスタをリセット */
204     /* RTC.RCR2.RESETが"0"であることを確認 */
205     while (0 != RTC.RCR2.BIT.RESET){
206     }
207
208 #endif
209 }
210
211 /* サブクロックのウェイトコントロールレジスタの再設定 ************************************************/
212
213 static void resetting_wtcr_subclk(void)
214 {
215
216 #if (SEL_RTC == B_USE) && (SEL_SUB == B_USE)
217
218     /* サブクロック発振器の動作停止 */
219     SYSTEM.SOSCCR.BYTE = 0x01;
220     /* 設定が正しく書き込まれたことを確認する */
221     while (0x01 != SYSTEM.SOSCCR.BYTE){
222     }
223
224     /* サブクロックで５サイクル分を待つ */
225     cmt0_wait((SUB_CLOCK_CYCLE*5/FOR_CMT0_TIME)+1);
226
227     /* サブクロック発振器の発振安定待機時間を選択 */
228     SYSTEM.SOSCWTCR.BYTE = 0x00;        /* サブロック発振器ウェイト時間設定（2s + 2サイクル） */
229
230     /* サブクロック発振器の動作開始 */
231     SYSTEM.SOSCCR.BYTE = 0x00;
232     /* 設定が正しく書き込まれたことを確認する */
233     while (0x00 != SYSTEM.SOSCCR.BYTE){
234     }
235
236     /* サブクロック発振の安定待ち */
237     cmt0_wait((SUB_CLOCK_CYCLE*2/FOR_CMT0_TIME)+1 );    /* 待機時間はサブクロックで２サイクル（約61μs） */
238
239 #endif
240 }
241
242 /* サブクロックをシステムクロックとして使用しない場合の設定 ******************************/
243
244 static void no_use_subclk_as_sysclk(void)
245 {
246
247 #if (SEL_RTC == B_USE) && (SEL_SUB != B_USE)
248
249     /* サブクロック発振器の動作停止 */
250     SYSTEM.SOSCCR.BYTE = 0x01;
251     /* 設定が正しく書き込まれたことを確認する */
252     while (0x01 != SYSTEM.SOSCCR.BYTE){
253     }
254
255 #endif
256 }
257
258 /* サブクロックの発振設定 ********************************************************************/
259
260 void CGC_oscillation_sub(void)
```

```
{

    /* サブクロックの発振 */
    oscillation_subclk();

    /* RTCを使用する場合の初期化 */
    enable_RTC();

    /* サブクロックのウェイトコントロールレジスタの再設定 */
    resetting_wtcr_subclk();

    /* サブクロックをシステムクロックとして使用しない場合の設定 */
    no_use_subclk_as_sysclk();

}

/* サブクロックの停止設定 */

void CGC_disable_subclk(void)
{

    int i;
    volatile BYTE dummy;

    /* サブクロック発振器の動作停止１ */
    SYSTEM.SOSCCR.BYTE = 0x01;
    /* 設定が正しく書き込まれたことを確認する */
    while (0x01 != SYSTEM.SOSCCR.BYTE){
    }

    /* サブクロック発振器の動作停止２ */
    RTC.RCR3.BIT.RTCEN = 0;

    /* RTC.RCR3.RTCENのダミーリードを３回 */
    for (i = 0; i < 3; i++){
        dummy = RTC.RCR3.BIT.RTCEN;
    }
    /* RTC.RCR3.RTCENが書かれたことを確認する */
    while (0 != RTC.RCR3.BIT.RTCEN){
    }

}


/*****************************************************************************/
/*                                メインクロック設定                           */
/*****************************************************************************/

/* メインクロックの発振設定 */

static void CGC_oscillation_main(void)
{
    /* メインクロック発振器ドライブ能力を設定 */
    SYSTEM.MOFCR.BYTE = REG_MOFCR;

    /* メインクロック発振器の発振安定待機時間を選択（発振子の発振周波数×設定サイクル数） */
    /* 注意 : メインクロック発振安定時間（tMAINOSC）以上の待機時間となるように設定すること */
    /* 注意 : MOSCWTCR レジスタは、MOSCCR.MOSTP ビットが"1"のときのみ書き換え可能 */
    SYSTEM.MOSCWTCR.BYTE = REG_MOSCWTCR;     /* メインクロック発振器ウェイト時間設定（131072サイクル） */
    /* メインクロック発振器の動作を開始する */
    /* 注意 : MOSCWTCRを設定してから本レジスタを設定すること */
    /* 注意 : メインクロック発振器動作の開始および停止に関して各種制限あり（マニュアル参照） */
    SYSTEM.MOSCCR.BYTE = 0x00;
    /* 設定が正しく書き込まれたことを確認する */
    while (0x00 != SYSTEM.MOSCCR.BYTE){
    }

    /* メインクロック発振の安定待ち */
    cmt0_wait((WAIT_TIME_FOR_MAIN_OSCILLATION/FOR_CMT0_TIME)+1);        /* 13.1msで安定する */

}

/*****************************************************************************/
/*                          高速オンチップオシレータ(HOCO)設定                  */
/*****************************************************************************/
/* HOCO クロックの発振設定 */

void CGC_oscillation_HOCO(void)
{
    /* 高速オンチップオシレータ(HOCO)の周波数を選択 */
    /* 注意 : HOCOCR.HCSTP ビットが“0”（HOCO 動作）のとき、HOCOCR2 レジスタへの書き込みは禁止 */
    SYSTEM.HOCOCR2.BYTE = REG_HOCOCR2;
    /* 高速オンチップオシレータ(HOCO)の発振安定待機時間を選択 */
    /* このレジスタはPRCR.PRC1ビットを“1”（書き込み許可）にした後で書き換えてください */
    SYSTEM.HOCOWTCR2.BYTE = REG_HOCOWTCR2;
    /* 高速オンチップオシレータを動作開始 */
    /* 注意 : HOCOCRはPRCR.PRC0=1（書き込み許可）にした後で書き換えること */
```

```
348     /* 注意 : HOCOCRはHOCOWTCR2を設定してから設定すること */
349     SYSTEM.HOCOCR.BYTE = 0x00;
350     /* 高速オンチップオシレータ発振の安定待ち */
351     cmt0_wait((WAIT_TIME_FOR_HOCO_OSCILLATION/FOR_CMT0_TIME)+1);    /* 20uSで安定する */
352 }
353
354 /*****************************************************************************/
355 /*                              クロック初期設定                             */
356 /*****************************************************************************/
357
358 static void Init_SystemClock(void)
359 {
360     /* クロック発生回路関連レジスタへの書き込みを許可する */
361     SYSTEM.PRCR.WORD = 0xA503;          /* クロック発生回路関連レジスタ、 */
362                                         /* 動作モード、消費電力低減機能、 */
363                                         /* ソフトウェアリセット関連レジスタへの書き込み許可 */
364
365 /* 高速オンチップオシレータを使用しない場合 */
366 #if (SEL_HOCO == B_NOT_USE)
367     SYSTEM.HOCOPCR.BYTE = 0x01;         /* HOCOの電源OFF */
368 #endif
369
370     /* サブクロックを設定する */
371 #if (SEL_SUB == B_USE) || (SEL_RTC == B_USE)
372     CGC_oscillation_sub();
373 #else
374     CGC_disable_subclk();
375 #endif
376
377     /* メインクロックを設定する */
378 #if (SEL_MAIN == B_USE)
379     CGC_oscillation_main();
380 #endif
381
382     /* HOCOクロックを設定する */
383 #if (SEL_HOCO == B_USE)
384     CGC_oscillation_HOCO();
385 #endif
386
387     /* システムクロックの周波数選択 */
388     SYSTEM.SCKCR.LONG = REG_SCKCR;
389     /* 設定が正しく書き込まれたことを確認する */
390     while (REG_SCKCR != SYSTEM.SCKCR.LONG){
391     }
392
393     /* システムクロックソース選択（初期状態では125kHzのLOCOになっている） */
394     SYSTEM.SCKCR3.WORD = SEL_SYSCLK;
395     /* 設定が正しく書き込まれたことを確認する */
396     while (SEL_SYSCLK != SYSTEM.SCKCR3.WORD){
397     }
398
399     /* 動作電力制御モードを選択 */
400     SYSTEM.OPCCR.BYTE = REG_OPCCR;
401     /* 動作電源制御モード遷移が完了したことを確認する*/
402     while (0 != SYSTEM.OPCCR.BIT.OPCMTSF){
403     }
404
405     /* クロック発生回路関連レジスタへの書き込みを禁止する */
406     SYSTEM.PRCR.WORD = 0xA500;          /* クロック発生回路関連レジスタ、     */
407                                         /* 動作モード、消費電力低減機能、     */
408                                         /* ソフトウェアリセット関連レジスタ、 */
409                                         /* LVD関連レジスタへの書き込み禁止    */
410 }
411
412 /*****************************************************************************/
413 /*              全てのI/Oポートを汎用入出力ポートに設定                      */
414 /*****************************************************************************/
415 /* 機能 : 全I/OポートのモードレジスタPORTm.PMRを初期状態（汎用入出力ポート）  */
416 /*      : にする。                                                           */
417
418 static void Init_PortModeReg(void)
419 {
420     /* ポートモードレジスタPORT0,1,2,3,4,5,A,B,C,D,E,H,J（初期状態：全ピン汎用入出力ポート） */
421     PORT0.PMR.BYTE = 0x00;
422     PORT1.PMR.BYTE = 0x00;
423     PORT2.PMR.BYTE = 0x00;
424     PORT3.PMR.BYTE = 0x00;
425     PORT4.PMR.BYTE = 0x00;
426     PORT5.PMR.BYTE = 0x00;
427     PORTA.PMR.BYTE = 0x00;
428     PORTB.PMR.BYTE = 0x00;
429     PORTC.PMR.BYTE = 0x00;
430     PORTD.PMR.BYTE = 0x00;
431     PORTE.PMR.BYTE = 0x00;
432     PORTH.PMR.BYTE = 0x00;
433     PORTJ.PMR.BYTE = 0x00;
434 }
```

```
435
436
437 /************************************************************************/
438 /*                          存在しないI/Oポートの初期化                 */
439 /************************************************************************/
440 /* 注：P35端子 (NMI) は入力専用のため、PORT3.PDR.B5ビットは設定不要。      */
441
442 static void Init_NonExistPort(void)
443 {
444     /* ポート０(PORT0) */
445 #if DEF_P0PDR != 0x00
446     PORT0.PDR.BYTE = PORT0.PDR.BYTE | DEF_P0PDR;
447 #endif
448     /* ポート１(PORT1) */
449 #if DEF_P1PDR != 0x00
450     PORT1.PDR.BYTE = PORT1.PDR.BYTE | DEF_P1PDR;
451 #endif
452     /* ポート２(PORT2) */
453 #if DEF_P2PDR != 0x00
454     PORT2.PDR.BYTE = PORT2.PDR.BYTE | DEF_P2PDR;
455 #endif
456     /* ポート３(PORT3) */
457 #if DEF_P3PDR != 0x00
458     PORT3.PDR.BYTE = PORT3.PDR.BYTE | DEF_P3PDR;
459 #endif
460     /* ポート４(PORT4) */
461 #if DEF_P4PDR != 0x00
462     PORT4.PDR.BYTE = PORT4.PDR.BYTE | DEF_P4PDR;
463 #endif
464     /* ポート５(PORT5) */
465 #if DEF_P5PDR != 0x00
466     PORT5.PDR.BYTE = PORT5.PDR.BYTE | DEF_P5PDR;
467 #endif
468     /* ポートＡ(PORTA) */
469 #if DEF_PAPDR != 0x00
470     PORTA.PDR.BYTE = PORTA.PDR.BYTE | DEF_PAPDR;
471 #endif
472     /* ポートＢ(PORTB) */
473 #if DEF_PBPDR != 0x00
474     PORTB.PDR.BYTE = PORTB.PDR.BYTE | DEF_PBPDR;
475 #endif
476     /* ポートＣ(PORTC) */
477 #if DEF_PCPDR != 0x00
478     PORTC.PDR.BYTE = PORTC.PDR.BYTE | DEF_PCPDR;
479 #endif
480     /* ポートＤ(PORTD) */
481 #if DEF_PDPDR != 0x00
482     PORTD.PDR.BYTE = PORTD.PDR.BYTE | DEF_PDPDR;
483 #endif
484     /* ポートＥ(PORTE) */
485 #if DEF_PEPDR != 0x00
486     PORTE.PDR.BYTE = PORTE.PDR.BYTE | DEF_PEPDR;
487 #endif
488     /* ポートＨ(PORTH) */
489 #if DEF_PHPDR != 0x00
490     PORTH.PDR.BYTE = PORTH.PDR.BYTE | DEF_PHPDR;
491 #endif
492     /* ポートＪ(PORTJ) */
493 #if DEF_PJPDR != 0x00
494     PORTJ.PDR.BYTE = PORTJ.PDR.BYTE | DEF_PJPDR;
495 #endif
496 }
497
498 /************************************************************************/
499 /*                          DMAC/DTCをモジュールストップする            */
500 /************************************************************************/
501
502 static void Stop_DMACDTCModule(void)
503 {
504     /* 動作モード、消費電力低減機能、ソフトウェアリセット関連レジスタへの書き込みを許可する */
505     SYSTEM.PRCR.WORD = 0xA502;          /* 動作モード、消費電力低減機能、ソフトウェアリセット関連レジスタへの書
    き込み許可 */
506     /* DMAC,DTCをモジュールストップ状態にする */
507     MSTP(DTC) = 1;                      /* MSTP(DMAC)=1; でも同じ (DTC,DMAC0,DMAC1,DMAC2,DMAC3は共通のMSTPCRA.M
    STPA28ビット)  */
508
509     /* 動作モード、消費電力低減機能、ソフトウェアリセット関連レジスタへの書き込みを禁止する */
510     SYSTEM.PRCR.WORD = 0xA500;          /* クロック発生回路関連レジスタ、      */
511                                         /* 動作モード、消費電力低減機能、      */
512                                         /* ソフトウェアリセット関連レジスタ、  */
513                                         /* LVD関連レジスタへの書き込み禁止     */
514 }
515
516 /************************************************************************/
517 /*                          全てをモジュールストップする                */
518 /************************************************************************/
519 /* 備考 : RAM0のモジュールストップは解除される。                          */
```

```
520
521 static void Stop_AllModule(void)
522 {
523     /* 動作モード、消費電力低減機能、ソフトウェアリセット関連レジスタへの書き込みを許可する */
524     SYSTEM.PRCR.WORD = 0xA502;          /* 動作モード、消費電力低減機能、ソフトウェアリセット関連レジスタへの書き込み許可 */
525     /* 全てモジュールストップ状態にする */
526     SYSTEM.MSTPCRA.LONG = 0x7FFFFFFF;   /* RAM0はモジュールストップしない(MSTPCRC.MSTPC0=0) */
527
528     /* 動作モード、消費電力低減機能、ソフトウェアリセット関連レジスタへの書き込みを禁止する */
529     SYSTEM.PRCR.WORD = 0xA500;          /* クロック発生回路関連レジスタ、      */
530                                         /* 動作モード、消費電力低減機能、      */
531                                         /* ソフトウェアリセット関連レジスタ、  */
532                                         /* LVD関連レジスタへの書き込み禁止     */
533 }
534
535 /*****************************************************************************/
536 /*                          電圧検出回路（LVDAa）の初期化                    */
537 /*****************************************************************************/
538 /* ※VCC < 3.10V になったとき、NMI割り込みを発生する。                       */
539 /* ※電圧検出２回路（vdet2）は未使用。                                       */
540
541 void init_voltage_detect(void)
542 {
543     volatile WORD delay = 0xFFFFu;
544
545     /* LVD関連レジスタへの書き込みを許可する */
546     SYSTEM.PRCR.WORD = 0xA508;          /* LVD関連レジスタへの書き込み許可 */
547
548     /* vdet1,vdet2を無効にする */
549     SYSTEM.LVCMPCR.BYTE = 0x00;
550     /* vdet1を3.10Vに設定する */
551     /* LVDLVLRレジスタを変更するときは、LVCMPCR.LVD1EおよびLVCMPCR.LVD2Eを共に"0"（電圧検出1/2 回路無効）にしてから行うこと。 */
552     /* 電圧検出１回路と電圧検出２回路は、同じ検出電圧レベル設定で使用しないこと。 */
553     SYSTEM.LVDLVLR.BYTE = 0x07;         /* 電圧検出レベル = 3.10V（vdet2は4.15V） */
554
555     /* デジタルフィルタを無効にする */
556     SYSTEM.LVD1CR0.BIT.LVD1DFDIS = 1;   /* 電圧監視１回路 */
557 //  SYSTEM.LVD2CR0.BIT.LVD2DFDIS = 1;   /* 電圧監視２回路 */
558
559     /* vdet1割り込みを有効にする */
560     SYSTEM.LVD1CR0.BIT.LVD1RI = 0;
561     /* vdet2割り込みを有効にする */
562 //  SYSTEM.LVD2CR0.BIT.LVD2RI = 0;
563
564     /* VCCがトリガ値を下回るとトリガするようにvdet1を設定する */
565     SYSTEM.LVD1CR1.BIT.LVD1IDTSEL = 1;  /* 電圧監視１回路：VCC＜Vdet1（下降）検出時（初期状態） */
566     /* VCCがトリガ値を下回るとトリガするようにvdet2を設定する */
567 //  SYSTEM.LVD2CR1.BIT.LVD2IDTSEL = 1;  /* 電圧監視２回路：VCC＜Vdet2（下降）検出時（初期状態） */
568
569     /* 出力比較を有効にする */
570     SYSTEM.LVD1CR0.BIT.LVD1CMPE = 1;    /* 電圧監視１回路 */
571 //  SYSTEM.LVD2CR0.BIT.LVD2CMPE = 1;    /* 電圧監視２回路 */
572
573     /* vdet1フラグをクリアする */
574     SYSTEM.LVD1SR.BIT.LVD1DET = 0;
575     /* vdet2フラグをクリアする */
576 //  SYSTEM.LVD2SR.BIT.LVD2DET = 0;
577
578     /* vdet1フラグがクリアされていることを確認する */
579     while(SYSTEM.LVD1SR.BIT.LVD1DET){
580     }
581
582     /* vdet1割り込みを許可にする */
583     SYSTEM.LVD1CR0.BIT.LVD1RIE = 1;
584     /* vdet2割り込みを許可にする */
585 //  SYSTEM.LVD2CR0.BIT.LVD2RIE = 1;
586
587     /* vdet1を有効にする */
588     SYSTEM.LVCMPCR.BIT.LVD1E = 1;
589     /* vdet2を有効にする */
590 //  SYSTEM.LVCMPCR.BIT.LVD2E = 1;
591
592     /* vdet1起動の遅延 */
593     while (delay--){                    /* 遅延が経過するまで何もしない */
594     }
595
596     /* vdet1 NMI割り込みを有効にする */
597     ICU.NMIER.BIT.LVD1EN = 1;
598     /* vdet2 NMI割り込みを有効にする */
599 //  ICU.NMIER.BIT.LVD2EN = 1;
600
601     /* LVD関連レジスタへの書き込みを禁止する */
602     SYSTEM.PRCR.WORD = 0xA500;          /* クロック発生回路関連レジスタ、      */
603                                         /* 動作モード、消費電力低減機能、      */
604                                         /* ソフトウェアリセット関連レジスタ、  */
```

```
605                                     /* LVD関連レジスタへの書き込み禁止     */
606 }
607
608 /************************************************************************/
609 /*                          NMI割り込みハンドラ                         */
610 /************************************************************************/
611 /* VCCがトリガ電圧より低かった(VCC＜Vdet1)場合、VCCがトリガ電圧より高い(VCC≧Vdet1)ときにトリガするように再設定する。 */
612 /* VCCがトリガ電圧より高かった(VCC≧Vdet1)場合、VCCがトリガ電圧より低い(VCC＜Vdet1)ときにトリガするように再設定する。 */
613 /* g_vcc_above_vdetフラグも更新する。 ←未使用                          */
614
615 void Int_NonMaskable(void)
616 {
617     /* LVD関連レジスタへの書き込みを許可する */
618     SYSTEM.PRCR.WORD = 0xA508;          /* LVD関連レジスタへの書き込み許可 */
619
620     /* vdet1フラグをクリアする */
621     SYSTEM.LVD1SR.BIT.LVD1DET = 0;      /* 電圧監視１回路 */
622
623     /* NMI割り込みフラグをクリアする */
624     ICU.NMICLR.BIT.LVD1CLR = 1;         /* NMISR.LVD1STフラグをクリア     */
625
626     /* VCCがトリガ電圧より低かった(VCC＜Vdet1)場合 */
627     if(0 == SYSTEM.LVD1SR.BIT.LVD1MON){ /* 電圧監視１回路/コンパレータA1ステータスレジスタ（LVD1SR） */
628         /* VCCがトリガ電圧より高い(VCC≧Vdet1)ときにトリガするように設定する */
629         SYSTEM.LVD1CR1.BIT.LVD1IDTSEL = 0;  /* 電圧監視１回路：VCC≧Vdet1（上昇）検出時 */
630         /* グローバルvdetステータスフラグを設定する */
631 //      g_vcc_above_vdet = false;
632     }
633     /* VCCがトリガ電圧より高かった(VCC≧Vdet1)場合 */
634     else{
635         /* VCCがトリガ電圧より低い(VCC＜Vdet1)ときにトリガするように設定する */
636         SYSTEM.LVD1CR1.BIT.LVD1IDTSEL = 1;  /* 電圧監視１回路：1:VCC＜Vdet1（下降）検出時（初期状態） */
637         /* グローバルvdetステータスフラグを設定する */
638 //      g_vcc_above_vdet = true;
639     }
640
641     /* LVD関連レジスタへの書き込みを禁止する */
642     SYSTEM.PRCR.WORD = 0xA500;          /* クロック発生回路関連レジスタ、      */
643                                         /* 動作モード、消費電力低減機能、      */
644                                         /* ソフトウェアリセット関連レジスタ、  */
645                                         /* LVD関連レジスタへの書き込み禁止     */
646 }
647
648 /************************************************************************/
649 /*                        ＭＰＵおよび周辺I/Oの設定                     */
650 /************************************************************************/
651
652 void HardwareSetup(void)
653 {
654     /* 全てモジュールストップ状態にする */
655     Stop_AllModule();
656
657     /* クロック初期設定 */
658     Init_SystemClock();
659
660     /* 全てのI/Oポートを汎用入出力ポートに初期化 */
661     Init_PortModeReg();
662
663     /* 存在しないI/Oポートの初期化 */
664     Init_NonExistPort();                /* #### PC1,PC0は出力ポートになる #### */
665
666     /* I/Oポートの初期化 */
667     /* I/Oポート１（PORT1.P17～P14を出力ポートにする例） */
668     // PORT1.ODR0.BYTE = 0x00;          /* オープンドレイン制御レジスタ0（初期状態：P13～P10 CMOS出力） */
669     PORT1.ODR1.BYTE = 0x00;             /* オープンドレイン制御レジスタ1（初期状態：P17～P14 CMOS出力） */
670     PORT1.PODR.BYTE = 0x00;             /* ポート出力データレジスタ（出力データの初期状態） */
671     PORT1.PDR.BYTE |= 0xF0;             /* ポート方向レジスタ（出力にするポート追加） */
672     // PORT1.PCR.BYTE = 0x00;           /* プルアップ制御レジスタ（初期状態：プルアップなし） */
673     PORT1.DSCR.BYTE = 0x00;             /* 駆動能力制御レジスタ（初期状態：通常出力） */
674
675     /* ～他にも初期化するものがあれば記述～ */
676
677 }
678
679 /* End of File */
```