マルチコア用プロセスペアフレームワーク
ソースコード

2010年02月

目 次

第 1 章　フレームワーク用ソースファイル　2

第 2 章　ＡＰＰ用ソースファイル　16

ＦＷのフォルダ構成は次のようになっています

▽ **Framework**

▽ FW_for_sh7205_CPU0

▷ src

▽ FW_for_sh7205_CPU1

▷ src

各 src フォルダには下記のファイル入っています（両フォルダ共通）

- FW_main.c(&h) ：ＦＷのメイン処理
- FW_recover.c(&h) ：システム回復処理
- FW_wdt.c(&h) ：ウォッチドッグタイマの処理
- FW_config.h ：ＦＷ設定ファイル
- FW_vector.c(&h) ：割り込みベクタテーブル
- AP_SetUp.c(&h) ：ＣＰＵの初期化とＡＰＰ依存の設定を行う
- AP_main.c(&h) ：ＡＰＰのメイン処理
- ckp_table.h ：チェックポイント関数テーブル
- IH_Template ：割り込みハンドラ用テンプレートファイル
- make_ckp_table.rb ：自動化スクリプト
  （以下は開発チップに依存するファイル）
- 7205.h ：SH7205 内部レジスタ定義ファイル
- BoradDepend.h ：ボード依存定義ファイル

次ページから詳細

第1章　フレームワーク用ソースファイル

FW main.h

```
1   /************************************************************************/
2   /*
3   /*              FW_mainのヘッダファイル
4   /*
5   /************************************************************************/
6
7   #ifndef         _FW_MAIN_H_
8   #define         _FW_MAIN_H_
9
10  #include        "FW_config.h"
11  #include        "FW_recover.h"
12  #include        "FW_wdt.h"
13  #include        "AP_SetUp.h"
14  #include        "AP_main.h"
15
16  void PowerON_Reset(void);
17  void main(void);
18  void cpu_seq_sw(void);
19
20  #endif  /* _FW_MAIN_H_ */
```

FW main.c

```
1  /************************************************************************/
2  /*
3  /*              フレームワークコンポーネント@@フローズンスポット@@
4  /*              名前 :: FW_main.c
5  /*              概略 :: ボードのスタートアップルーチン呼び出し後、
6  /*                              FWの機能の設定を行う
7  /*
8  /************************************************************************/
9
10 /* FW用必須ヘッダファイル */
11 #include "FW_main.h"
12 int cpu_seq = MAIN_CPU; /*CPU#0用*/
13 /*----------------------------------------------------------*/
14
15 /* 開発環境に依存するヘッダファイル */
16 #include        <machine.h>     /* 組み込み関数用ヘッダファイル */
17 #include        "7205.h"        /* SH7205内部レジスタ定義ファイル */
18 /*----------------------------------------------------------*/
19
20 #pragma section START
21 /*************************************/
22 /*                      CPU初期化                                   */
23 /*************************************/
24 void PowerON_Reset(void)
25 {
26         /* スタートアップルーチン呼び出し */
27         AP_SetUp_board();
28         main(); /*FWメイン処理*/
29         while(1){;}
30 }
31
32 void main(void){
33         if(cpu_seq == SUB_CPU){
34                 /* サブCPU :: 待機状態へ移行 */
35                 FW_recover_core_standby();
36                 FW_wdt_init();
37                 /* 周辺モジュール・割込みコントローラの再設定 */
```

```
38                  AP_SetUp_IH();
39                  /* ロールバック回復 */
40                  FW_recover_rollback();
41          }
42          else{
43                  FW_wdt_init();
44                  /* 周辺モジュール・割込みコントローラの設定 */
45                  AP_SetUp_IH();
46          }
47
48  #ifdef _AUTO_CKP_OFF
49          AP_main();      /*APP メイン処理*/
50  #else
51          while(1){
52                  /* 割込み発生まで待機 */
53                  sleep();        /* HEW の組み込み関数 */
54                  /* 全ての割込みが処理されたらチェックポインティング */
55                  FW_recover_checkpoint();
56          }
57  #endif  /*_AUTO_CKP_OFF*/
58  }
59
60  /*      cpu_seq main > sub */
61  void cpu_seq_sw(){
62          cpu_seq = SUB_CPU;
63  }
```

FW recover.h

```
1  /*****************************************************************************/
2  /*
3  /*              FW_recoverのヘッダファイル
4  /*
5  /*****************************************************************************/
6
7  #ifndef         _FW_RECOVER_H_
8  #define         _FW_RECOVER_H_
9
10 void FW_recover_checkpoint(void);
11 void FW_recover_rollback(void);
12 void FW_recover_core_standby(void);
13 void FW_recover_core_switch(void);
14
15 #endif  /* _FW_RECOVER_H_ */
```

FW recover.c

```
1   /*************************************************************************/
2   /*
3   /*              フレームワークコンポーネント@@フローズンスポット@@
4   /*              名前 :: FW_recover.c
5   /*              概略 :: プロセスペア (回復機能) に関する処理を集めた
 クラス                  */
6   /*
7   /*************************************************************************/
8
9   /* FW用必須記述 */
10  #include        "FW_recover.h"
11  #include        "FW_config.h"
12  #include        "ckp_table.h"
13  /*-----------------------------------------------------------*/
14
15  /* ボード及び開発環境に依存するヘッダファイル */
16  #include        <machine.h>     /* 組み込み関数用ヘッダファイル */
17  #include        "7205.h"        /* SH7205内部レジスタ定義ファイル */
18  /*-----------------------------------------------------------*/
19
20  /* チェックポイント関数ポインタ数の計算 */
21  int array_size = sizeof ckptable / sizeof ckptable[0] ;
22
23  /*************************************************************/
24  /* チェックポイント情報退避処理
25  /*************************************************************/
26  void FW_recover_checkpoint(){
27          unsigned int add = CKP_MEM;
28          int mask, i;
29          mask = get_imask();
30          set_imask(15);
31          for(i=0;i < array_size;i++){
32                  add = (*ckptable[i])(add,CKP_SAVE);
33          }
34          set_imask(mask);
35  }
36
```

```
37  /*****************************************************************/
38  /*      チェックポイント情報復帰処理                             */
39  /*****************************************************************/
40  void FW_recover_rollback(){
41          unsigned int add = CKP_MEM;
42          int mask, i;
43          mask = get_imask();
44          set_imask(15);
45          for(i=0;i < array_size;i++){
46                  add = (*ckptable[i])(add,CKP_BACK);
47          }
48          set_imask(mask);
49  }
50
51  /*****************************************************************/
52  /*      サブCPUスタンバイ操作                                    */
53  /*****************************************************************/
54  void FW_recover_core_standby(){
55          set_imask(14);
56          INTC.C0IPER.WORD |= 0xc000;     /* CPU#0へのプロセッサ間割
    込み15有効 */
57          /* スリープモードに移行(省電力状態) */
58          /* CPU#1の異常通知がくるまで待機 */
59          sleep();        /* 組み込み関数 */
60  }
61
62  void FW_recover_core_switch(){
63          INTC.C1IPCR15.WORD |= 0x0001;   //サブCPUへメインCPU異常
    通知
64  #ifdef _FW_M_SW_OFF
65          set_imask(15);
66          sleep();
67  #else
68          AP_SW_reset();/*メイン系→サブ系*/
69  #endif  /*_FW_M_SW_OFF*/
70  }
```

FW wdt.h

```
1  /************************************************************************/
2  /*
3  /*              FW_wdtのヘッダファイル
4  /*
5  /************************************************************************/
6  #ifndef _FW_WDT_H_
7  #define _FW_WDT_H_
8
9  #include        "FW_recover.h"
10
11
12 void FW_wdt_init(void);
13 void FW_wdt_context_save(void);
14 void FW_wdt_context_back(void);
15
16 void FW_wdt_start();
17 void FW_wdt_stop();
18 void FW_wdt_reset();
19 void FW_wdt_cntset(int);
20
21 void FW_wdt_overflow();
22
23 #endif  /* _FW_WDT_H_ */
24
```

FW wdt.c

```
1   /*****************************************************************************/
2   /*
3   /*              フレームワークコンポーネント@@フローズンスポット@@
4   /*              名前 :: FW_wdt.c
5   /*              概略 :: WDTに対する操作を集めたファイル
6   /*
7   /*****************************************************************************/
8
9   /* FWの必須ヘッダファイル */
10  #include        "FW_wdt.h"
11  #include        "FW_config.h"
12  /*-----------------------------------------------------------*/
13
14  /* ボード及び開発環境に依存するヘッダファイル */
15  #include        <machine.h>     /* 組み込み関数用ヘッダファイル */
16  #include        "7205.h"        /* SH7205内部レジスタ定義ファイル */
17  /*-----------------------------------------------------------*/
18
19  /* WDTカウント値の退避用変数 */
20  typedef struct{
21          short int state;
22          int count;
23  }WDT_CONTEXT;
24
25  WDT_CONTEXT wdt_context[MAX_CONTEXT];
26  static short int context_no = 0; /*退避WDTコンテキスト数*/
27
28  /* WDT初期化処理 */
29  void FW_wdt_init(void){
30          context_no = 0;
31          WDT.WTCSR0.WORD = 0xA500;               /* ウォッチドッグタ
  イマ停止 */
32          WDT.WTCSR0.WORD = 0xA500;               /* カウントクロック
  設定(001 : 0.984msec) */
33          WDT.WTCNT0.WORD = 0x5A00;               /* タイマ初期値設定 */
34          INTC.IDCNT26.WORD = 0x4100;             /* 41(43) : CPU0(1) 割
  込み許可 */
```

```
35          INTC.COIPR11.WORD |= 0xF000;    /* WDT0 IT 割り込みレベル15 */
36  }
37
38  /* WDT コンテキスト退避処理 */
39  void FW_wdt_context_save(void){
40          int mask;
41          mask = get_imask();
42          set_imask(15);
43          if(context_no == 0){/* 単発割込みの場合 */
44                   context_no++;
45          }
46          else{/* 多重割り込みの場合 */
47                  wdt_context[context_no].state = 1;
48                  wdt_context[context_no].count = WDT.WTCNT0.WORD;
49                  context_no++;
50          }
51          set_imask(mask);
52  }
53
54  /* WDT コンテキスト復帰処理 */
55  void FW_wdt_context_back(void){
56          int mask;
57          mask = get_imask();
58          set_imask(15);
59
60          if(context_no == 1){
61                  context_no = 0;
62          }
63          else if(context_no > 1){
64                  context_no--;
65                  WDT.WTCNT0.WORD = 0x5A00 | wdt_context[context_no].count;
  カウント値復帰 */
66          }
67          else{;}
68          set_imask(mask);
69  }
70
71  /* WDT 開始 */
72  void FW_wdt_start(){
```

```
73          FW_wdt_context_save();
74          WDT.WTCNT0.WORD = 0x5A00;       /* カウント初期化 */
75          WDT.WTCSR0.WORD =  (WDT.WTCSR0.WORD | 0xA520);  /* WDT稼働 */
76  }
77
78  /* WDT停止 */
79  void FW_wdt_stop(){
80          FW_wdt_context_back();
81          if(context_no == 0){/* 退避コンテキストが無ければ */
82                  WDT.WTCSR0.WORD = (WDT.WTCSR0.WORD & 0x00DF) | 0xA500;  /* WDT
    停止 */
83          }
84  }
85
86  /* WDTカウント値リセット */
87  void FW_wdt_reset(){
88          WDT.WTCNT0.WORD = 0x5A00;               /* カウント初期化 */
89  }
90
91  /* WDTカウント値設定 */
92  void FW_wdt_cntset(cnt){
93          /* 残りカウント値を255 − cntに設定 */
94          WDT.WTCNT0.WORD = 0x5A00 | cnt;
95  }
96
97
98  /* WDTオーバーフロー処理 */
99  void FW_wdt_overflow(){
100 #ifndef _FW_WDT_OFF
101         volatile unsigned char c;       /* フラグ初期化用*/
102         WDT.WTCSR0.WORD = 0xA500;       /* WDT停止 */
103         WDT.WTCNT0.WORD = 0x5A00;       /* カウント初期化 */
104         c = WDT.WTCSR0.BIT.IOVF;        /* IOVFクリアのための読み
    込み*/
105         WDT.WTCSR0.WORD = 0xA500;       /* IOVFクリア */
106         FW_recover_core_switch();       /* コア切替処理 */
107 #else
108         AP_wdt_int();
109 #endif /*_FW_WDT_OFF*/
```

```
110     }
111
```

FW config.h

```
1  /*****************************************************************************/
2  /*
3  /*              フレームワークの設定ファイル
4  /*
5  /*****************************************************************************/
6
7  #ifndef _FW_CONST_H_
8  #define _FW_CONST_H_
9
10 #define MAIN_CPU        1/*MAIN_CPU_ID*/
11 #define SUB_CPU         2/*SUB_CPU_ID*/
12
13 /*************************************************************/
14 /*      次の場合は"_AUTO_CKP_OFF"のコメント指定を除去する                    */
15 /*      1)      ユーザがメイン処理を記述する場合
16 /*      2)      APPでチェックポイントを残す場合
17 /*************************************************************/
18 //#define       _AUTO_CKP_OFF
19
20 //#define       _FW_WDT_OFF
21 //#define       _FW_M_SW_OFF
22
23 /*************************************************************/
24 /*      チェックポイント退避領域サイズの指定                                  */
25 /*      バイト単位。必ず4の倍数で指定する       0〜32000[byte]          */
26 /*************************************************************/
27 //#define       CKP_MEM_SIZE 32000
28 #define MAX_CONTEXT     10      /* 最大多重割り込み数 */
29
30 #ifdef  CKP_MEM_SIZE
31 #define CKP_MEM 0xFFF90000 - CKP_MEM_SIZE/*ckp退避領域*/
32 #else
33 #define CKP_MEM 0xFFF88000      /*ckp退避領域*/
34 #endif
35
36 #define CKP_SAVE        1
37 #define CKP_BACK        2
```

```
38
39  #endif  /* _FW_CONST_H_ */
```

ckp table.h

```
1   #ifndef _CKP_TABLE_H_
2   #define _CKP_TABLE_H_
3
4
5   unsigned int dummy_ckp(unsigned int,int);
6   /*       declarative checkpoint func    */
7   /* ここでチェックポイント関数を宣言する */
8   /**** 例 *****/
9   /* unsigned int samp_ckp(unsigned int,int); */
10  /*************/
11
12  /*      register checkpoint func        */
13  unsigned int (*ckptable[])(unsigned int,int)={dummy_ckp,
14          /*宣言した関数を登録する*/
15          /**** 例 *****/
16          /* samp_ckp,*/
17          /*************/
18
19  };
20
21  unsigned int dummy_ckp(add,mode){return add;}
22  #endif  /*_CKP_TABLE_H_*/
```

第2章　ＡＰＰ用ソースファイル

AP main.c

```
1   /************************************************************************/
2   /*
3   /*              ユーザ開発コンポーネント@@ホットスポット@@
4   /*              名前 :: AP_main.c
5   /*              概略 :: APPのメイン処理
6   /*
7   /************************************************************************/
8
9   /* FW用必須記述 */
10  #include        "AP_main.h"
11  /*--------------------------------------------------------*/
12  /* 以下開発環境依存 */
13  #include        <machine.h>     /* HEW組み込み関数用ヘッダファイル */
14  #include        "7205.h"        /* SH7205内部レジスタ定義ファイル */
15  #include        "BordDepend.h"/* ボード依存ヘッダファイル */
16
17  /* APP用 */
18
19  /**********************************************************/
20  /*      APPのメイン関数を記述
21  /*      "FW_config.h"の設定を変更することで、AP_SetUp_IH()          */
22  /*      終了後、呼び出されます
23  /**********************************************************/
24  void AP_main(){
25          ;
26  }
27
```

AP SetUp.h

```
1  /*****************************************************************************/
2  /*
3  /*              AP_SetUpのヘッダファイル
4  /*
5  /*****************************************************************************/
6  #ifndef         _AP_SETUP_H_
7  #define         _AP_SETUP_H_
8
9  void AP_SetUp_board(void);
10 void AP_SetUp_IH(void);
11 void AP_SW_reset(void);
12
13 #endif  /*_AP_SETUP_H_*/
```

AP SetUp.c

```
1   /************************************************************************/
2   /*
3   /*              ユーザ開発コンポーネント@@ホットスポット@@
4   /*              名前 :: AP_SetUp.c
5   /*              概略 :: ボードのセットアップルーチン、および APP で使
 用する                          */
6   /*                              周辺モジュールや割込みコントローラの
 設定を行う                      */
7   /*
8   /*
9   /************************************************************************/
10
11  /* FW 用必須記述 */
12  #include        "AP_SetUp.h"
13  /*--------------------------------------------------------*/
14  /*      以下開発環境依存        */
15  #include        <machine.h>     /* 組み込み関数用ヘッダファイル */
16  #include        "7205.h"        /* SH7205 内部レジスタ定義ファイル */
17  #include        "BordDepend.h"/* ボード依存ヘッダファイル */
18
19  /* 以下 SH7205 向けサンプル */
20  /*========================================================*/
21  /* 浮動小数点ステータス・コントロールレジスタ初期値 */
22  #define FPSCR_INIT      0x00040001
23  /* ステータスレジスタ初期値 */
24  #define SR_INIT         0x000000F0
25  /* CPU1 ベクタベースアドレス */
26  #define CPU1_VECTOR_BASE        0x00800000
27  static void set_stack_cpu(unsigned long stack);
28  void (*cpu1_boot) (void);
29  void sectionInit(void);
30  void cache_init(void);
31  /*========================================================*/
32
33  /**********************************************************/
34  /*      ボードに必要なスタートアップルーチンを記述してください      */
35  /*      ※ CPU#0 側
```

```
36  /************************************************************/
37  void AP_SetUp_board(){
38
39          /* サンプルコード記載 /*
40
41  }
42
43  /************************************************************/
44  /*      アプリケーションプログラムで使用する周辺モジュールや       */
45  /*      割込みコントローラの設定を行ってください                   */
46  /*  ※周辺モジュールからの割込みは CPU#0 で受けつけるように        */
47  /*      割込みコントローラで設定してください                       *
48  /************************************************************/
49  void AP_SetUp_IH(){
50          ;
51  }
52  /*      SW リセット処理 (サンプル)                  */
53  /*      AP_SetUp_board() の内容により異なる        */
54  void AP_SW_reset(){
55
56          /* サンプルコード記載 */
57
58          /*下記は必須*/
59          cpu_seq_sw();    /*系統の切替メイン→サブ*/
60          main();
61  }
```

AP main.h

```
1  /*****************************************************************************/
2  /*
3  /*              AP_mainのヘッダファイル
4  /*
5  /*****************************************************************************/
6  #ifndef         _AP_MAIN_H_
7  #define         _AP_MAIN_H_
8
9  /* ユーザプログラムからチェックポイントを残す場合は、 */
10 /* "FW_config.h"を修正し、AP_mainを定義してください */
11 void AP_main(void);
12
13 #endif  /*_AP_MAIN_H_*/
```

AP main.c

```
1  /************************************************************************/
2  /*
3  /*              ユーザ開発コンポーネント@@ホットスポット@@
4  /*              名前 :: AP_main.c
5  /*              概略 :: APPのメイン処理
6  /*
7  /************************************************************************/
8
9  /* FW用必須記述 */
10 #include        "AP_main.h"
11 /*--------------------------------------------------------*/
12 /* 以下開発環境依存 */
13 #include        <machine.h>     /* HEW組み込み関数用ヘッダファイル */
14 #include        "7205.h"        /* SH7205内部レジスタ定義ファイル */
15 #include        "BordDepend.h"/* ボード依存ヘッダファイル */
16
17 /* APP用 */
18
19 /**********************************************************/
20 /*      APPのメイン関数を記述
21 /*      "FW_config.h"の設定を変更することで、AP_SetUp_IH()          */
22 /*      終了後、呼び出されます
23 /**********************************************************/
24 void AP_main(){
25         ;
26 }
27
```

IH Template.h

```
1   /************************************************************************/
2   /*
3   /*              ハンドラ記述用テンプレートファイル@@IH_Template.c@@
4   /*              名前 :: IH_Template.c
5   /*              概略 :: 本ファイルコピーして各割り込みハンドラを作成
 してください      */
6   /*                              関数名を変更して使用してください。ベ
 クタテーブルには、              */
7   /*                              (1) の関数名を変更し、登録してくださ
 い                                  */
8   /*                              また (2) の関数名をユニークな関数名に
 変更し、"ckp_table.h"    */
9   /*                              に登録してください
10  /************************************************************************/
11
12  #include        "FW_config.h"
13
14  /*************************************************************/
15  /*      チェックポイント情報の登録
16  /*      チェックポイントとして残したい変数は次の構造体の                  */
17  /*      メンバとして宣言し、使用してください。                                *
18  /*      型名 (ckp_info) と構造体名 (ckp) は変更しないこと                  */
19  /*************************************************************/
20  typedef struct{
21  /*      サンプル    */
22          int dummy;
23  }ckp_info;
24  /*初期化*/
25  static ckp_info ckp = {0};
26
27  /*      (1)
28  /*************************************************************/
29  /*      割り込みハンドラの定義
30  /*      同じファイル内で複数のハンドラ定義可                                *
31  /*************************************************************/
32  #pragma interrupt handler_name(resbank)/*(1)<handler_name>"を変更
 し"FW_vector.c"に登録する*/
```

```
33  void handler_name(void)
34  {
35          FW_wdt_start(); /* この行は変更しないでください */
36                  ;
37          /* ハンドラ処理 */
38                  ;
39          FW_wdt_stop();  /*      この行は変更しないでください  */
40  }
41
42
43  /*      (2)
44  /************************************************************/
45  /*      チェックポイント情報の退避･復帰関数                                    */
46  /*      任意の関数名に変更し、"ckp_table.h"に登録してください       */
47  /*      関数名以外は変更を加えないでください                                   *
48  /************************************************************/
49  unsigned long ckp_func(add,mode){       //@@fname@@
50          /*      これより以下は変更不可     */
51          if(mode == CKP_SAVE){
52                  *((ckp_info*)add) = ckp;
53          }
54          else if(mode == CKP_BACK){
55                  ckp = *((ckp_info*)add);
56          }
57          else{}
58          return add + sizeof ckp ;
59  }
```