```
/*
 *  TOPPERS/JSP Kernel
 *      Toyohashi Open Platform for Embedded Real-Time Systems/
 *      Just Standard Profile Kernel
 *
 *  Copyright (C) 2000 by Embedded and Real-Time Systems Laboratory
 *                              Toyohashi Univ. of Technology, JAPAN
 *
 *  上記著作権者は，以下の条件を満たす場合に限り，本ソフトウェア（本ソ
 *  フトウェアを改変したものを含む．以下同じ）を使用・複製・改変・再配
 *  布（以下，利用と呼ぶ）することを無償で許諾する．
 *  (1) 本ソフトウェアをソースコードの形で利用する場合には，上記の著作
 *      権表示，この利用条件および下記の無保証規定が，そのままの形でソー
 *      スコード中に含まれていること．
 *  (2) 本ソフトウェアをバイナリコードの形または機器に組み込んだ形で利
 *      用する場合には，次のいずれかの条件を満たすこと．
 *    (a) 利用に伴うドキュメント（利用者マニュアルなど）に，上記の著作
 *        権表示，この利用条件および下記の無保証規定を掲載すること．
 *    (b) 利用の形態を，別に定める方法によって，上記著作権者に報告する
 *        こと．
 *  (3) 本ソフトウェアの利用により直接的または間接的に生じるいかなる損
 *      害からも，上記著作権者を免責すること．
 *
 *  本ソフトウェアは，無保証で提供されているものである．上記著作権者は，
 *  本ソフトウェアに関して，その適用可能性も含めて，いかなる保証も行わ
 *  ない．また，本ソフトウェアの利用により直接的または間接的に生じたい
 *  かなる損害に関しても，その責任を負わない．
 *
 *  @(#) $Id: wait.h,v 1.1 2000/11/14 14:44:21 hiro Exp $
 *
 *
 *  リアルタイムカーネル勉強会用にコメントを追加した
 *  Mon Sep 03 00:15:39 2001 modified by Keizyu Anada YAMAHA CORPORATION
 *
 *
 */

/*
 *  待ち状態管理モジュール
 */

#ifndef _WAIT_H_
#define _WAIT_H_

#include "task.h"
#include "time_event.h"



/*
 *  待ち状態への移行
 *
 *  実行中のタスクを待ち状態に移行させる．具体的には，実行中のタスクを
 *  レディキューから削除し，TCB の winfoフィールド，WINFO の tmevtbフ
 *  ィールドを設定する．
 *
 *  この関数は slp_tsk から呼ばれる可能性がある。
 *
 */
Inline void make_wait(WINFO *winfo)
{
  /*
     このタスクをレディキューから削除する。
     このタスクと同じ優先度のタスクがなければ、
     レディキューサーチマップからこの優先度ビットをクリアし
     このタスクが最高優先順位タスクならば最高優先順位のタスクを更新する。
     このタスクと同じ優先度のタスクがあれば、
     次の優先順位のタスクを最高優先順位のタスクに設定する。
   */
  make_non_runnable(runtsk);

  runtsk->winfo = winfo;

  /* 永久待ちのなので NULL を設定する */
  winfo->tmevtb = NULL;
}

/*
 *  待ち状態への移行（タイムアウト指定）
 *
 *  実行中のタスクを，タイムアウト指定付きで待ち状態に移行させる．具体
 *  的には，実行中のタスクをレディキューから削除し，TCB の winfoフィー
 *  ルド，WINFO の tmevtbフィールドを設定する．また，タイムイベントブ
 *  ロックを登録する．
 */
extern void make_wait_tmout(WINFO *winfo, TMEVTB *tmevtb, TMO tmout);

/*
 *  待ち解除
 *
 *  tcb で指定されるタスクの待ち状態を解除する．具体的には，タイムイベ
 *  ントブロックが登録されていれば，それを登録解除する．また，タスク状
 *  態を更新し，待ち解除したタスクからの返値を E_OK とする．待ちキュー
 *  からの削除は行わない．待ち解除したタスクへのディスパッチが必要な場
 *  合には TRUE を返す．
 */
extern BOOL wait_complete(TCB *tcb);

/*
 *  タイムアウトに伴う待ち解除
 *
 *  tcb で指定されるタスクが，待ちキューにつながれていれば待ちキューか
 *  ら削除し，タスク状態を更新する．また，待ち解除したタスクからの返値
 *  を，wait_tmoutでは E_TMOUT，wait_tmout_ok では E_OK とする．待ち解
 *  除したタスクへのディスパッチが必要な時は，reqflg を TRUE にする．
 *  wait_tmout_ok は，dly_tsk で使うためのもので，待ちキューから削除す
 *  る処理を行わない．
 *  いずれの関数も，タイムイベントのコールバック関数として用いるための
 *  もので，割込みハンドラから呼び出されることを想定している．
 */
extern void wait_tmout(TCB *tcb);
extern void wait_tmout_ok(TCB *tcb);

/*
 *  待ち状態の強制解除
 *
 *  tcb で指定されるタスクの待ち状態を強制的に解除する．具体的には，タ
 *  スクが待ちキューにつながれていれば待ちキューから削除し，タイムイベ
 *  ントブロックが登録されていればそれを登録解除する．
 *  wait_cancel は，タスクの状態は更新しない．
 *  wait_release は，タスクの状態を更新し，待ち解除したタスクからの返
 *  値を E_RLWAI とする．また，待ち解除したタスクへのディスパッチが必
 *  要な場合には TRUE を返す．
 */
extern void wait_cancel(TCB *tcb);
extern BOOL wait_release(TCB *tcb);

/*
 *  同期・通信オブジェクトのコントロールブロックの共通部分操作ルーチン
 *
 *  同期・通信オブジェクトの初期化ブロックとコントロールブロックの先頭
 *  部分は共通になっている．以下は，その共通部分を扱うための型およびルー
```

```
 133:  *  チン群である.
 134:  *  複数の待ちキューを持つ同期・通信オブジェクトの場合，先頭以外の待ち
 135:  *  キューを操作する場合には，これらのルーチンは使えない．また，オブジェ
 136:  *  クト属性の TA_TPRI ビットを参照するので，このビットを他の目的に使っ
 137:  *  ている場合も，これらのルーチンは使えない.
 138:  */
 139:
 140: /*
 141:  *  同期・通信オブジェクトの初期化ブロックの共通部分
 142:  */
 143: typedef struct wait_object_initialization_block {
 144:   ATR wobjatr;  /* オブジェクト属性 */
 145: } WOBJINIB;
 146:
 147: /*
 148:  *  同期・通信オブジェクトのコントロールブロックの共通部分
 149:  */
 150: typedef struct wait_object_control_block {
 151:   QUEUE wait_queue;         /* 待ちキュー */
 152:   const WOBJINIB *wobjinib; /* 初期化ブロックへのポインタ */
 153: } WOBJCB;
 154:
 155: /*
 156:  *  同期・通信オブジェクト待ち情報ブロックの定義
 157:  */
 158: typedef struct wait_object_waiting_information {
 159:   WINFO winfo;     /* 標準の待ち情報ブロック */
 160:   WOBJCB *wobjcb;  /* 待ちオブジェクトのコントロールブロック */
 161: } WINFO_WOBJ;
 162:
 163: /*
 164:  *  同期・通信オブジェクトに対する待ち状態への移行
 165:  *
 166:  *  実行中のタスクを待ち状態に移行させ，同期・通信オブジェクトの待ちキュー
 167:  *  につなぐ．また，待ち情報ブロック（WINFO）の wobjcb を設定する.
 168:  *  wobj_make_wait_tmout は，タイムイベントブロックの登録も行う.
 169:  */
 170: extern void wobj_make_wait(WOBJCB *wobjcb, WINFO_WOBJ *winfo);
 171: extern void wobj_make_wait_tmout(WOBJCB *wobjcb, WINFO_WOBJ *winfo,
 172:                 TMEVTB *tmevtb, TMO tmout);
 173:
 174: /*
 175:  *  タスク優先度変更時の処理
 176:  *
 177:  *  同期・通信オブジェクトに対する待ち状態にあるタスクの優先度が変更さ
 178:  *  れた場合に，待ちキューの中でのタスクの位置を修正する.
 179:  */
 180: extern void wobj_change_priority(WOBJCB *wobjcb, TCB *tcb);
 181:
 182: #endif /* _WAIT_H_ */
 183:
```