```
/*****************************************************************************
       RTM_H8(for H8/3664F) C-Compiler Version

 [Note]
       June    27,1998       Kenji Arai
       January   1,1999      Change starting method for Cyclic task
       November  4,2001      Start porting to H8/3664F
       December 15,2001

       Copyright (C) 1998,'99,'00,2001 Kenji Arai/JH1PJL
       All rights reserved. Permission is granted to use, modify,or redistribute
    this software so long as it is not sold or exploited for profit.

       THIS SOFTWARE IS PROVIDED AS IS AND WITHOUT WARRANTY OF ANY KIND,
       EITHER EXPRESSED OR IMPLIED.
*****************************************************************************/
//            July    19,2004
//            July    30,2004       Add user timer

#include "rtm_H8_3664.h"

#define RTM_TM        4       /* 500uS*4 = 2mS */

#define RTM_ON        1
#define RTM_OFF               0
#define RTM_RUN               1
#define RTM_STOP      0

/*****  Extern function prototype ******/
extern void edtsk_dispatch();
extern void edtsk_end( void );
//extern void irq_enable( void );
//extern void irq_disable( void );

extern void ed0_main( void );
extern void ed1_main( void );
extern void ed2_main( void );
extern void ed3_main( void );
extern void cy0_main( void );
extern void cy1_main( void );
extern void cy2_main( void );
extern void cy3_main( void );
extern void cy4_main( void );
extern void cy5_main( void );
extern void cy6_main( void );
extern void cy7_main( void );

/*****  RAM assign  ******/
unsigned char ccr_save;             /* for RTM */
unsigned int
       tim_usr0,      /* user timer no.0 */
       tim_usr1,      /* user timer no.1 */
       tim_usr2,      /* user timer no.2 */
       tim_usr3;      /* user timer no.3 */
unsigned int
       tim_usr_0;     // user timer based on 500uS
```

```
static unsigned char
       tim_sys,      /* system timer */
       tim_cyc0,     /* cyclic task no.0 period timer */
       tim_cyc1,     /* cyclic task no.1 period timer */
       tim_cyc2,     /* cyclic task no.2 period timer */
       tim_cyc3,     /* cyclic task no.3 period timer */
       tim_cyc4,     /* cyclic task no.4 period timer */
       tim_cyc5,     /* cyclic task no.5 period timer */
       tim_cyc6,     /* cyclic task no.6 period timer */
       tim_cyc7;     /* cyclic task no.7 period timer */

struct {
       unsigned char ed0_req:      1; /* event driven task no.0 request flag */
       unsigned char ed1_req:      1; /* event driven task no.1 request flag */
       unsigned char ed2_req:      1; /* event driven task no.2 request flag */
       unsigned char ed3_req:      1; /* event driven task no.3 request flag */
       unsigned char not_used:     2;
       unsigned char edtsk_run:1; /* ed task running flag */
       unsigned char cytsk_run:1; /* cyclic task running flag */
}status;

static union {
       unsigned char request;      /* cyclic task request flag */
       struct {
              unsigned char cy0:   1;
              unsigned char cy1:   1;
              unsigned char cy2:   1;
              unsigned char cy3:   1;
              unsigned char cy4:   1;
              unsigned char cy5:   1;
              unsigned char cy6:   1;
              unsigned char cy7:   1;
       }req;
}r;

static union {
       unsigned char priorty;      /* cyclic task priority flag */
       struct {
              unsigned char cy0:   1;
              unsigned char cy1:   1;
              unsigned char cy2:   1;
              unsigned char cy3:   1;
              unsigned char cy4:   1;
              unsigned char cy5:   1;
              unsigned char cy6:   1;
              unsigned char cy7:   1;
       }prio;
}p;


/********  Task Branch Table **************/
void (*const tsk_st_addr[12])() = {
       ed0_main, ed1_main, ed2_main, ed3_main,
       cy0_main, cy1_main, cy2_main, cy3_main,
       cy4_main, cy5_main, cy6_main, cy7_main
};

/* External data */
```

```
extern const unsigned char cyclic_period[8];
extern const unsigned char cyclic_init_period[8];


/****************************** Basic routine
*********************************************************/
extern void rtm_core( void )
{
	if( p.priorty != 0 ){
	/* cyclic task dispatch */
		if( p.prio.cy7 ){
			goto diptch7;
		}else if( p.prio.cy6 ){
			goto diptch6;
		}else if( p.prio.cy5 ){
			goto diptch5;
		}else if( p.prio.cy4 ){
			goto diptch4;
		}else if( p.prio.cy3 ){
			goto diptch3;
		}else if( p.prio.cy2 ){
			goto diptch2;
		}else if( p.prio.cy1 ){
			goto diptch1;
		}else if( p.prio.cy0 ){
			goto diptch0;
		}
	}
	if( r.request != 0 ){
		/* cyclic task dispatch */
		if( r.req.cy0 ){
			/* cyclic task no.0 check and go */
diptch0:
			p.prio.cy0 = RTM_OFF;
			r.req.cy0 = RTM_OFF;
			status.cytsk_run = RTM_RUN;
			edtsk_dispatch( tsk_st_addr[4], edtsk_end );
			status.cytsk_run = RTM_OFF;
			return;
		}else if( r.req.cy1 ){
			/* cyclic task no.1 check and go */
diptch1:
			p.prio.cy1 = RTM_OFF;
			r.req.cy1 = RTM_OFF;
			status.cytsk_run = RTM_RUN;
			edtsk_dispatch( tsk_st_addr[5], edtsk_end );
			status.cytsk_run = RTM_OFF;
			return;
		}else if( r.req.cy2 ){
			/* cyclic task no.2 check and go */
diptch2:
			p.prio.cy2 = RTM_OFF;
			r.req.cy2 = RTM_OFF;
			status.cytsk_run = RTM_RUN;
			edtsk_dispatch( tsk_st_addr[6], edtsk_end );
			status.cytsk_run = RTM_OFF;
			return;
		}else if( r.req.cy3 ){
```

```
                        /* cyclic task no.3 check and go */
diptch3:
                        p.prio.cy3 = RTM_OFF;
                        r.req.cy3 = RTM_OFF;
                        status.cytsk_run = RTM_RUN;
                        edtsk_dispatch( tsk_st_addr[7], edtsk_end );
                        status.cytsk_run = RTM_OFF;
                        return;
                }else if( r.req.cy4 ){
                        /* cyclic task no.4 check and go */
diptch4:
                        p.prio.cy4 = RTM_OFF;
                        r.req.cy4 = RTM_OFF;
                        status.cytsk_run = RTM_RUN;
                        edtsk_dispatch( tsk_st_addr[8], edtsk_end );
                        status.cytsk_run = RTM_OFF;
                        return;
                }else if( r.req.cy5 ){
                        /* cyclic task no.5 check and go */
diptch5:
                        p.prio.cy5 = RTM_OFF;
                        r.req.cy5 = RTM_OFF;
                        status.cytsk_run = RTM_RUN;
                        edtsk_dispatch( tsk_st_addr[9], edtsk_end );
                        status.cytsk_run = RTM_OFF;
                        return;
                }else if( r.req.cy6 ){
                        /* cyclic task no.6 check and go */
diptch6:
                        p.prio.cy6 = RTM_OFF;
                        r.req.cy6 = RTM_OFF;
                        status.cytsk_run = RTM_RUN;
                        edtsk_dispatch( tsk_st_addr[10], edtsk_end );
                        status.cytsk_run = RTM_OFF;
                        return;
                }else if( r.req.cy7 ){
                        /* cyclic task no.7 check and go */
diptch7:
                        p.prio.cy7 = RTM_OFF;
                        r.req.cy7 = RTM_OFF;
                        status.cytsk_run = RTM_RUN;
                        edtsk_dispatch( tsk_st_addr[11], edtsk_end );
                        status.cytsk_run = RTM_OFF;
                        return;
                }
        }
}

/*********************************** Interrupt routine
************************************************
        Caution!!
        A part of the following routine has strong relationship with created code by the C compiler.
        Please check _edtsk_dispatch and _edtsk_end subroutines included in source file named
rtm_H8tiny.src.
*************************************************************************************************
/
extern void rtm_h8( void )
{
```

```
if( tim_usr_0 )
        --tim_usr_0;
if( !--tim_sys ){     /* 2mSec interval check */
        tim_sys = RTM_TM;
        if( !--tim_cyc0 ){
                tim_cyc0 = cyclic_period[0];
                if(r.req.cy0)
                        p.prio.cy0 = RTM_ON;
                else
                        r.req.cy0  = RTM_ON;
        }
        if( !--tim_cyc1 ){
                tim_cyc1 = cyclic_period[1];
                if(r.req.cy1)
                        p.prio.cy1 = RTM_ON;
                else
                        r.req.cy1  = RTM_ON;
        }
        if( !--tim_cyc2 ){
                tim_cyc2 = cyclic_period[2];
                if(r.req.cy2)
                        p.prio.cy2 = RTM_ON;
                else
                        r.req.cy2  = RTM_ON;
        }
        if( !--tim_cyc3 ){
                tim_cyc3 = cyclic_period[3];
                if(r.req.cy3)
                        p.prio.cy3 = RTM_ON;
                else
                        r.req.cy3 = RTM_ON;
        }
        if( !--tim_cyc4 ){
                tim_cyc4 = cyclic_period[4];
                if(r.req.cy4)
                        p.prio.cy4 = RTM_ON;
                else
                        r.req.cy4  = RTM_ON;
        }
        if( !--tim_cyc5 ){
                tim_cyc5 = cyclic_period[5];
                if(r.req.cy5)
                        p.prio.cy5 = RTM_ON;
                else
                        r.req.cy5  = RTM_ON;
        }
        if( !--tim_cyc6 ){
                tim_cyc6 = cyclic_period[6];
                if(r.req.cy6)
                        p.prio.cy6 = RTM_ON;
                else
                        r.req.cy6 =  RTM_ON;
        }
        if( !--tim_cyc7 ){
                tim_cyc7 = cyclic_period[7];
                if(r.req.cy7)
                        p.prio.cy7 = RTM_ON;
                else
```

```
                        r.req.cy7 =  RTM_ON;
                }
                if( tim_usr0 )
                        --tim_usr0;
                if( tim_usr1 )
                        --tim_usr1;
                if( tim_usr2 )
                        --tim_usr2;
                if( tim_usr3 )
                        --tim_usr3;
        }
        if( status.edtsk_run == RTM_STOP ){
                /* event driven task dispatch */
                if( status.ed0_req ){
                        /* event driven task no.0 check and go */
                        status.ed0_req = RTM_OFF;
                        status.edtsk_run = RTM_RUN;
                        edtsk_dispatch( tsk_st_addr[0], edtsk_end );
                        status.edtsk_run = RTM_STOP;
                }else if( status.ed1_req ){
                        /* event driven task no.1 check and go */
                        status.ed1_req = RTM_OFF;
                        status.edtsk_run = RTM_RUN;
                        edtsk_dispatch( tsk_st_addr[1], edtsk_end );
                        status.edtsk_run = RTM_STOP;
                }else if( status.ed2_req ){
                        /* event driven task no.2 check and go */
                        status.ed2_req = RTM_OFF;
                        status.edtsk_run = RTM_RUN;
                        edtsk_dispatch( tsk_st_addr[2], edtsk_end );
                        status.edtsk_run = RTM_STOP;
                }else if( status.ed3_req ){
                        /* event driven task no.3 check and go */
                        status.ed3_req = RTM_OFF;
                        status.edtsk_run = RTM_RUN;
                        edtsk_dispatch( tsk_st_addr[3], edtsk_end );
                        status.edtsk_run = RTM_STOP;
                }
        }
        if( status.cytsk_run == RTM_STOP ){
                rtm_core();
        }
}

/********************************** Event Driven Task request
****************************************/
extern void reqed0()
{
        status.ed0_req = RTM_ON;
}

extern void reqed1()
{
        status.ed1_req = RTM_ON;
}

extern void reqed2()
{
```

```
        status.ed2_req = RTM_ON;
}

extern void reqed3()
{
        status.ed3_req = RTM_ON;
}

/********************************** Initialize routine
***********************************************/
extern void rtm_h8_init()
{
        tim_sys = RTM_TM;     /* timer base period */
        tim_cyc0 = cyclic_init_period[0];
        tim_cyc1 = cyclic_init_period[1];
        tim_cyc2 = cyclic_init_period[2];
        tim_cyc3 = cyclic_init_period[3];
        tim_cyc4 = cyclic_init_period[4];
        tim_cyc5 = cyclic_init_period[5];
        tim_cyc6 = cyclic_init_period[6];
        tim_cyc7 = cyclic_init_period[7];

        status.ed0_req = RTM_OFF;
        status.ed1_req = RTM_OFF;
        status.ed2_req = RTM_OFF;
        status.ed3_req = RTM_OFF;

        r.request = 0;
        p.priorty = 0;
        status.edtsk_run = RTM_STOP;

        tim_usr0 = 0;         /* idle */
        tim_usr1 = 0;         /* idle */
        tim_usr2 = 0;         /* idle */
        tim_usr3 = 0;         /* idle */
}
```