```
$NOMOD51
;------------------------------------------------------------------------------
;  This file is part of the RTX-51 TINY  Real-Time Operating System Package
;  Copyright KEIL ELEKTRONIK GmbH and Keil Software, Inc. 1991-2002
;  Version 2.01
;------------------------------------------------------------------------------
;  CONF_TNY.A51:  This code allows the configuration of the
;               RTX-51 TINY Real-Time Operating System
;
;  Copy this file to your project folder and add the copy to your uVision2
;  project.  You can customize several parameters of RTX51 Tiny within this
;  configuration file.
;
;  If you use command line tools, translate this file with:
;
;    Ax51 CONF_TNY.A51
;
;  If you use command line tools, link the modified CONF_TNY.OBJ file to
;  your application with:
;
;    Lx51 <your object file list>, CONF_TNY.OBJ <controls>
;
;------------------------------------------------------------------------------
;
;  RTX-51 TINY Hardware-Timer
;  ==========================
;
;  With the following EQU statements the initialization of the RTX-51 TINY
;  Hardware-Timer can be defined (RTX-51 TINY uses the 8051 Timer 0 for
;  controlling RTX-51 software timers).
;
;  Define the register bank used for the timer interrupt.
INT_REGBANK EQU     1           ; default is Registerbank 1
;
;  Define Hardware-Timer tick time in 8051 machine cycles.
INT_CLOCK     EQU     500       ; default is 10000 cycles
;
;  Define Round-Robin Timeout in Hardware-Timer ticks.
TIMESHARING EQU     2           ; default is 5 Hardware-Timer ticks.
;                                 ; 0 disables Round-Robin Task Switching
;
;  Long User Interrupt Routines: set to 1 if your application contains
;  user interrupt functions that may take longer than a hardware timer
;  interval for execution.
LONG_USR_INTR         EQU     0           ; 0 user interrupts execute fast.
;                           ; 1 user interrupts take long execution times.
P3 EQU 0xB0
P0 EQU 0x80
P1 EQU 0x90
P2 EQU 0xa0
;
;------------------------------------------------------------------------------
;
;  USER CODE FOR 8051 HARDWARE TIMER INTERRUPT
;  ===========================================
;
;  The following macro defines the code executed on a hardware timer interrupt.
;
;  Define instructions executed on a hardware timer interrupt.
HW_TIMER_CODE       MACRO
          MOV P0,#0                         ; Empty Macro by default
          MOV P0,#0x80
          MOV P0,#0
```

```
                RETI
                ENDM
;
;
;-------------------------------------------------------------------------
;
;  CODE BANKING SUPPORT
;  ====================
;
;  The following EQU statement controls the code banking support for RTX51 TINY.
;
;  Enable or disable code banking support
CODE_BANKING    EQU     0       ; 0 (default) application uses no code banking
;                       ; 1 application uses code banking
;
;-------------------------------------------------------------------------
;
;  RTX-51 TINY Stack Space
;  =======================
;
;  The following EQU statements defines the size of the internal RAM used
;  for stack area and the minimum free space on the stack.  A macro defines
;  the code executed when there is there is not enough free stack on the
;  CPU stack.
;
;  Define the highest RAM address used for CPU stack
RAMTOP                  EQU     0FFH    ; default is address (256-1)
;
FREE_STACK      EQU     20          ; default is 20 bytes free space on stack
;                       ; the value 0 disables stack checking
;
STACK_ERRORMACRO
                CLR     EA          ; disable interrupts
                SJMP    $           ; endless loop if stack space is exhausted
                ENDM
;
;
;-------------------------------------------------------------------------
;
;  8051 CPU IDLE CODE
;  ==================
;
;  Many 8051 devices provide an IDLE MODE that reduces power consumption and
;  EMC.  The following macro defines the code executed when there is no
;  ready task in the system.  The code must set the CPU into an IDLE MODE
;  that stops instruction execution until an 8051 hardware interrupt occurs.
;

; Disable or Enable CPU_IDLE CODE
CPU_IDLE_CODE   EQU     1       ; 0  CPU_IDLE MACRO is not inserted
                        ; 1  CPU_IDLE MACRO is executed

PCON            DATA    087H    ; Power Control SFR on most 8051 devices

; Stop CPU execution until hardware interrupt; executed when there is no
; active task in the system.
CPU_IDLE        MACRO
                ORL     PCON,#1             ; set 8051 CPU to IDLE
                ENDM
;
;
;-------------------------------------------------------------------------
;---------------- !!! End of User Configuration Part    !!! ------------------
```

```
;---------------- !!! Do not modify code sections below !!! ------------------
;-------------------------------------------------------------------------------

; SFR Symbols
PSW     DATA    0D0H
ACC     DATA    0E0H
……,,,,,,,,,,,,,,,
EX0     BIT     0A8H

; Check Configuration Values


                NAME  ?RTX51_TINY_KERNAL

PUBLIC          ?RTX_CURRENTTASK
PUBLIC          ?RTX_RAMTOP
PUBLIC  os_switch_task
PUBLIC  ?RTX?SET_ISR

EXTRN NUMBER (?RTX_MAXTASKN)            ; max Task Number

?RTX_RAMTOP     EQU   RAMTOP
?RTX_CLOCK      EQU   -INT_CLOCK

?RTX_REGISTERBANK EQU     INT_REGBANK * 8
                DSEG  AT    ?RTX_REGISTERBANK
                DS      2     ; temporary space
?RTX_SAVEACC:     DS  1
saveacc         EQU   R2    ; for access in interrupt service routine
?RTX_SAVEPSW:     DS  1
savepsw         EQU   R3    ; for access in interrupt service routine
?RTX_CURRENTTASK: DS        1
currenttask     EQU         R4    ; for access in interrupt service routine

IF (TIMESHARING <> 0)
?RTX_ROBINTIME:   DS          1
robintime       EQU   R5    ; for access in interrupt service routine
ENDIF

IF (CODE_BANKING <> 0)
EXTRN DATA    (?B_CURRENTBANK)
EXTRN   CODE    (?B_RESTORE_BANK)
ENDIF


;-------------------------------------------------
; Table of Task Entry Pointers
;-------------------------------------------------
PUBLIC          ?RTX_TASKENTRY

?RTX?TASKENT?S  SEGMENT CODE
                RSEG    ?RTX?TASKENT?S
?RTX_TASKENTRY:     DS      2

;-------------------------------------------------
; Table of Stack Pointers for each task
;-------------------------------------------------
PUBLIC          ?RTX_TASKSP

?RTX?TASKSP?S   SEGMENT     IDATA
                RSEG    ?RTX?TASKSP?S
?RTX_TASKSP: DS     1
```

```
;------------------------------------------------
; Table of Task Timer/State Pointers
;------------------------------------------------
PUBLIC          ?RTX_TASKSTATUS

?RTX?TASKSTATE?S  SEGMENT IDATA
                  RSEG    ?RTX?TASKSTATE?S
?RTX_TASKSTATUS:
TimerVal:       DS      1       ; Task Timer (Software Timer for each task)
TaskState:      DS      1     ; Task Status (state of each Task)

; Definitions for Bits in Task State
;  TaskState.0  = Wait for Signal
;  TaskState.1  = Wait for TimeOut
;  TaskState.2  = Signal Flag
;  TaskState.3  = TimeOut Flag
;  TaskState.4  = Task Ready (Wait for Running)
;  TaskState.5  = Task Active (enabled with os_create)
;  TaskState.6  = Round Robin Time Out
;  TaskState.7  = Run Flag

; byte mask definitions
K_SIG       EQU       1
K_TMO       EQU       2
SIG_EVENT   EQU     4
TMO_EVENT   EQU     8
K_READY             EQU     16
K_ACTIVE    EQU     32
K_ROBIN             EQU     64
K_IVL       EQU   128  ; not a task state bit; only used in os_wait
RDY_EVENT   EQU   128  ; READY status flag
K_RDY       EQU   128  ; READY status flag

; bit position definitions
B_WAITSIG   EQU     0
B_WAITTIM   EQU     1
B_SIGNAL    EQU     2
B_TIMEOUT   EQU     3
B_READY             EQU     4
B_ACTIVE    EQU     5
B_ROBIN             EQU     6
B_IVL       EQU   7   ; not a task state bit; only used in os_wait
B_RDY       EQU   7


IF (TIMESHARING OR CPU_IDLE_CODE)
?RTX?BITS   SEGMENT     BIT
            RSEG    ?RTX?BITS
ENDIF

IF (TIMESHARING)
?RTX_TS_DELAY:      DBIT    1     ; Status bit set when task switch in progress
ENDIF

IF (CPU_IDLE_CODE)
?RTX_ISR_SIG: DBIT    1       ; Status bit set when interrupt or os_set_signal
ENDIF


            CSEG    AT      0BH
        JMP     TIMERINT

?RTX?CODE   SEGMENT CODE
```

```
            RSEG  ?RTX?CODE
                  USING  0                  ; Registerbank 0 for following code

IF (FREE_STACK <> 0)
?RTX_STACKERROR:
            STACK_ERROR           ; User defined Stack Error Code
ENDIF

HW_TIMER:     HW_TIMER_CODE

TIMERINT:

IF (LONG_USR_INTR)
                  PUSH    ACC
                  MOV     A,PSW
                  ANL     A,#018H
                  XRL     A,#?RTX_REGISTERBANK
                  JNZ     CONT_TIMINT
; avoid recursive timer interrupt
                  POP     ACC
                  RETI              ; Return from Recursive Timer Interrupt
CONT_TIMINT:    POP   ACC

ENDIF

                  CALL    HW_TIMER      ; Enable Interrupts again.

                  MOV     ?RTX_SAVEPSW,PSW
                  MOV     PSW,#?RTX_REGISTERBANK
                  MOV     saveacc,A
; Update 8051 Interrupt Timer
                  CLR     TR0
                  MOV     A,TL0
                  ADD     A,#LOW (?RTX_CLOCK + 7)
                  MOV     TL0,A
                  MOV     A,TH0
                  ADDC    A,#HIGH (?RTX_CLOCK + 7)
                  MOV     TH0,A
                  SETB    TR0

IF (FREE_STACK <> 0)
; Check if enough free stack is available
                  MOV     A,currenttask
                  ADD     A,#?RTX?TASKSP?S+1
                  MOV     R0,A
                  MOV     A,@R0
                  CJNE    currenttask,#?RTX_MAXTASKN,checkstack
                  MOV     A,#RAMTOP
checkstack:       CLR     C
                  SUBB    A,SP
                  CJNE    A,#FREE_STACK,$+3
                  JC      ?RTX_STACKERROR
ENDIF

; Update & Check Task Timers
                  MOV     R1,#?RTX_MAXTASKN+1
                  MOV     R0,#?RTX?TASKSTATE?S
TIMERLOOP:    DEC     @R0         ; Decrement timer
                  MOV     A,@R0
                  INC     R0          ; advance to TaskState
                  JNZ     NoTimeout
                  CLR     EA
                  MOV     A,@R0
```

```
                JNB     ACC.B_WAITTIM,NoWaitTimeout
                ORL     A,#(K_READY+TMO_EVENT)
                MOV     @R0,A
NoWaitTimeout:  SETB    EA
NoTimeout:      INC     R0          ; advance to TaskTimer
                DJNZ    R1,TIMERLOOP

                MOV     A,saveacc
                MOV     PSW,savepsw
                USING   0                   ; Registerbank 0 for following code

IF (TIMESHARING == 0)
; Round Robin Task Switching not required.  System Interrupt ends here
?RTX?SET_ISR:
IF (CPU_IDLE_CODE)
            SETB  ?RTX_ISR_SIG
ENDIF
                RET
ENDIF

IF (TIMESHARING)
; Round Robin Task Switching required.  Check if task generates timeout
; Check for Round Robin Timeout on the current task
                JNB     ?RTX_TS_DELAY,CheckRobinTime
NoRobinTimeout:
?RTX?SET_ISR:
IF (CPU_IDLE_CODE)
            SETB  ?RTX_ISR_SIG
ENDIF
            RET
CheckRobinTime: DJNZ    ?RTX_ROBINTIME,NoRobinTimeout

?RTX_TASKSWITCHING:
                PUSH    ACC
                PUSH    PSW
                PUSH    B
                PUSH    DPH
                PUSH    DPL
                PUSH    AR0
                ……
                PUSH    AR7
IF (CODE_BANKING <> 0)
                PUSH    ?B_CURRENTBANK
ENDIF

                MOV     A,?RTX_CURRENTTASK
                RL      A
                ADD     A,#?RTX?TASKSTATE?S+1
                MOV     R0,A
                MOV     A,#K_ROBIN
                CLR     EA
                ORL     A,@R0
                MOV     @R0,A
                SETB    EA
IF (CODE_BANKING <> 0)
                SJMP    os_switch_task1
ENDIF
ENDIF

;------------------------------------------------
; Perform a Task-Switch
;  void os_switch_task (void)
;      uchar i;
```

```
;     uchar limit;

;---- Variable 'current' assigned to Register 'R6' ----
;---- Variable 'next' assigned to Register 'R7' ----
;---- Variable 'i' assigned to Register 'R0' ----
;---- Variable 'limit' assigned to Register 'R5' ----
;
;------------------------------------------------

os_switch_task:

IF (CODE_BANKING <> 0)
                    PUSH    ?B_CURRENTBANK
ENDIF

os_switch_task1:

;     next = current;
IF (TIMESHARING <> 0)
                    SETB    ?RTX_TS_DELAY                   ; Delay Task Switching
ENDIF
                    MOV     A,?RTX_CURRENTTASK
                    MOV     R7,A
;     while (1)  {
                    RL      A
                    ADD     A,#?RTX?TASKSTATE?S+1
                    MOV     R0,A
?C0001:
;       if (++next == MAXTASKN+1)  next = 0;
                    INC     R7
                    INC     R0
                    INC     R0
IF (CPU_IDLE_CODE)
                    MOV     A,R7
                    CJNE    A,?RTX_CURRENTTASK,NoIDLE
                    JBC     ?RTX_ISR_SIG,NoIDLE
                    CPU_IDLE        ; CPU sleep
NoIDLE:
ENDIF
                    CJNE    R7,#?RTX_MAXTASKN+1,?C0003
                    MOV     R7,#0
                    MOV     R0,#?RTX?TASKSTATE?S+1
?C0003:
;       if (STATE[next].st & K_READY)  break;
                    MOV     A,@R0
                    JNB     ACC.B_READY,?C0001
;     }
;

PUBLIC              ?RTX_NEXTID
PUBLIC  ?RTX_NEXTTASK

?RTX_NEXTID  EQU    AR7
?RTX_NEXTTASK:      NOP                 ; for Debugging

;     while (current < next)  {
?C0005:
                    MOV     A,?RTX_CURRENTTASK
                    CLR     C
                    SUBB    A,R7
                    JNC     ?C0011

;       current++;
```

```
                INC     ?RTX_CURRENTTASK
;       i = STKP[current];
                MOV     A,#?RTX?TASKSP?S
                ADD     A,?RTX_CURRENTTASK
                MOV     R0,A
                MOV     A,@R0
                MOV     R5,A
;       STKP[current] = SP;
                MOV     @R0,SP
;       if (current == MAXTASKN) limit = RAMTOP;
                INC     R0
                MOV     A,@R0
                MOV     R6,?RTX_CURRENTTASK
                CJNE    R6,#?RTX_MAXTASKN,?C0007
                MOV     A,#RAMTOP
?C0007:
                XCH     A,R5
                MOV     R0,A
;       else                    limit = STKP[current+1];
;
;       while (i != limit)  {
?C0009:
                MOV     A,R0
                XRL     A,R5
                JZ      ?C0005
;         SP++;
;         i++;
;         STACK[SP] = STACK[i];
                INC     R0
                MOV     A,@R0
                PUSH    ACC
                SJMP    ?C0009
;       }
;     }
?C0011:
;
;     while (current > next)  {
                MOV     A,?RTX_CURRENTTASK
                SETB    C
                SUBB    A,R7
                JC      ?C0012

                MOV     A,?RTX_CURRENTTASK
                ADD     A,#?RTX?TASKSP?S+1
                MOV     R0,A
                MOV     A,@R0
;       if (current == (MAXTASKN)) i = RAMTOP;
;       else                    i = STKP[current+1];
                MOV     R6,?RTX_CURRENTTASK
                CJNE    R6,#?RTX_MAXTASKN,?C0013
                MOV     A,#RAMTOP
?C0013:
                MOV     R5,A
;       limit = STKP[current];
                DEC     R0
                MOV     A,@R0
                XCH     A,R5
                MOV     R0,A
;
;       while (SP != limit)  {
?C0015:
                MOV     A,SP
                XRL     A,R5
```

```
                JZ      ?C0016
;       STACK[i] = STACK[SP];
;       i--;
;       SP--;
                POP     ACC
                MOV     @R0,A
                DEC     R0

                SJMP    ?C0015
?C0016:
;       }
;       STKP[current] = i;
                MOV     A,?RTX_CURRENTTASK
                ADD     A,#?RTX?TASKSP?S
                XCH     A,R0
                MOV     @R0,A
;       current--;
                DEC     ?RTX_CURRENTTASK
                SJMP    ?C0011
?C0012:
;   }

;   RoundRobinTime = ?RTX_TIMESHARING
IF (TIMESHARING)
                MOV     ?RTX_ROBINTIME,#TIMESHARING
ENDIF

;       if (STATE[current].st & K_ROBIN)  goto RobinOn;
                MOV     A,?RTX_CURRENTTASK
                RL      A
                ADD     A,#?RTX?TASKSTATE?S+1
                MOV     R0,A
                MOV     R7,#SIG_EVENT
                CLR     EA
                MOV     A,@R0
IF (TIMESHARING)
                JBC     ACC.B_ROBIN,RobinOn
ENDIF
;       if ((STATE[current].st & K_SIG) && (STATE[current].st & SIG_EVENT)
;         goto SignalOn;
            JNB     ACC.B_WAITSIG,SignalOff
                JBC     ACC.B_SIGNAL,SignalOn
SignalOff:
;       if ((STATE[current].st & K_TMO) && (STATE[current].st & TMO_EVENT)
;         goto TimeOutOn;
            MOV     R7,#0               ; No Event
                JNB     ACC.B_WAITTIM,NoEvent
                JNB     ACC.B_TIMEOUT,NoEvent
TimeOutOn:
                MOV     R7,#TMO_EVENT
                ANL     A,#0F4H
SignalOn:
NoEvent:        CLR     ACC.B_RDY       ; Clear RDY bit
                XCH     A,@R0
                SETB    EA

                ANL     A,#K_RDY
                ORL     AR7,A
IF (TIMESHARING <> 0)
 IF (CODE_BANKING)
            POP     ACC
                CALL    ?B_RESTORE_BANK
 ENDIF
```

```
                CLR     ?RTX_TS_DELAY
                RET
ELSE
 IF (CODE_BANKING)
        POP     ACC
                JMP     ?B_RESTORE_BANK
 ENDIF
                RET
ENDIF



;-------------------------------------------------
IF (TIMESHARING <> 0)
RobinOn:        MOV     @R0,A
                SETB    EA
IF (CODE_BANKING)
                POP     ACC
                CALL    ?B_RESTORE_BANK
ENDIF
                POP     AR7
                ‘’’’’’’’’’’’’’’’
                POP     AR0
                POP     DPL
                POP     DPH
                POP     B
                POP     PSW
                POP     ACC
                CLR     ?RTX_TS_DELAY
                RET                     ; Restart Task
ENDIF
;   }
;  }



;-------------------------------------------------
; Start RTX-51 Tiny Kernel
;-------------------------------------------------

EXTRN CODE (?C_STARTUP)
PUBLIC          main

main:           MOV     R0,#?RTX?TASKSP?S
                MOV     @R0,SP
                MOV     A,#?RTX_MAXTASKN
                JZ      main2
                MOV     R7,A
main1:          INC     R0
                MOV     @R0,#RAMTOP
                DJNZ    R7,main1
main2:          MOV     R7,#?RTX_MAXTASKN+1
                CLR     A
                MOV     R0,#?RTX?TASKSTATE?S
main1x:         MOV     @R0,A
                INC     R0
                MOV     @R0,A
                INC     R0
                DJNZ    R7,main1x
                MOV     R0,#?RTX?TASKSTATE?S+1
                MOV     @R0,#K_ACTIVE+K_READY
                MOV     DPTR,#?RTX?TASKENT?S
                MOV     A,#1
```

```
                MOVC  A,@A+DPTR
                PUSH  ACC
                CLR   A
                MOVC  A,@A+DPTR
                PUSH  ACC
IF (TIMESHARING <> 0)
                MOV   ?RTX_ROBINTIME,#TIMESHARING
ENDIF
                ORL   TMOD,#01H      ; Timer 0 Mode 1
                MOV   TL0,#LOW (?RTX_CLOCK)
                MOV   TH0,#HIGH (?RTX_CLOCK)
                SETB  TR0
                SETB  EA
                SETB  ET0
                RET             ; Start Task 0


;-------------------------------------------------

PUBLIC ?RTX_TASKIDX
?RTX_TASKIDX:       DB      ?RTX_MAXTASKN               ; for Debugging

        END
```