RTX Real Time Operating System

Eduard Jadroň, Ivan Feňo, Pavol Špánik*

Summary: A realistic view point at multi-tasking Real Time Operating System for the 8051 family of microcontrollers. A Real-Time operating system simplifies system design, programing of complex aplications which required fast reaction of CPU. Typical aplications are: real-time data acqusition and monitoring, microcontroller networks, process control, machine control and robotics. RTX is fully integrated in the A51 assembler language and easy to use. System consistency and task descirption tables are automatically controlled by the L51 linker/locater.

1. Real Time Operating System RTX

RTX can run on single-chip systems wihout XDATA memory requirements and task switching without task priorites and works in parallel with interupt functions. Real Time or multitasking applications are composed of one or more tasks that perform specific operations. Task are simplify 8051 instructions into end-less loops. RTX small allows for up to 8 tasks, RTX compact allows for up to 16 tasks and RTX large up to 32 tasks at the same time.

2. Introduction to RTX

Many microcontroller aplications require simultaneous execution of multiple jobs, called tasks. For such aplications a Real Time Operating System alows flexible scheduling of the availble processing time to several task. RTX implements a powerfull Real Time Operating System which is easy to use and applicable to all 8051 derivates. RTX allows „parallel“ execution of several end-less loops at the same time and RTX divides the available CPU time into time-slice and assigns every task a time-slice. Each task is allowed to execute for a predetermined amount of time. Then RTX switches to another task that is ready to run and allows that task to execute for a while. Real Time Operating System provides a timing functions which is interrupt driven by the 8051 hardware timer T0. The periodic interrupt generates timer tics which drive the RTX clock. RTX recognizes two task states: **running**, **sleeping**. A task may be in one an only one these states at any given time. The **running** states are considered to be active states since task of these states were started by user program. The **sleeping** state is in inactive state since tasks of this state either have not been started or have been terminated.

* MSc. Eduard Jadroň, tel. +421 89 565 2231, ej@edo.spsmt.sk
MSc. Ivan Feňo, department of electrical traction and energetics, Faculty of electrotechnical engineering, University of Žilina, Veľký diel, 010 26 Žilina, Slovac republic, tel. +421 89 565 2231, feno@kete.utc.sk
Associate Professor Pavol Špánik, PhD, tel. +421 89 513 2175, spanik@fel.utc.sk

Table 1. System Requirements

Memory model for RTX	Maximum count tasks	Using register bank RB(x)	Using internal memory IRAM	Using external memory XRAM	Family 51
SMALL	8	RB(3)	030h - 070h	-	8051
COMPACT	16	RB(3)	030h - 0B0h	-	8051
LARGE	32	RB(3)	-	0000h - 00FFh	8052

3. Generation of an RTX Application

RTX is fully integrated in the assembler language, generation of executable RTX applications is pretty easy. The user does need to write any assembler statments or instructions. Translate the program modules with A51 and link with L51.

A51.EXE PROGRAMS.ASM
L51.EXE RTXS.OBJ,PROGRAMS.OBJ TO PROGS.OBJ
OHS51.EXE PROGS.OBJ

Example 1.

```
public Task00,Task01,Task02,Task03,Task04,Task05,Task06,Task07
public Time,CNT,MaxTask,Init,Reload       ;Public vars
extrn number (Max,Bank0,Bank1,Bank2,Bank3,BegRAM,EndRAM)
extrn code (Start,Logo)
;*********************************************
Basic   equ 0000h
CNT     equ 08h
P4      equ 0c0h                ;Define port P4
MaxTask equ 08h                 ;Max. tasks at the same time with RTX
Reload  equ 38h                 ;TH0=Reload
;*********************************************
Prem    segment bit
;*********************************************
        rseg Prem
Flag:   dbit 1          ;Definition var
;*********************************************
Var     segment data    ;Relocable segment
        rseg Var
;*********************************************
Counter: ds 01h
program segment code            ;Relocable segment
        rseg program
;*********************************************
; Task      0     1     2     3     4     5     6     7       ;Tasks[0..7]
Time:  db 01h,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh      ;Define time-slice for Task[i]
;*********************************************
Init:   nop             ;Setting 8051 before start ...
        ljmp Start
;*********************************************
```

```
;*********************************************
        cseg at Basic+001bh
IntT1:  reti
;*********************************************
Prg0    segment code
        rseg Prg0
;*********************************************
Task00:
        mov a,p1
        xrl p1,a
        mov p1,a
        ljmp Task00
Task01:         inc Counter
        ljmp Task01
Task02:         nop
        ljmp Task02
Task03:
        mov psw,#Bank0
        mov dptr,#Logo
Loop:   clr a
        movc a,@a+dptr
        inc dptr
        mov p4,a
        cjne a,#'$',Loop
        ljmp Task03
Task04:
        setb Flag
        clr Flag
        ljmp Task04
Task05:
        nop
        ljmp Task05
Task06:         nop
        ljmp Task06
Task07:         nop
        ljmp Task07
```

4. Conclusion

There are number of general advantages to using a real-time kernel:

- A program may be broken down into individual task are easier to understand and implement
- The modular approach to multitasking programs promotes software resue and allows task to be used in other projects
- Since the kernel addresses the real-time and multitasking issues, more time can be devoted to creating and testing the applications

5. Refferences

[1] ATMEL: 8-bit Microcontroller with 4kB Flash 89C51, Finland Atmel Corp. 1995
[2] SIEMENS: Microcontrollers - Data Catalog, Edition 10.92, Siemens AG 1992
[3] Philips Semiconductors: 80C51-Based 8-bit Microcontrolers, Book IC20, 1994
[4] KEIL: Macro Assembler and Utilites for 8051 and Variants, User's Guide 7.2000
[5] V. Šubrt: Aplikácie jednoèipových mikropoèítaèov Intel, Grada Publishing, Praha 1996