Matt Zimmerer
CPE 439
Spring 2013

AVRX – The ghost RTOS

AVRX is a RTOS that is targeted for AVR devices that have 512 or more bytes of SRAM. It is written entirely in assembly and exposes 36 APIs (tasks, timers, semaphores, and message queue). Unfortunately when I tried to find the developers website, I ran into the following message: “After a hard day of contract work on autonomous cargo delivery systems I just don't have time to maintain a web site.” Nevertheless, there was a tutorial available that contained enough information to get a glance under the hood. (http://tutorialsto.com/mechine/mcu/avr-mcu-rtos-avrx-application.html)

Overview of AVRX features:

- full support for Preemptive, priority-driven task scheduling algorithm;
- 16-level priority, the same priority the task of using Round-robin scheduling algorithm to implement a rotational basis;
- semaphore can be used for signal transmission, synchronization and mutual exclusion semaphores, blocking and non-blocking support for syntax;
- task message queue can be used between the mutual transmission of information, receiving and confirmation message can be blocked and non-blocking call;
- interruption subroutine, the majority of non-blocking of the interrupt service routine can be used;
- support of a single timer queue management time, any process can set a timer, and any task can wait for a timer to time;
- support for debugging single-step process of running;
- space program, including the occupation of all the features of version 1000 bytes;
- with the timer / counter some of the services can be written in the task-level code AVRX.

Due to the fact that this RTOS is written in assembly, it is very hard to port it to other platforms. Although I cannot find examples to back up what the tutorial says, it seems like AVRX is built as a library that has C API. The best information I found was in the form of function names for 28 of the 36 APIs AVRX implements (on the next page), which was spread around the poorly written tutorial I referenced. A note to whoever is reading this, those symbols were copied directly from the tutorial, so all misspellings are intact.

Tasks
AvrxInitTask

Semaphores
AvrXSetSemaphore
AvrXIntsetSema-pore
AvrXWaitSemaplaore
AvrXTestSemapIlorc
AvrXIntTestSemaphore
AvrxResetSemaphore.

Timers
AvrXStartTimer
AvrXTim-erHandler
AvrXCancel Timer
AvrXWaitTimer
AvrX-TestTimer
AvrXDelay

Messages
AvrXSendMessage
AvrXIntSendMessage
AvrXRecvMessage
AvrXWaitMes-sage
AvrXAckMessage
AvrXTestMessage
AvrXWait-MessageAck

Single-step operation
AvrxStepNext
AvrXSin-gleStepNext

System Object
AvrXSetObjectSama-phore
AvrXIntObjectSamaphore
AvrXResetObiectSama-phore
AvrXWaitObjectSamaphore
AvrXTestObjectSama-phore
AvrXInfTestObjectSamaphore

Documentation is non existent, technical support is null. The tutorial I refereed to in this document was poorly written at best. Half the time I felt like I was translating rubbish English into understandabe English, but even then the descriptions of this RTOS's architecture made no sense. Either the writers first language was not English, or he participated in enough assembly level development to damage his language processing abilities. However, the RTOS is free and open source. Also, the kernel is said to be very fast, and compiles to a size of only 500-700B. My personal opinion is to avoid using this RTOS at all costs, and search for an alternative that has more support. However if you want to master AVR assembly, I recommend this RTOS because the source code itself looks to be the only documentation available for this kernel.