Cubesat 2001

Bootstrap and RTOS

by

Jacob Dahl c958209
Kris Kegel Sørensen c973389
Hallgeir Wilhelmsen c971790

Intro

The purpose of this group is to find a RTOS (Real Time Operating System) and design a bootstrap, which has a safemode that loads the RTOS into the on-board computer. The Safemode is the first step in the boot up mechanism. It implements a simple communication protocol between the earth and the satellite. The information gathered from earth will decide the next step in the bootup process. Basically there should be two options, either put the RTOS in the memory and start/execute the on-board software (embedded software) or to upload new patches from earth to the software. Because of the importance of the bootstrap software, it needs to be thoroughly tested and preferably small in program-size/footprint.

The selection of RTOS is mainly decided by the choice of CPU that the on-board computer team selects. The on-board computer team has two computers under consideration, the ARM7DMPI and some MIPS variant (in this rapport, it will be assumed it will have the same specifications as the embedded MIPS32 4Kc CPU).

From a software perspective the following need to be supported by the hardware:

- 32-bit RISC
- Memory Protection
- Sufficient Memory
- Timer and clocks

With the two processors given we have found tree suitable RTOS solutions. From

- VxWorks AE from WindRiver
- ThreadX from GreenHills
- Integrity from GreenHills
- QNX

The CubeSat project is being developed in several small teams. The chance of a software failure resulting from programming errors (thinking primarily address/memory violation) must be dealt with. One way is to have a RTOS with a strong process concept, meaning subprograms are separated in different address spaces. Failures are contained to the given process and will not affect the other systems. Another method is to use the *typesafe* language Ada with its *tasking* capability. To use the inherent synchronisation in Ada, a runtime environment is needed (it is usually implemented using the treads of an RTOS).

VxWorks

VxWorks is a fairly big system witch has been used in several big missions, VxWorks is the most widely used RTOS in the world. One of the missions was the pathfinder mission to Mars. This of course gives VxWorks some reliability and makes it a possible candidate in our RTOS choice. The system is highly scaleable in size, it can wary from a few KB to over 256 KB. It has great support of multitasking and communication protocols witch is important to the software programmers. It has dynamic memory management witch will give us some writing protections. Full ANSI C and C++ is the developing language of VxWorks, it does apparently not support the Ada language. Another good thing with VxWorks is the debugging environment, Tornado. It gives the programmers a lot of neat features and help in their programming. For some reason Tornado 3.0 is a requirement for the VxWorks. The two processors architectures, ARM7 and MIPS, are both supported by VxWorks. Motorola PowerPC and the Intel families are also supported. The price of VxWorks is at this time unknown, but from different sites on the Internet, we estimate it to be around 15k+ $. There is only two disadvantages with this RTOS: The price and that it does not support Ada witch is the software teams preferred language. The VxWorks is very well documented in difference to many smaller RTOS.

ThreadX

ThreadX is one of the RTOS from Green Hills that we have been looking at.
ThreadX supports many processors (http://www.ghs.com/ products/rtos/threadx.html). For our purpose, what is important is, that it supports ARM and MIPS, because these are the ones the hardware group is considering. It has got complete MIPS and ARM7 support. It is royalty free and includes the complete C and assembly source code. The price of the RTOS is about one month salary for a SW-engineer (how much this is, no one knows). By e-mail Green Hills responded that they would give us University discount. Green Hills have their own development environment MULTI, which support the programming languages Ada and C/C++. Unfortunately ThreadX has no Memory Protection. For this purpose Green Hills have developed the Integrity RTOS.

Integrity

Integrity is, as Green Hills write it themselves, "…a secure, royalty-free RTOS intended for use in maximum reliability embedded systems." Integrity uses hardware memory protection to survive user-programming errors. It supports the Ada language and also C/C++, FORTRAN and Assembly. There are several development tools supporting Integrity, the before mentioned MULTI and its Ada extended version AdaMULTI, both are from Green Hills. The Kernel leaves a small memory footprint. The size can be reduced if you do not need all the protection features. As mentioned above, Green Hills are willing to give us some kind of university discount. The source code is not free, as for Thread X, but it is "affordable". Integrity supports MIPS and PowerPC but not ARM7 for the Atmel processor.

AdaMULTI

Green Hills are one of the few companies who have a certified Ada environment.
AdaMULTI suports the ThreadX, Integrity and VxWorks. Its runtime environment can be implemented through POSIX threads (possible support for QNX). The AdaMULTI development environment appears to be expensive, quote "…development costs, at an industry average of 400$ per day, per programmer, can easily exceed the purchase price of the AdaMULTI Development Environment."

QNX

The QNX RTOS is based on the neutrino kernel. The kernel only contains the minimum required:

- thread services
- signal services
- message-passing services
- synchronisation services
- scheduling services
- timer services

If a file system, process-manager or some other resource are needed, the kernel can be augmented/extended through *memory protected* processes. The RTOS is based on several POSIX standards, which make the interface from a programmer's perspective very UNIX like.

QNX can be scaled from small to larger, depending on the amount of augmentation to the RTOS. Unfortunately they do not give any numeric value to ' small' . But it is constantly stressed in all their PR material "how little memory footprint it leaves". They seem to expect an embedded system to use approximately 100k of memory.

MetroWorks have a development environment for the QNX RTOS. It only supports C/C++ compiler (no ADA).

Source code for a generic bootup and a driver is supplied, making it easier for a developer to tailor the software to a specific platform.

Also they allow free download of the QNX® real-time platform. Their online documentation is thorough.

The QNX supports a multitude of platforms. Among them MIPS (R4000, R5000), PPC and StrongARM.

As a note the company has 20 years experience in the embedded field.

Bootstrap

The flow diagram to the left describes how we imagine the Bootstrap going to function.

Description of the flow diagram:

Reset and setup
Receive a reset signal from either a software-reset or the system watchdog. Initialise the safemode function.

Safemode
Wait until a connection to Earth via the radio system has been established. When a command is received *Evaluate Command* decides what to do.

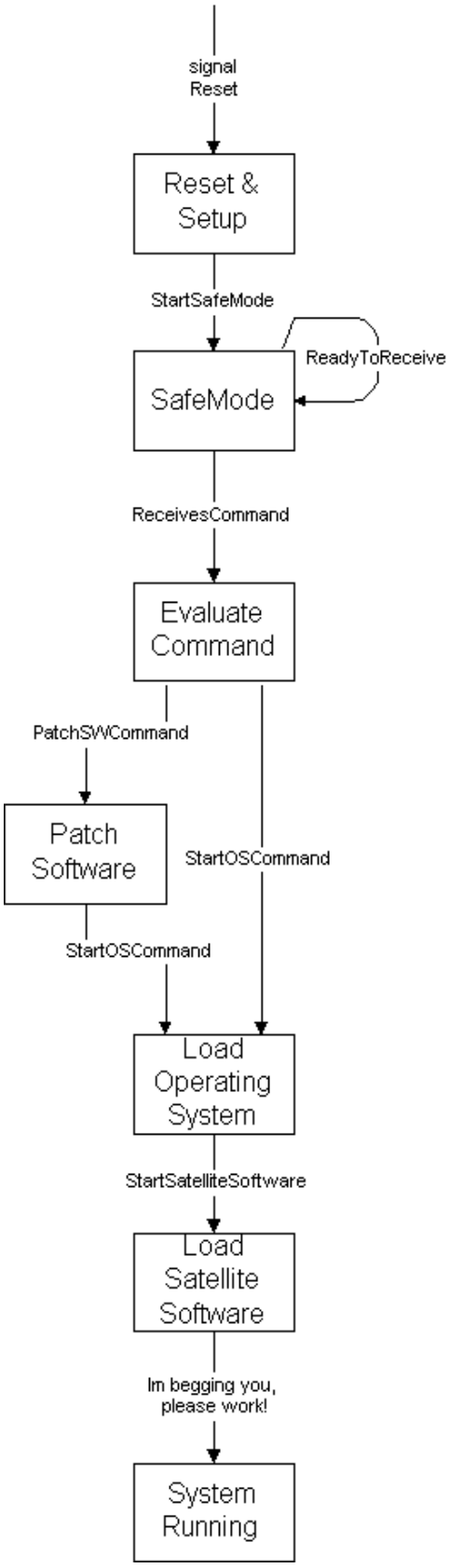
Evaluate Command
Ground Station can send two types of commands:

a) upload new software
b) load the existing RTOS and the Satellite Software

Patch Software
In case of a software failure a piece of new software can be uploaded. How the software is uploaded depends on the radio bandwidth. If there is a large bandwidth it might be possible to upload an entire system and avoid having to patch the software.

Load operation system
Loads the RTOS.

Load satellite software
Loads the current software version. This will activate payload.



Plan

The next two moths we have planned to:

October:

- Week 1 – Investigating QNX and try to create a small embedded system
- Week 2 – Investigating QNX and try to create a small embedded system
- Week 3 – Holiday.
- Week 4 – We assume that the CPU, the development tool and the RTOS that have been chosen and delivered. Make basic testing of the system.

November:

- Week 1 – Install RTOS on the system.
- Week 2 – Bootup software.
- Week 3 – Bootup software.
- Week 4 – Bootup software.

December:

- Week 1 – Communication protocol.
- Week 2 – Communication protocol.
- Week 3 – Communication protocol.

Links

www.ghs.com – Green Hill Software (ThreadX and Integrity)
www.threadX.com
www.qnx.com
www.windriver.com – VxWorks