Managing Complexity – Developing the Node Control Software For The International Space Station

Donald B. Woods
Boeing
13100 Space Center Blvd.
Houston, TX 77059
Donald.B.Woods@Boeing.Com

Introduction

On December 4th, 1998 at 3:36 AM STS-88 (the space shuttle Endeavor) was launched with the "Node 1 Unity Module" in its payload bay. After working on the Space Station program for a very long time, that launch was one of the most beautiful sights I had ever seen! As the Shuttle proceeded to rendezvous with the Russian/American module know as Zarya, I returned to Houston quickly to start monitoring the activation of the software I had spent the last 3 years working on.

The FGB module (also known as "Zarya"), was grappled by the shuttle robotic arm, and connected to the Unity module. Crewmembers then hooked up the power and data connections between Zarya and Unity. On December 7th, 1998 at 9:49 PM CST the Node Control Software was activated. On December 15th, 1998, the Node-1/Zarya "cornerstone" of the International Space Station was left on-orbit.

The Node Control Software (NCS) is the first software flown by NASA for the International Space Station (ISS). The ISS Program is considered the most complex international engineering effort ever undertaken. At last count some 18 countries are active partners in this global venture. NCS has performed all of its intended functions on orbit, over 200 miles above us. I'll be describing how we built the NCS software.

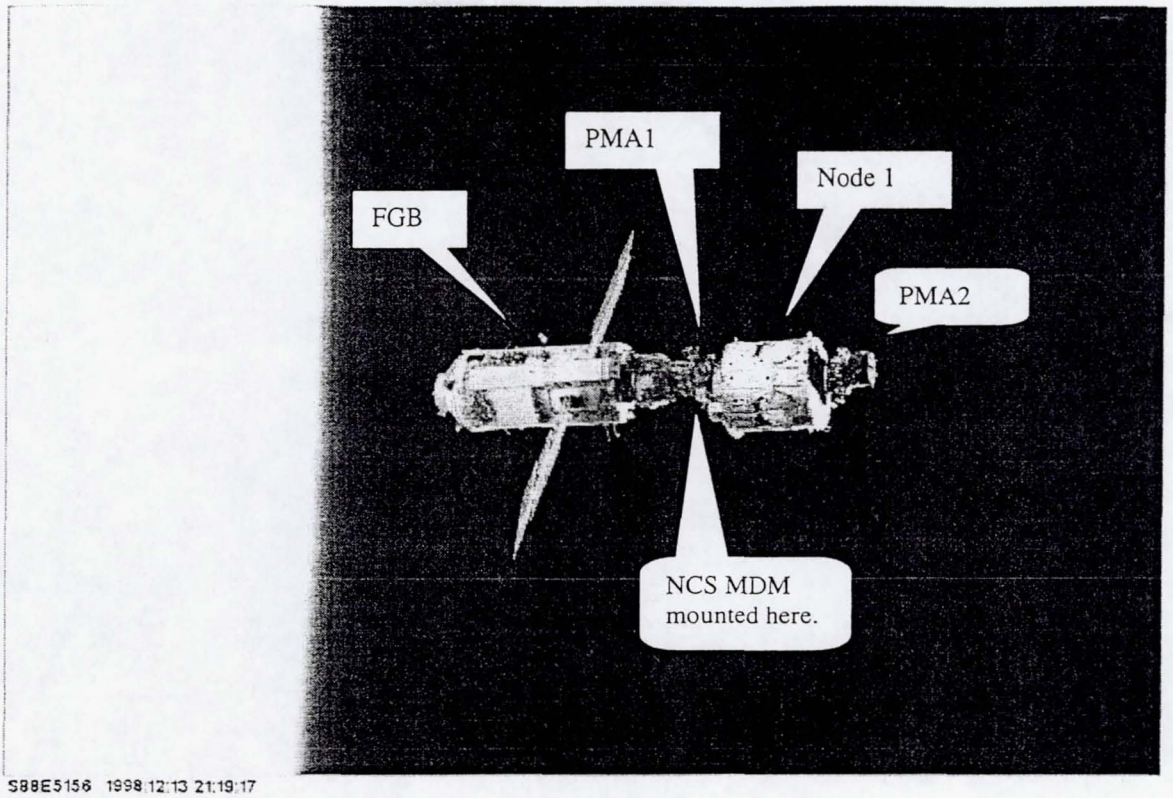

Figure 1. Node 1 Computer/MDM location on-orbit

The Challenges

The NCS is coded in Ada, and hosted on an embedded processor with fairly tight resource constraints (2 MB of RAM, 1 MB of "storage" and a 12 MHz. Processor). This computer is referred to as a Multiplexer/De-multiplexer (MDM).

Early in the assembly sequence of the station, the NCS is the "master" computer; however later on it is one of the "slave" or lower tier computers. NCS controls many of its lower tier components for only a "relatively" short period of time. The hardware below the NCS is not symmetrical, meaning in some cases a device is capable of being controlled by either MDM, but in other cases only one MDM can control a device. Most of the station's computers do not need to function on a "shifting ground work".

System Wide Synergy and Interactions

A summary, high level overview, or context diagram, of the NCS functions and it's external interfaces is shown in figure 2, while each of the NCS functions per subsystem is summarized in Table 2. I say high level because this illustration summarizes some 300 distinct interfaces; some digital, but most are analog.

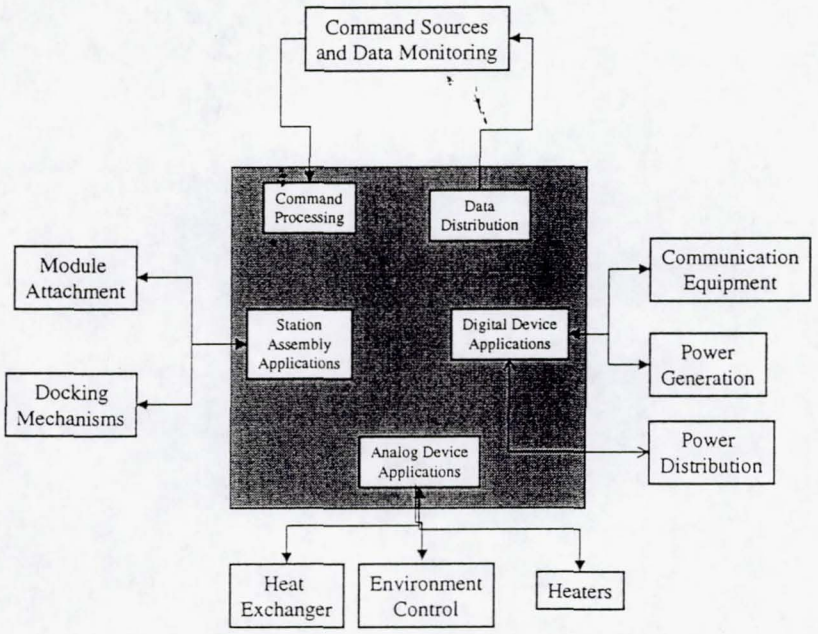

Figure 2. NCS "Context Diagram" – The Node control software is represented here with its interfaces to the external systems. The topside of the figure represents command sources, and telemetry or data sinks. The bottom side of the figure represents the direct hardware connections for the NCS. The left side shows some of the device interfaces pertaining to station assembly and operations, while the right side represents interfaces with digital or 1553 devices

Table 2. NCS Control of Subsystems

Subsystem	NCS Function
Electric Power System (EPS)	• Monitors power and energy usage • Monitors power distribution equipment. • Recovers failures of power computers • Performs "load sheds" if too much power is used.
Environmental Control	• Controls Ventilation (fans and valves) • Monitors for fires and isolates modules if needed • Air quality sampling
Thermal Control	• Monitors temperatures and Controls Structural and Avionics and heaters (over 100) • Controls heat exchanger between internal and external thermal systems.
Assembly Operations	• Controls the Space Station's propulsion systems when the Shuttle docks • Controls latches and bolts for module attachment

NCS has been designed to deal with many different external hardware configurations, as the on-orbit assembly of the Station is completed. Some of the major configuration changes of the station are shown in Figure 3. The software was designed to deal with these configuration changes by allowing the different configurations to be selected by operator command or changes to data tables. Another major challenge was that only one software load was to be built for both computers, even though the hardware connected below the computers is not symmetrical. This challenge was dealt with by configuring the software to deal with all hardware permutations and having computer specific data tables.

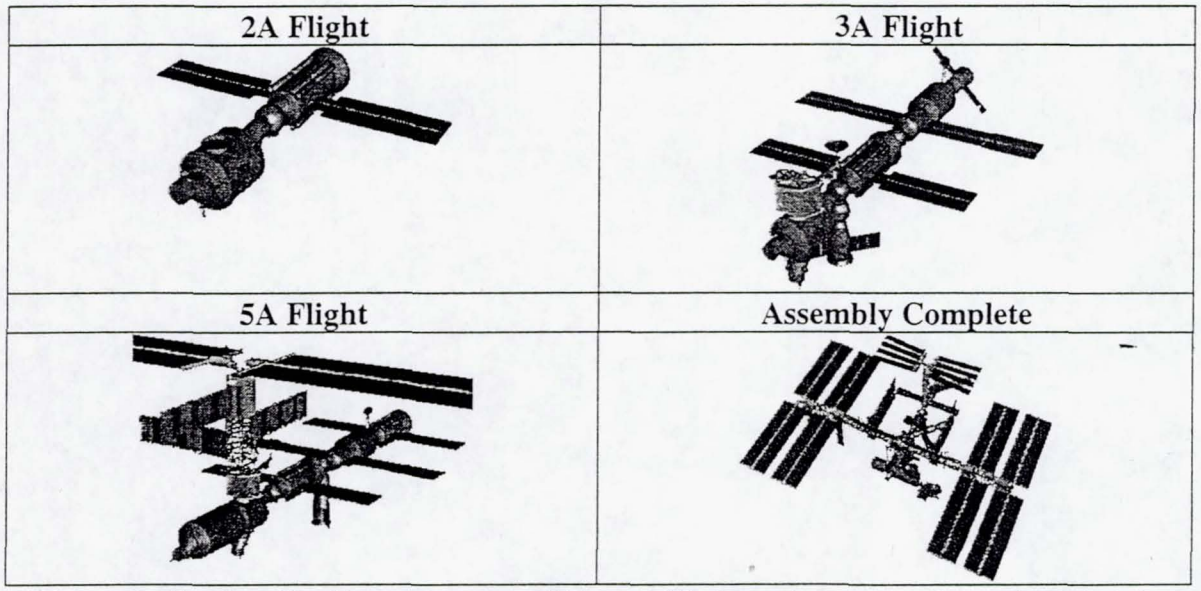

Figure 3. Station Configurations on Different Flights. As new hardware is added the software must adapt to the changing configurations.

Building the Software

I was assigned responsibility for the Software Requirements Specification (SRS) in March of 1995, 3 months later we completed our Software Specification Review (SSR). The 3 years from start to flight definitely seemed to be in fast forward.

At the macroscopic level we followed the traditional MIL-STD-2167A "waterfall" life cycle, with a set of serial reviews of the requirements, design and tests. Some of the practices we adopted which were helpful are described here.

Define the importance of the Problem - Early in the life cycle a board consisting of contractor management (requirements, design and test) authorized changes to the software. Eventually these boards were expanded to include NASA, safety, and operations personnel. When this joint contractor NASA board was chartered they wisely decided that different issue had different operational consequences or severity's. Previously our team would work issues as quickly as possible, but often without seeing "the forest for the trees". With severity's attached to the issues, we could figure out which ones we absolutely needed to fix. The next class of fixes we would do our best to fix as time allowed. In some cases, we would not fix minor problems/annoyances at all.

"Live with it" – One of the philosophical changes in the transition from the "Space Station Freedom" program to the "International Space Station" was a new spirit of compromise and acceptance, often epitomized by the saying "Can you live with it?" Since we realized that not everything had to be fixed, we ended up defining operational notes, which would alert the operators to items where the software behaved strangely or in conflict with the requirements. On flight 2A, the NCS and the early portable computer system were flown with approximately 100 of these operational notes. Most were minor annoyances or only likely to occur after several failures. Most have since been fixed in the "follow on" releases of software.

Find Solutions Outside of your Box - Since the NCS was the first CSCI to be delivered, the NCS team was often straddled with the job of having to define processes in addition to executing to those processes. It was a rare occasion that an existing "space station process" was available for use or adaptation. Occasionally shuttle program experiences and processes provided useful guidance, and inspiration (e.g. "They figured it out, so can we!").

Test the Heck out of the Software - Three different test environments provided feedback into the NCS software development effort. The 3 different test environments were:

- **Flight Qualification Testing** (FQT) – exhaustive testing of the software based on SRS requirements, this is done in the lab with simulated external interfaces. We found most of the "logic" errors in this environment. These tests were normally executed using scripts, which caused changes to input data, and monitoring of NCS response(s).
- **Stage Testing** – higher fidelity functional testing of the software with other flight equivalent computers, including real software. We found timing and data transfer problems in this environment. The stage and integration testing were primarily done manually with test procedures, since most of this testing followed operator

in the loop procedures, which could not be easily automated, since they were operated from a portable computer.

- **H/W and S/W Integration Testing** (HSI)– Highest fidelity testing of the software with the flight hardware, using the actual launch package. This is where we found out about problems with "real hardware" .

Plan for Problems – In the beginning, every test failure sent our team into a "tail spin", but as time moved on we began to expect that tests would cause updates to our software and we started planning accordingly.

Problems are not the Enemy - After each series of tests, the NCS development team was presented with a new set of problems to unravel and analyze. Determining the root cause of a problem is often difficult when one takes into consideration, multiple software loads, data product definitions, test procedures, assumptions, test environment problems, and actual planned operational usage. The original testing schedule was serial (first FQT, then Stage, then HSI), but schedule concerns eventually forced some testing activities to be done in parallel. There was also no real reason they had to be done in a serial fashion, since to a large extent the testing was "orthogonal" and not overlapping. After the initial testing was done, we would build a new release of the software (based on approved issues) and go back for a "delta" or "regression" test of the software. The FQT teams approach to testing was based on scripts; this approach was very far sighted of them, since it allowed them to re-run the whole test suite in 10 to 14 days.

Co-Locate - As problems we uncovered in the code, requirements and interfaces we were able to react quickly because most of our teams were tightly coupled and working in the same area. For example the programmer, tester and data provider were all within 100 feet of each other, so the changes to the products could be quickly coordinated and updates delivered. On the other hand in the bigger sense of the program, we had to work with people in California, Alabama, Florida, Russia, and multiple locations in Houston.

Define your "First Principles" – I eventually defined what I called the commandments or first principals of embedded flight software. While I doubt they are all inclusive, they have helped keep us on track.

Commandments of Flight Software

I. Thou shalt not convert to engineering units.

II. II. Thou shalt provide either a command response or cyclic data for each command received..

III. Thou shalt confirm all failures before alerting the operators or attempting to recover or safe the system.

IV. Thou shalt retry recovery and safing actions (unless the operator has inhibited this response). "Try, try again!"

V. Thou shalt test thy software with real hardware.

VI. Thou shalt require two different commands to destroy things, three to kill. Don't let operator mistakes cause trouble.

VII. Thou shalt synchronize message transfers with an incrementing roll-over counter.

VIII. Thou shalt not open a message unless it is for you. Just route it on its way.

IX. Thou shalt make everything easily modifiable, because you rarely know what is going to actually happen. This applies to limits, failure confirmations and recovery actions.

X. Thou shalt remember the operators commanded values, and never mislead him/her with erroneous data!

XI. Thou shalt base failure status on the current physical state of the system and let the permanent storage of transient information be in time stamped logs of the unique event changes. Don't latch the data, unless there is a real good reason.

XII. When all else fails make sure the PEEK and POKE commands work. (memory modification).

Successes and Failures

Our big success was the 2A assembly flight of the Space Station. On Flight 2A, also referred to as "STS-88" (Space Transportation System Flight 88), the orbiter (e.g. the Space Shuttle) was launched with the node 1 module and PMA1 and PMA2 in its payload bay. After reaching low earth orbit, the Node 1 module was attached to the airlock in the shuttle payload bay. The next major task was to grapple the Zarya/FGB module and attach it to the Node 1 module; this was performed by the Shuttle's robotic arm. Next we hooked up the data and power lines between the FGB and Node 1 modules. See Figure 4 for photos of some of the assembly operations.

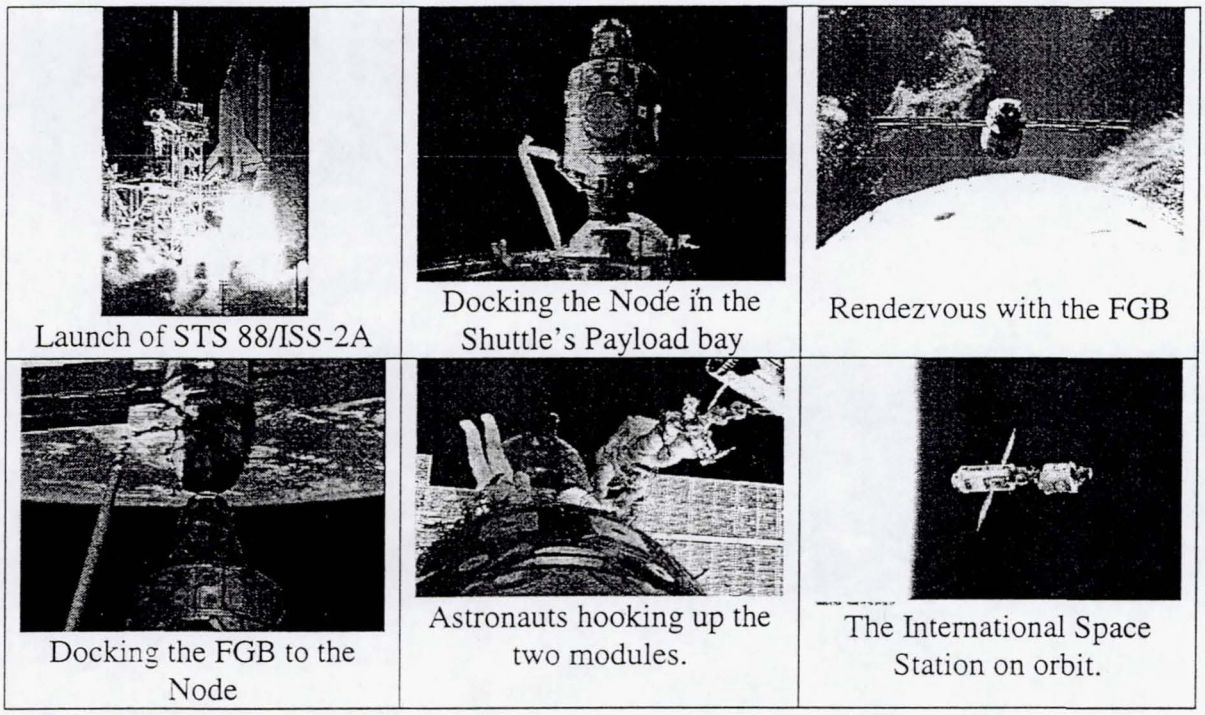

Figure 4. Space Station Assembly and Activation. Some of the major steps in getting the International Space Station assembled and activated on orbit in December of 1998.

At this point the orbiter crew activated the NCS computers, and they began sending data to the orbiter on the status of the station's systems. A representative example of the displays on the portable computer system is shown in figure 5.

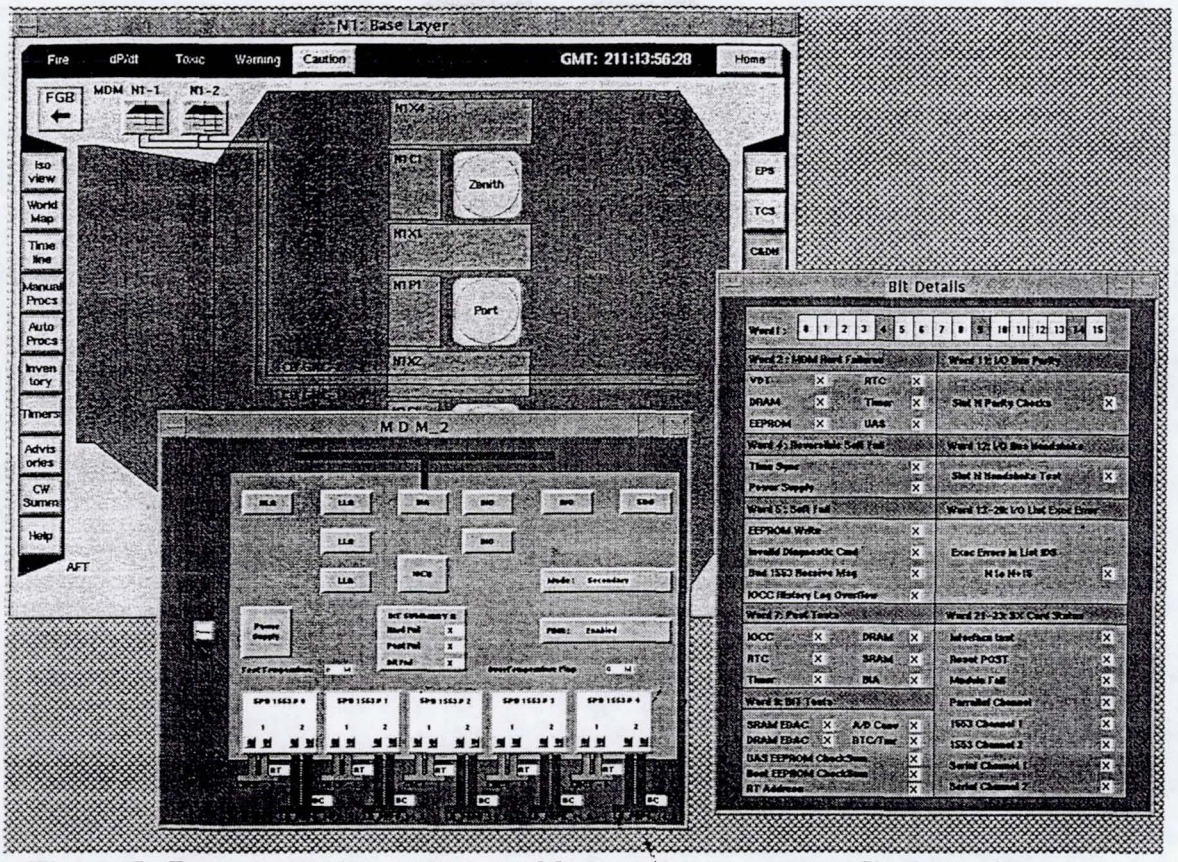

Figure 5. Representative early portable computers systems displays of computer system status for flight 2A. The left most window shows the computers and their buses in relationship to the Space Station Modules, the center window shows the configuration of one of the MDMs and its associated interface cards, while the right window shows the built in test results for the MDM.

The NCS began to monitor the pressure in the node and started operating the Node 1 and PMA-1 heaters. After the Shuttle returned to Kennedy Space Center, the NCS has continued to route telemetry and accepts commands from the ground control stations. During this time period the Node 1 is limited to usage of 800 watts of power in most configurations. See Figure 6 for an example of a ground based monitoring display. Ground operators can also use the portable computer type of screens or review historical data plots.

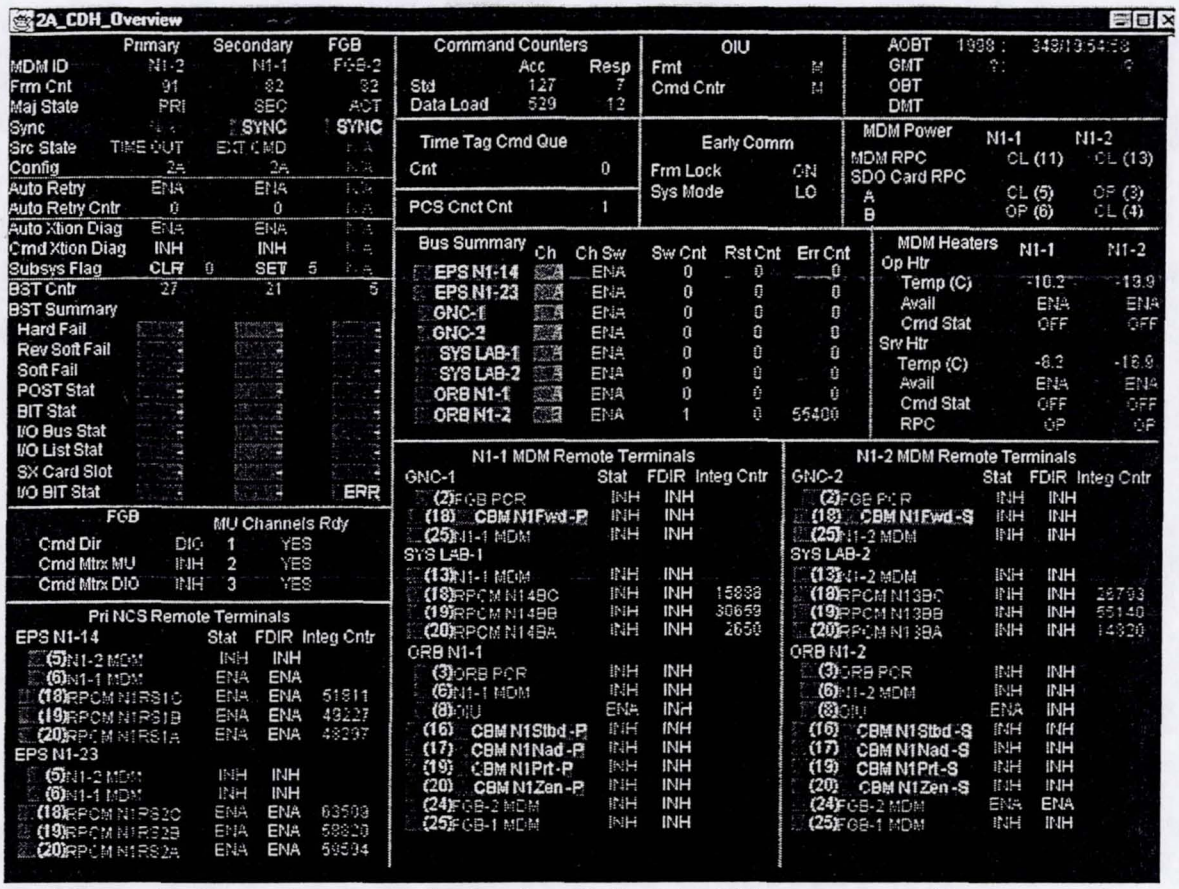

Figure 6. Ground type of display for flight 2A. This screen provides an overall overview of the Station and its computer equipment. Parameters are displayed such as: built in test results, command counters, event counters, power system status, and event response inhibits, time, temperatures of the MDMs, and heaters.

After a year of training and simulations with multiple failures, the ground control team was almost disappointed at how smoothly things went during 2A operations.

Since we had worked out most of our problems on the ground, the few problems encountered on-orbit were minor annoyances and the NCS performed all of its intended functions as planned. Some of the "anomalies" or "funnies" in the computer subsystem were:

- Some of the FGB advisory events were toggling causing the advisory log (a circular buffer) to be overwritten quicker than we could dump the data to the ground. Even though the NCS only logs state changes, these events were toggling so quickly that our circular buffer or log of 100 Advisories was soon filled. Advisories are the least severe anomaly, and are not normally down-linked to the ground. A customized telemetry collection set could be developed to down-link them for this early point in the program, but later on, the bandwidth would not be available due to the large amount of available advisory data parameters for down-link. We worked around this problem by inhibiting these events, which caused them to quit being logged, the raw data is still stored on the ground, but our on-orbit buffers are free to log actual new failures.
- One set of display screens had the C&W events time tag off by four days. This has been fixed.
- The Russian ground station sent 3 commands to the Station which caused the FGB-2 MDM to restart, which in turn led to the FGB sending the NCS a command which was rejected. Proving the international nature of this vehicle, analysts and test teams, in both Moscow and Houston determined the root cause of this event. It turned out the cause was a stale command. Rejecting stale commands is one way to make sure a command isn't executed at the wrong time.
- Two of the Common Berthing Mechanism (CBM) devices were having troubles with timeouts on their local RS-485 buses. Even though the device was within its operational temperature range, they were colder than the other two CBMs. After heating the surrounding area by activating structural heaters, the CBM testing was successful with no 485 timeouts.

Conclusions

Ultimately we were successful, and the STS-88/ISS-2A mission worked, with only minor problems. In building the Node Control Software this quickly several mistakes were made. Some of the lessons learned were:

- **Work Together** - Co-locate the primary team focals (e.g. design, test, and requirements), if at all possible. Although the team started off co-located, it was eventually separated, although technology (e-mail, voice-mail, and telecons) allowed us to continue to work successfully, it was easier "face to face". Co-location leads to fewer chances for misinterpretation or errors in communication. Work toward solutions; instead of identifying the "guilty party".
- **Categorize** - Assign each issue a severity so that an effective work plan can be developed. For example: fix all severity 1 and 2 issues, and fix as many severity 3 issues as possible, fix none of the severity 4 or 5 issues. Assign operational notes for issues that are harder to fix than work around operationally.
- **Good ICDs** – In a program of the size and scope of the International Space Station, the importance of interface control documents (ICDs) can not be over-emphasized. They also need to be provided in a timely fashion.
- **Modifiable** - Make control parameters such as limits, command lists and inhibit/enables modifiable. NCS allowed modification to these items by commands and/or memory writes.
- **Keep Specifications Simple** - Don't specify something unless it is actually required. Every specification will result in design and test team activities; they need to stay focused on the important behaviors, not superfluous requirements.
- **Know the Configuration** - Know the software and hardware configuration and initial conditions before you try to solve any problem. If an issue doesn't specify this contact the initiator or tester.

NCS is the first Ada code used in Manned Space Flight by NASA (Zarya's FGB MDM is also coded in Ada). It was the last computer added to the program and the first delivered to NASA; the entire life cycle from SSR to flight was completed in little more than three years. NCS has 56 digital interfaces, over 260 hardware interfaces, and 42,000 SLOCs (Source Lines Of Code) of Ada. While only using worst case 60% of the available computer processor and memory resources (on a 386sx running at 16 MHz, with 2 MB of RAM!), in fact most of the time NCS has been running at 20% utilization of the CPU.

The 13th release of the NCS software was launched on the 13th flight of the shuttle Endeavor, during my 13th year on the program, showing that sometimes 13 isn't an unlucky number. NCS is truly the first International manned space flight computer, interfacing with 2 different Russian computers.

The Zarya and Unity combination has been called the corner stone of the ISS and in many ways it is just that it is where the two great Space powers on earth, are "joined at the hip". The combined vehicle while "small" in comparison to the station at assembly complete is still 6 stories high, and 35 tons. While the Apollo Soyuz mission connected Russian and United States space vehicles over 20 years ago, this was strictly a mechanical hookup, the ISS connection is also power, data and software.

As we embark on the new millennium, it is good to know that we can work together on a global scale to construct something positive, and have it work!

References:

International Space Station
http://spaceflight.nasa.gov/station/

Node Control Software Requirements Specification –
http://iss-www.jsc.nasa.gov/ss/issapt/scm/srs.htm

Acknowledgements:

This was a team effort and many folks helped, most of them are listed here:
Phil Stranahan, Jeff Schikner, Steve Suchting, Bob Byington, Mike Weston, Kevin Murphy, Rusty Morrison, Joe Reed, Cliff Martin, Carlos Valrand, Dan Leonard, Greg King, Chris Strong, Sarma Susarala, Gary Barber, Barry Greenberg, Dennis Horn, Randy Thurman, Ike Higgins, Chris Land, Mike Langan, Jim Strecker, Mike Surber, Cindy Noble, Kevin Graves, Sivon Kalminov, Wes Tamabayashi, Karl Schab, Bud Wyman, Mindy Cohen, Bill Schaefer, Fred Brown, Bob Stogsdill, Frank Vinz, Michael Van Sickle, Jack Lerohol, Randy Kinney, Kirk Reher, Brian Ellis, Buster Edmonson , Kevin Mutz, Tony Charles, Hai T. Nguyen, Gary Cunha, Jeff Ragsdale, Emilio Santos, Bob Brown, Don Eyles, Jim Dell, Sandra Pinkney, Ceresse Nada, Cheryl Newman, Kevin McDaniels, Daryl Winch, George Hadley, Frank Davis, Fred Decaro, Charlotte Gogola, Jim Peck, Grayum Watts, and Kevin Jackson, Scott Haase, Jimmy Williams, Mark Wilson, Kevin Window, Chris Christenth, Jim Purçell, Kent Lennington, Rich Robitalle, Dave Ansel, Jana Larson, Donna Lilly, Helen Liveringhouse, Chas Curtin, Bob Strate, Bill Krumery, Quin Sheperd, Bob Whiting, Erick Mock, Robert Dave, Laura Matler, Peter Nguyen, Nam Nguyen, Nichole Nguyen, Gary Cooper, , Pete Sardelich, Art Madrid, David Blummentrit, Sherri Carlson, Rocky Chinnapan, Issrael Gonzales, Dave Guibeau, Greg Johnson, Lloyd Johnson, Joe Martinez, Sherri Mutchler, Frank Standa, Margaret Savoy, Becky Kirk, Mike Rodriggs, Colleen Crawford, Rick Mastrachio, Chuck Armstrong, Jose Rodriguez, James McKinnie, Ted Kenny, Ken Bahn, Jim McDede, Bill Stockton, Derek Hassman, Elaine Goddard, Keith Tran, Steve Miller, Macresia Alibaruho,