NIRSPEC

UCLA Astrophysics Program U.C. Berkeley W.M.Keck Observatory

George Brims **September 1, 1998**

NIRSPEC Software Design Note 20.00
Root transputer and housekeeping

1 Introduction

The root transputer is the transputer that is connected directly to the host[1], forming the "root" of our transputer network. Its main function is to examine each message coming from the host and route it to the other transputers, depending on which command identifier ("cid") it contains. Similarly, messages or data coming back from the other transputers are passed back to the host computer.

The other functions the root transputer has to perform are "housekeeping" functions to do with checking the cabinet temperatures inside the electronics enclosure, and turning on and off the lamps in the Calibration Unit. The lamps are monitored by sensor circuits, so the transputer also checks that they are operating properly.

The root transputer lives on a DAQ17 clock/motor board. As such it has four I/O ports (labeled as "motor ports" even though they are in fact completely general-purpose) which are used for the temperature monitoring and lamp control functions. The clock section of the board is not used. The board is identical to the other DAQ17s in the system, though configured slightly differently (see NEDN23 for details). Available RAM is 4 Megabytes.

2 Overall program structure

The program consists of two parallel sections of code, which are best thought of as message buffers. The first (the "from-host" buffer) idles in a loop, checking for messages from the host computer, and either passes them on to the appropriate transputer elsewhere in the net, or in a very few cases performs one of its housekeeping functions itself. The second parallel section (the "to-host" buffer) is essentially a mirror image of the first; it monitors the links from the three branches of the network for messages from the other transputers, and passes them all on to the host. The two parallel sections are not allowed by occam convention to talk to the same link in the same direction, so if the first section needs to reply to the host it has to do so by passing a message to the second section. This is accomplished over a virtual link between the two buffer sections.

[1] The hardware connection is through the external SCSI port on the host workstation, to a device called a Matchbox. Matchbox library software allows the server program to talk to the transputers. The connection to the Matchbox is the same as to any other transputer; a bidirectional link plus network control signals.

3 Buffer functions

The two buffer sections are fairly simple, though the second (to-host) section does have the complication that it has to deal with data packets coming back from the camera sections. Each set of data is preceded by a simple message packet informing the root transputer and the host that data is on the way.

3.1 From-host buffer

This code section consists of an infinite loop (WHILE TRUE construct) which starts with an input statement on the link from the host. This means the process is constantly looking for message packets from the host. When a message packet comes in it extracts the cid and parameter (for details of the message protocol see NSDN03). It then parses the message through a CASE statement operating on the value of the cid. This CASE statement is actually the bulk of the program in terms of lines of code. There are four possible outcomes when parsing a cid value; messages go along one branch of the network to the RS232 serial and motor control transputers, they go along another branch to the slit-viewing camera, they go along the third branch to the slit-viewing camera, or this section of the root transputer program does some simple housekeeping task itself. Once the message has been dealt with through the CASE statement, we go back to the top of the loop to deal with the next incoming message.

The result of the housekeeping functions is always in the form of a simple message packet, usually the same cid as the requesting packet, with the required data as the parameter. These replies are sent to the other buffer section over the link “buffer2buffer”, and the other buffer then passes them to the host.

3.2 To-host buffer

This section, as we said above, is a mirror image of the other buffer, monitoring four possible sources of input and routing everything to one output. In this case however, there can be two types of message. There is the simple message protocol consisting of one byte, one integer, but this time there is also the possibility that the message is a data packet from either of the camera subsystems.

Again this code section is an infinite loop, but this time rather than reading messages from one place, it needs to monitor four links at once. This is made simple for the programmer by the ALT construct of occam. An ALT allows us to list a number of pieces of code, each starting with an event such as a message input. Whichever event actually happens first, its following piece of code is executed to completion. The code then immediately goes back to the top of the loop and waits in the ALT for the next event, so this section services all four possible inputs on a first come first served basis.

In the case of messages from the RS232 and motor transputers, or from the from-host buffer, the message is simply passed straight on to the host. Data packets are always preceded by a message in the simple protocol, notifying the root transputer (and as a result, the host) that a frame is ready. In the event that a message is received from either camera section, it is checked for its cid. If it is anything but a “frame ready” message it is simply passed to the host. “Frame ready” from either camera sends the program into a section of code where it listens for the appropriate number of data packets (2 for the SCAM, 16 for the spectrometer) and passes them all through. This means the host will never get a piece of SCAM data in the middle of a spectrometer frame, or vice versa. There is one difference between the way the program deals with the two types of data. The ancestor of this program in the “Gemini” twin channel IR camera also handled data from two camera subsystems, but those were both 256^2 devices. In each channel data were collected by multiple processors (as here) and re-arranged into a single frame by the root transputer. In this case the size of the array in the spectrometer channel is too large (1024^2) for the memory capacity of the root transputer, so although the SCAM data is rearranged before being passed to the host, the packets from the spectrometer channel are simply passed along to the host workstation, which does the rearrangement instead.

4 Housekeeping functions

The housekeeping functions performed by this program are necessarily simple. If this program spends too much time executing a housekeeping function then it will be unable to service messages passing in either direction between the host and the other transputers, leading to various forms of trouble. There are two groups of functions handled by this processor. One set reads Dallas Semiconductor DS1820 temperature sensors, and the other controls and monitors the Calibration Unit light sources.

4.1 DS1820 temperature sensors

The Dallas Semiconductor DS1820 devices are temperature sensors with a little built-in intelligence which allows the user to request temperature data at any time over a serial line. We use these sensors to monitor temperatures inside the electronics enclosure, which is insulated to keep in the heat produced by the electronics. Each device has a unique serial number burned into its memory at the factory. Because each chip will only respond if you send this number with any request for data, multiple devices can be daisy-chained together on a single 3-wire cable.

The devices are all connected to motor port 1 on the DAQ17 board. Three wires go from this port, which is set up to be bi-directional. These are ground, +5V and signal. When the DAQ17 code drives the signal line low the DS1820 chips respond after set intervals by pulling the line high or low to return the requested data serially.

There are three functions to be performed: reset the temperature sensors, get the 64-bit ID from a sensor, and request the temperature from one of them. The get ID command actually has to be sent twice, since our return parameter is a 32-bit number.

4.2 Lamp control

The control of the lamps is very simple. Two of the general purpose I/O ports (“motor ports”) of the DAQ17 board are used to send out logic signals which control relays, turning the lamps on and off. The four bits controlling the arc lamps come out of port 4, and the three bits controlling the incandescent lamps from port 2. The status bits from the sensor circuits are read through the input bits of port 3.

There are two commands, one of which comes with a bit pattern indicating which lamps are to be on or off, and the other requesting the status bits to check which lamps are lit. The result of the status inquiry should be identical to the last set of command bits sent, otherwise we have a hardware problem, most likely a blown bulb or arc lamp.