USING the RSS, low-level firmware command info UPDATED 06-06-2001
By L. Harrison, ASRC, SUNY at Albany

****
******

RSSmicro and CCDmicro

This documents the status of firmware for RSSmicro(20combo).asm and CCDmicro(20combo).asm, which implement the new PIO interface, and also the digital PID. (Yankee doesn't use the digital PID code at present.) These codes also implement the conventional/"electronic" shuttering modes, and support the Portable Calibrators.

WARNING: to ASRC users, note to YES users, this code adds a RH watchdog to protect the Peltier, this changes the RSSmicro Z command and drives a Rort2 output pin.

An important comment: these are the first "merged" codes where one code base with careful conditional bit-definitions and only one non-portable routine (RPHASE in the CCDmicro code) can support both systems. With this, there is much less issue about deviance in user-interface or code between the two.

****
******
* Quick Introduction to Basic RSS Code and functionality *
****
******

The two CPUS in the system communicate via their own interface. In normal (as opposed to diagnostic/test) operation the user interacts only with the "RSSmicro;" it acts in turn as the master controlling the "CCDmicro" as a slave. The user needn't be explicitly aware of the interaction between the two or deal with the CCDmicro.

A direct serial interface to the CCDmicro exists, and it has its own set of commands through it. These are used for many calibration and test purposes, and can also aid system debugging.

The RSSmicro can control temperatures in the system directly with a full digital proportional integro-differential (PID) working in the background. ASRC-built systems uses this, Yankee's do not. The code does no harm in Yankee's applications.

Here is the full command set listing of the RSSmicro. This is the only

device a user must communicate with in normal service. Note that the RSSmicro keeps time, and has a command set very much like the MFRSR for user convenience between the two. The RSS micro has the same auto-baud and login behavior as the MFRSR (but the password is different and there is no super-user password as it is assumed there is only one user).

After login the RSSmicro will give prompts which are either ?> or !> the former meaning the time is not set, just like the MFRSR. On the otherhand the CCDmicro doesn't autobaud or login, doesn't keep time, and sends >> as its prompt. If you see >> you are talking to the CCDmicro!

These commands for the RSSmicro are explained briefly assuming the reader is familar with the standard MFRSR command set.

"A" display analog channels. Same as MFRSR, debug use only. See addendum 5 for a list of channels vs function. Note that temperature channels are thermistor bridges and have non-linear outputs, operating code calibrates these for display in degrees C elsewhere but not in this command.

"B" set baud rate, same as MFRSR

"C" change memory. same as MFRSR, debug use only.

"D" display memory. same as MFRSR, debug use only.

"E" this sets an exposure to the numeric value (in hundredths of seconds), then does a "light" (shutter open) and subsequent "dark" (shutter closed) exposures, and then outputs the data. If the value $FFFF is selected for the exposure ($FFFF = -1) then it will trigger an auto-exposure cycle first and do that exposure. Note that this command is not permitted unless the time is set AND the standard shadowband operation is halted.

"F" The "Force" command for a one-shot directly-commanded shadowband cycle. Note that this command is not permitted unless the time is set AND the standard shadowband operation is halted.

"G" This is the "GO" command, same as MFRSR. "G 0" stops routine operation, any other number starts it.

"H" Hang-up, establish Superuser, same as MFRSR. Super-user currently not implemented... any user is "superuser" but some dangerous commands are directly guarded other ways.

"I" Initialize run operation, similar to MFRSR. Sets sample time only

(unit doesn't average), and controls header output data. The first 32-channel AUX parameter sets analog inputs to be reported as temperatures (lienarized and converted to degrees C), the second reports channels as digital counts. On ASRC's RSS the following is the typical initialization:

I 0 $04FE $0B00 60 1 \ report all temperatures and other analog signals

or

I 0 $04FE $0A00 60 1 \ as above except do not report thermistor excitation

"J" provides a user command to set the Portable Calibration control variables in the following order: # cycles of the exposure sequence, minimum exposure, exposure increment, and maximum exposure (in hundredths of second). The cycles are limited to 1 ... 50 and all the other numbers must be 10 ... 1791 ... the latter upper limit somewhat arbitrary but this tends to limit really CRAZY total burning times and also prevents endless loops. The current standard for ASRC's RSS (and what is set by default on reset) is:

J 10 20 20 120 \ ten cycles of 0.20, 0.40, 0.60, 0.80, 1.00, 1.20 sec

The old Temp-test time-series output (engineering test use only) function which was J remains available as a hidden code available from "R"

"K = Kind of error/prompt behavior

"L" LAT/LONG entry, same as MFRSR. The parameters are Band-offset, Lat, Long. Note that the band offset will be negative, and our sign convention is that western longitudes and northern latitudes are positive.

"M" MIKBUG load, diagnostic only. DON'T USE THIS UNLESS YOU KNOW WHAT YOU are doing. It is guarded against casual-use error. See the code source for instructions; used with the X command to reload the CAT EEPROM

"N" Set bottom and top of the stripe binning range. This is important. Note that there are now 4 variables, BOTBIN, BOTFIL, TOPFIL, TOPBIN. These must obey the relationship

$$0 \le \text{BOTBIN} \le \text{BOTFIL} < \text{TOPFIL} \le \text{TOPBIN} \le 255$$

These are geometric parameters of the illuminated swath on the CCD array.

"O" This sets the latitude offset for coordinate rotation mode. It functions identically to MFRSR. See extended discussion below.

"P" Serial-set PORT-behavior command, same as MFRSR

"Q" This allows the shutter state to be altered from the RSS port. This
Is ABNORMAL, and for diagnostic/test purposes ONLY. In order to prevent
user error the first parameter must be $ABCD. The second parameter is
sent to the CCDmicro. If its highbyte = 0 it forces conventional
mechanical shuttermode, if non-zero the new "electronic" shuttering.
The low byte sets OSHBIT. You better know what you are doing to do this!
Note that this state persists until reset, and that it can also be set
from the CCDmicro interface (generally laboratory use only).

See addendum 5 for the bit definitions, here are the three commonly-used
choices :

Q $ABCD $FFC0 \ this returns the system to standard "electronic" mode
Q $ABCD $00C0 \ this selects older conventional-shutter mode
Q $ABCD $FF02 \ enable flat-field test (ASRC unit only, Yankee's doesn't)

"R" Run routine, DON'T DO THIS, it is guarded ditto

"S" Display system status

"U" Update clock (set time and date) routine; same as MFRSR

"V" Memory test algorithm

"W" This command allows the user to set the CCD autoexposure parameters
from the RSS interface. The command requires 3 parameters, which are
in order the maximum pixel target count, the minimum exposure, and
maximum exposure. The target count is in instrument counts, the
minimum and maximum exposures are in hundredths of seconds (just like
"E")

In order to prevent hopeless error the target count is not permitted to
be less than 3000, and the two exposures are not permitted to be less
than 0.05 sec or more than 40.95 seconds, and the max must be > min.

"Y" Set PID parameters ... see addendum 3 below

"X" This is the entry, together with the "M" command for in-field reprogramming.
It is guarded to prevent casual errors. It is not documented here; you
must read the code source comments for info.

"Z" Set AC Params ... see addendum 3 below. NOTE THAT THIS NOW HAS THE
RH setpoint as the first parameter.

********** Operating the RSS
****

The new control paradigm and user-interface is much simpler than the old one where the two CPUs shared a single RS-232 interface with the user. In the new one the user is communicating with the "RSS-micro" (this is the microprocessor that keeps time, runs the shadowband, does the PID functions, etc.); it treats the "CCD-micro" (that runs the CCD array and shutter... and that's it) as a slave, communicating by a private interface. When the CCD-micro takes data under the command of the RSS-micro the data are transferred back to the RSS-micro over this private interface, and then sent to the user.

The CCD-micro still has it's own RS-232 interface, but this is now a diagnostic port on an RSS, not connected in "normal" use. The easiest way to introduce the new command set is to go through the login and set-up process:

Hello:
CCD-RSS (PIO/PID) rev 2.2, L. Harrison, 06/04/2001
CCD controller rev 2.2 (ASRC) June 09, 2001

For now the login password is "Hello:" (that shouldn't cause any strain to remember!) The supervisor password isn't needed (nothing is guarded). Note that the login message is telling you what the CCD controller is. This is real (polled from the CCD controller), not a hardcoded string in the RSS-micro prom. Furthermore what you don't see is that in addition to the ascii message there is a number passed, which the master compares against a list of CCD proms it can handle. The other messages are:

'CCD info not avail. while running'

'CCD not responding'

'Unknown/Incompatible CCD'

The first message can't appear now (because you can't leave it running and not logged in), but it is there for the future. The others are obvious. After that here are the things you would likely want to do through the master. First, set the time, if instrument has lost power or reset.

?>U 0
YEAR= 1999
MON = 5
DAY = 28
HOUR= 12
MIN = 1

```
SEC = 1
!>N 64 196       \ sets BOTTOM ROW, TOP ROW {illegal while running, duh}
!>L 0 0.0 0.0
$0000 $0000
!>I 0 $F 0 30 1  \ just like I command on logger,
!>               \ but the first 32bit AUX sets temp measurements, and the
                 \ second 32bit AUX sets measurements returned as counts.
```

The # to average isn't functional, but it is checked ... so set to 1.

```
!>G 1
Homing the band....DONE
!>

*36307.50138    -99.99  -99.99  -99.99  -99.99  # 25
32767   32769   32769   58957
32771   32771   32771   19611
32775   32775   32775   6455
```

et cetera..... lots of lines deleted: these data are garbage because no CCD attached. Note it prints 4 temps due to $F bit flags, these will be analog channels 0 ... 3 in order, converted to temps. Note -99.99 = nothing connected (as here), like the UV instruments now.

```
*36307.50173    -99.99  -99.99  -99.99  -99.99  # 25
32767   32769   32769   32769
32643   32771   32771   32771
```

ditto

```
G 0                         stop instrument
!>O 100 20 30    "ONESCAN" equals old "Z" command
*36307.50260    -99.99  -99.99  -99.99  -99.99  # 25
32767   32769   32769   32769
32771   32771   32771   32771
32775   32775   32775   32775
```

ditto ... note this command waits until motor is home before returning prompt

```
!>E 100   this sets EXPOSURE _and_ does old "W" command
*36307.50422    -99.99  -99.99  -99.99  -99.99  # 100
32769   32769
32769   32769
```

2 long columns deleted ditto

********** CCD Diagnostic Port Info **********
* *
* this section documents the command set of the "CCDmicro" by itself *
* *
****

Following reset, the CCDmicro starts running its parallel interface command loop as its standard background task. In "normal" RSS usage (where the CCDmicro is a slave to the RSSmicro) there is no need for any connection to the serial port of the CCDmicro. However while running as a slave the serial port emits diagnostic information (useful really only for debugging of the parallel interface software, and at the time of this addendum [6/8/2001] extrememly well tested), but also the CCDmicro can be exercised as a stand-alone device through its serial port, either for engineering/calibration testing of a complete RSS or as a stand-alone item for instruments not requiring the shadowband sequencing of the RSS.

DIAGNOSTIC OUTPUT FROM THE CCDmicro

When running parallel commands issued by the RSS-micro diagnostic data comes out the serial port of the CCD-micro. It is normal to see sequences like

D31
D32
D33
D34

etc. This cycle is exactly what you will see while operating in the standard shadowband-running mode for the four measurements.

Any errors announce themselves as "BELL' and then something.

There is one circumstance where a BELL is legal/expected ... on reset (after about 1 sec) if the MASTER doesn't come up out of reset by this time. In our present system this would be an indication of hardware fault somewhere, because these two systems have the same digital logic supply. Other than that you should never get one and I want to know about it (with the file) if you do.

In addition to these low-level parallel interface diagnostics ALL THE serial-interface commands on the slave that DID ANYTHING FUNCTIONAL remain EXACTLY the same -- not a byte of code or function changed. These remain for a good reason: they are our bridge to all the past testing. There is only one functional difference: on the old CCD controller proms you

explicitly started the (much dumber then) parallel control mode. In the new ones of course it must be running in background all the time. The "G" and "F" commands have disappeared ... and ONE new command been created. Q $ABCD (the "QUIET" command) _disables_ the parallel control interface parser -- It could be a nuisance if you are doing stand-alone testing talking to the CCD controller itself. Q any other number re-enables. There is a possibility (meaning I can't say there isn't) that you could permanently hang the master (as opposed to temporarily hanging it while a command is not being obeyed) if it was trying to run the slave while you had "Q" on the slave. Common sense says stop the master running (if it is connected). Oh yeah, one other thing -- the CCD controller now automatically does an A/D calibration on reset. It also does it once per standard shadowband cycle (which it always has, on the new board/CCD firmware).

STAND-ALONE SERIAL-INTERFACE COMMAND SET OF THE CCDmicro

On reset the CCDmicro communicates 19200 baud, 8 bits, no parity, 1 stop bit.

Just to avoid confusion, a reminder: this section documents the serial interface command set of the _CCDmicro_, through its RS-232 connector, which is not the normal interface to a RSS. This is the "diagnostic" port on a RSS, but the CCDmicro can also be used as a stand-alone controller for an instrument which doesn't shadowband.

When using a standard CCDmicro prom in a stand-alone application it is advisable to issue as the first command Q $ABCD (the "QUIET" command) which disables the parallel control interface parser. This would be unnecessary so long as the parallel interface is inactive, but why risk noise problems?

As is the case with the MFRSR, RSS, etc. all commands are parsed only by the first letter, and require at least one numeric argument, which may be in decimal or hexadecimal.

Here is the table of all serial interface commands, from CCDmicro12.asm, with annotated updates for 12b and 12c ... and Yankee's "combo" CCDmicro code. (last update to this 03/13/01)

"A" any number -- calibrates the A/D

"B" sets the baud rate. Choices are the numeric argument values 1200, 2400, 9600, 19200. Any other value will be rejected.

"C" change memory command, identical to MFRSR. Don't use this unless you know what you are doing!

"D" display memory contents, identical to MFRSR ditto. Less dangerous, but still don't use this unless you know what you are about.

"E" sets the exposure time in hundredths of a second, for all exposures (light and dark).

"F" this sets/clears electronic exposure mode. Any non-zero number forces electronic exposure mode, zero sets conventional mode. WARNING: if you use this command through the serial port of an operating RSS (where it is being driven by the RSSmicro) the shutter-mode will persist until hardware reset. The RSSmicro doesn't set the shutter mode. The reset shutter-mode is "electronic."

"G" requires four parameters 0..255 in order to set BOTBIN, BOTFIL, TOPFIL, TOPBIN. The relationship $0 \le BOTBIN \le BOTFIL < TOPFIL \le TOPBIN \le 255$ is enforced. This is the ONLY place these can be set, via the CCDmicro interface. However they can also be set from the RSSmicro interface.

"H" This command allows the user to set the autoexposure parameters. The command requires 3 parameters, which are in order the maximum pixel target count, the minimum exposure, and maximum exposure. The target count is in instrument counts, the minimum and maximum exposures are in hundredths of seconds (just like "E")

In order to prevent hopeless error the target count is not permitted to be less than 3000, and the two exposures are not permitted to be less than 0.05 sec or more than 40.95 seconds, and the max must be > min.

"K" error recovery kind ... allows user to turn on/off echo and on/off prompts.

0 = prompt, echo 1 = no-prompt, no-echo
2 = no-prompt, echo, 3 = no-prompt, no-echo

"M" any number, binary form of "N" dump (see next). Either the M or N commands are used to readout the data collected from a "W" exposure. Here is the full commentary on this command

"M" does the "same thing" as "N", but with a binary output for speed. The code is also written to be as fast as practical. See warnings below, and note that the output is organized differently:

first: a preamble or synchronization frame is written that is $FC, $FD, $FE, $FF

Then the NPIXEL elements of the first exposure come out as

2-byte unsigned numbers (higher-order first), then the NPIXEL elements of the second (or dark) exposure come out subsequently.

WARNING: if you have echo on, and you SEND any char during the dump you'll get garbage.

WARNING#2: this routine puts out UNSIGNED data for consistency with old codes and N below. The new A/D creates signed data and this is what the routines that read the CCD store into memory. The sign conversion for the ASCII output is done "nondestructively" by all other routines (see "N"), such that you could invoke those routines more than once on a set of stored data and get identical outputs. HOWEVER, this binary output routine works differently: in the interest of maximum speed and minimum latency this routine simply "remaps" the xmit buffer pointers "onto" the data, rather than copying it. It first does a sweep through the data changing all the signs. (This is done while the serial port is spooling out the preamble, so it doesn't "cost anything" in time).

The consequence of this is that REPEATED USE OF THE "M" ROUTINE WILL FLIP THE SIGN BIT EACH TIME.

"N" any number, puts out two columns of ascii numbers with a tab in between, used to output the results of a "W" command in tabular ascii (or to put out the result of the new "F" command)

"P" Port command, same as RSS/MFRSR. Sets/clears char echo, LF after CR, etc.

"Q" This is the "QUIET" command mentioned above. Q $ABCD disables the parallel interface. Any other value reenables it. The RSSmicro may hang if the CCDmicro disables its parallel interface: do this only for stand-alone operation or for debugging when it is acceptable to reset the instrument to restart it.

"R" runs a routine, i.e. jumps to the address specified. Obviously you need to be able to create/load code to use this, otherwise don't.

"S" this command is available on codes after 12c only. It allows you to remap the 8-bit field which defines the "open-shutter" state (sent to the latch), or to simply force the shutter open or closed for test purposes.

Redefining the bits is primarily useful for test purposes and for supporting a flat-field illuminator.

The "S" command responds to positive non-zero arguments by redefining the 8-bit bit mask of "open shutter." The low-byte

of the operand is OR'ed with the NRCbit and stored to the variable which defines the open-shutter state. (it is not possible to use this command to do shutter open with NRCbit low, but it is easy now to use the C command to achieve this simply by directly fiddling OSHTbit if you read the code)

Zero resets the standard shutter

-1 OPENS the shutter, sending whatever is the current defined state.

- any other number closes it.

Remember here that numbers are treated as two's complement but the input number routine also accepts the unsigned range ... e.g. for the purpose of this routine 65535 = $FFFF = -1

"T" access to test routines. You must read the comments in the source code to use these. The test routines are version specific.

"U" allows the user to readout and store 3 selected "row stripes" through the array, where the stripes are column-binned. This routine has the same interface behavior as "X" below in that the 4 negative argument control options are obeyed, and if positive then you are building a table of the row positions. Note that the table storage overwrites the storage for the table for the "X" command and vice-versa. Read the comments at the routine for more info.

"V" Verify Memory -- this routine does a "marching bits" memory test through the specified memory range. It reads and restores the data in the test locations, so this routine is 'safe' to run on the external RAM space (because NMI routines only directly access the CPU's smaller internal ram space, and the core loop that saves, checks, restores, each byte is doing SEI..CLI to make this operation 'atomic' WRT IRQ interrupts).

In all cases the instrument should be stopped to do these tests.

Note that our (ASRC's) RSS instruments have older CPU hardware with external RAM from address $4000 to $BFFF. (This is the same memory map as the MFRSRs, and the V command is on all instruments, including MFRSRs, where it originated.)

The newer YES built instruments have the same memory map as our newer U1000 instruments, but not the MFRSRs or our earlier RSS instruments. The external static RAM is from $200 (used) to $7FFF in these instruments.

The CPU also has a small internal SRAM store where many state-critical variables

are stored. You can check the used range of this by verifying $40 $100, but there is a possibility that doing so will corrupt variables used by NMI or report them as not working properly when they are (because the NMI cannot be masked. With the instrument not idle, the RSSmicro NMI code should only affect these addresses...

```
SSTATE  = $A0  ! byte: MOTLATCH register state

! The following variables are used by the SHADOWBAND stepper (Motor2)

NSTEP   = $9E  ! word: the number of steps to move
STEPN   = $9D  ! byte: controls step interval (in msec)
STEPC   = $9C  ! byte: counter for STEPN
INDEX   = $9A  ! word: Index Position offset

! Timekeeping registers

NORENT =  $8A   ! Flag to avoid re-entrancy of NMI
CNT5   =  $89   ! BYTE: modulo 5 counter for 1kHz interrupt
MSEC   =  $87   ! two bytes (counts 1000 - 1)
SEC    =  $84   ! three bytes (counts DOWN !! $15180 -1)
```

and the odds should be very high that an NMI doesn't "catch" you at these addresses. If you get a diagnosed memory fault at any other address, or reproducible memory faults for any of these addresses, the CPU itself is bad.

The V command should not alter the state of the system UNLESS you test the CPU's static ram, where an NMI collision with the routine could corrupt the registers above. You should check/reset the time afterwards (the stepping motor parameters will recover from upset, in any event)

"W" This is the work-horse routine of the instrument. For any value other than $FFFF (aka -1) it takes two measurements, both with the pre-specified exposure interval. (For $FFFF it first does an auto-exposure to set the exposure value.) This command will obey the shutter-mode selected with the F command. The first of the two reported measurements is shutter-open and the second shutter-closed; both are stored into memory. If you selected the autoexposure mode this command prints the resulting exposure found.

WARNING: this command has changed compared to previous versions, it no longer uses positive arguments to set BOTBIN TOPBIN. These can ONLY be set using the N command, which sets all four params.

DON'T send W -1 by misake!

The stored data can then be read out with either the "M" or "N" command.

"X" implements the capability to read out selected rows of pixels (up to 10 ... all there's room to store). If the first argument is 0 or greater then the routine continues to build the ordered table of rows you want to read. Elements are 0 ... 255 and the list is terminated with a negative number. Any number > 255 is detected as an error. Out of order numbering is also detected as an error. See comments at "U" command also

If negative here are your "processing" choices:
-1 shutter open, ascii output
-2 shutter closed, binary output
-3 shutter open, binary output
-4 shutter closed, ascii output

"Z" The "ZAP" command is a low-level Parallel-PORT control command useful only for specialized instrument interface purposes. It is documented in addendum 4 below.

****
*
*
* Addendum 2: Coordinate Rotation Code *
*
****
*

RSSmicro(16) and later implement the coordinate rotation capability from the P2 MFRSR codes. If the user does not want to use the coordinate rotation capability it is identical to the earlier code and user interface capability with two differences, which are important:

1.) The O command now sets the axis-offset (as it does in the MFRSR with coordinate rotation capability), and the F command is now the one for the "one-shot" FORCESB routine.

2.) The other 32bit parameter in the I command now sets up analog channels to be emitted after the temperature channels are emitted.

About the Coordinate Rotation capability, taken from the release notes for the P2 MFRSR code:

The new solar ephemeris algorithm has both improved accuracy, and a full coordinate-rotation extension which can do the generalized alt-azimuth rotation. (a MFRSR never never goes all the way to alt-azimuth because the shaft cannot be vertical.) This permits the motor shaft to be at a "latitude angle" where its axis is no longer the polar axis, as defined by the variable ROTA, set with the "O" command. On reset ROTA is initialized so that no coordinate rotation is done, so unless subsequently overwritten by the user the default behavior is "like the old systems," but CosZA accuracy is substantially improved, a particular benefit for those averaging data in MFRSRs, but no particular issue in RSS applications.

The offset-latitude capability extends usability of the instrument at high latitudes, can improve very low-latitude use (< 15 degrees) by getting the motor housing below the horizon, and potentially simplifies mechanics of of the motor mount, which can now have a small number of fixed positions.

The side-correction measurement angles

The old algorithm simply used a fixed angle of 9 degrees in Hour-Angle for the two "corection measurements" to either side of the one which blocks the sun. The actual angular separation of the sun from the band then depended on the solar declination (maximum when the declination was zero) and a minimum of 8.3 degrees at the solstices. With the coordinate rotation the pseudo-declination can be larger than 22.5 degrees, and at fixed band correction angle the angular separation narrows rapidly (nearly quadratically) as the declination increases. Hence when the coordinate rotation algorithm is used, a routine is called to evaluate the band rotation angle to hold a fixed 8.3 degree angular separation. If no coordinate rotation is being applied (meaing that the motor shaft is on the polar axis) *this*code* uses the old 9 degree fixed hour-angle solely to provide as complete backward functionality (n all subtle details which could affect data) as possible. The new algorithm is more accurate, and in a sense "should" be done no matter what, but here continuity is cheap to provide.

Associated with this are changes to the control logic for whether the motion is full-circle (standard MFRSR at low latitiudes) or to-and-fro (MFRSRs at high-latitudes, UV-MFRS and RSS at all latitudes). This motion change to to-and-fro is needed when mechanical limits prevent the band from swinging through the nadir. The previous control-condition for the motion change was a fixed threshold latitude constant; when the instrument was at a latitude (either south or north) equal or greater than that value the motion would be to-and-fro. For UV-MFR and RSS instruments the constant was simply set to zero.

The old strategy won't work in any usable way given the much larger number

of permutations possible given the coordinate rotation capability. The new decision algorithm for full-circle vs. to-and-fro motion is very simple and controlled by the value of the INDEX offset parameter (which the user enters for the instrument).

In any case where the band cannot swing through the nadir the home-detect position must be offset substantially (to the side which is east in the northern hemisphere). Typical values of INDEX for these cases are -70 to -100 for RSS instruments at any latitude.

DON'T SCREW UP -- DON'T SET INDEX > -20 on a RSS. (This isn't yet guarded against in the code!

If the INDEX parameter is greater than -20 then the full-circle ("low-latitude standard MFRSR") mode is entered. The value of -20 chosen here is somewhat arbitrary; it simply allows a generous range (9 degrees) of software trim when operating in the low-latitude mode. In any high latitude or UV-MFR the negative INDEX value will be substantially beyond 9 degrees, so this provides a simple and direct decision rule.

In "high latitude" operation there is similarly a motion travel limit so that the band doesn't hit the detector or arm near local midnight. With the coordinate rotation capability the new angular limit algorithm simply insures that the band doesn't go beyond the complement of the INDEX offset on the far side of the band motion. This is both easier and better than the old scheme; the user sets the INDEX offset to obtain band clearance on the east side (in the northern hemisphere), and the motion will not come closer to the detector or arm on the other... taking advantage of the east-west symmetry of the instrument.

The user sets the INDEX parameter as the first parameter with the Lat/Long command, so it must be set for any instrument startup. On reset however the INDEX is now set to -128 to protect UV-MFRs from "head banging" if they are started without a proper latitude set.

Setting up an Instrument with a Latitude-Offset (Using the coordinate rotation capability)

Set up itself is simple: set the desired mechanical angle, and then enter the "rotation angle" ROTA by using the "O" command (this is the letter Oh, the only new/changed command with this version). Note that the value entered is an _integer_ and represents an angle in 16-bit fraction-of-a-circle units, which you compute as follows:

Define AROT to be the geometric angle between the true polar-axis and the physical mounted shaft axis, where

In the Northern Hemisphere AROT is POSITIVE if the motor housing lies "below" the position it would to be for a polar axis drive (i.e. the shaft is pointing closer to the zenith than the polar axis would). This case arises naturally at low latitudes where we want to mount the motor-housing "lower" so that the housing doesn't protrude above the horizon, but could occur at other latitudes if there were only a fixed number of mounting positions and one needs to pick the closest.

In the Northern Hemisphere AROT is NEGATIVE if the motor housing lies "ABOVE" the position it would to be for a polar axis drive (i.e. the shaft is pointing farther from the zenith than the polar axis would). This will be the case inevitably at very high latitudes where the shaft cannot reach the polar axis without the motor housing colliding with the instrument. Ditto happening at other latitudes for mechanical convenience.

In the southern Hemisphere, FOR A MFRSR, the above sign convention is reversed because the instrument has been rotated so that the motor axis points to the SOUTH pole.

ROTA(degrees) = 90 - AROT

When AROT is negative this angle will be greater than 90 degrees; don't worry about that. Having computed the ROTA angle in degrees it must then be converted into 16bit fraction-of-circle by multiplying by $4000/90 = 16384/90 = 182.0444444 and rounding to the nearest integer. This is the value to which the ROTA variable should be set.

Background about this angle definition if you care:

Because the Alt-azimuth rotation is used so commonly the internal sign conventions and the rotation matrix are expressed with ROTA defined as equal to latitude for an alt-azimuth rotation. However the actual included angle of the rotation done for the Alt-azimuth problem is is 90 degrees minus the latitude. This becomes clear if you consider the case where the Latitude = 90 = the north pole. Here the Hour-angle is already the Azimuth and the the Elevation angle is the Declination; no rotation is needed. So for the purpose of implementing a "shaft latitude offset" the actual angle rotated (call it AROT) is again AROT = 90 degrees - ROTA.

****
*
*
* Addendum 3: PID Temperature Control System (ASRC Units only) *
*
*

****

The temperature control system is run by the RSSmicro.

Only the ASRC-built units use the digital PID temperature control system at the time of writing [06/05/01]. The Yankee units have more temperature control channels and use an analog PID system derived from an earlier design used by ASRC. There are trade-offs both ways; the digital design is more flexible and tunable and saves some hardware, the analog system is more conservative and eliminates a user-interaction issue.

********** ASRC Temperature Control Subsystems **********

The RSS firmware initializes the temperature control parameters so that under "normal" circumstances the user need not set these up explicitly after turn-on. NOTE THAT AT PRESENT THE HEATERS ARE INTENTIONALLY HELD DISABLED
UNTIL THE TIME IS SET. It will take at least 30 minutes after the heaters are running for the instrument to settle into near steady-state, and several hours should pass before "final" wavelength calibrations attempted. During all of this time however the instrument is serviceable for all basic measurements.

The following is a more detailed description of the system thermal housekeeping functions and user-commands to control set-points and alter behavior. Even though you may not need to reset any of these parameters it is important to understand what the instrument is doing so that you can detect and understand "off-normal" events, interpret the status outputs the instrument displays, etc.

Starting with VERSION 20, the system implements an internal RH detector and control watchdog for the Peltier cooler.

0) The Peltier watchdog: The Z command (see below) allows you to to set a theshold (in mV) above which the RH signal input is deemed unsafe for the CCD. At present the on-rest value for this is 1250.

   This control algorithm ALSO has a hardwired broken-lead detect function, if the input signal is below 255 mV.

   On reset the Peltier is turned off until time is set. This has precidence over all other control states of the Peltier. (Same as PIDS) When the time is set, if the RH is valid AND ERRFLAG = 0 (the system error state) it will turn the Peltier on.

   Anytime the RH input either exceeds the setpoint or is detected as a broken

lead a counter is incremented once each second. This is set to zero if a valid signal is received. If this counter reaches 128 then:

The Peltier is disabled

Shadowband operation is halted

A RHERR state is set into ERRFLAG.

The consequence of the last is that the Peltier will not be turned back on even if the RH returns valid, until a G 1 is done.

The special value for the RH threshold of $7FFF will DISABLE the watchdog so that the Peltier will be forced on any time there is not a ERR state.

Warning: the Z input command detects any Peltier threshold below the value 256 or above the value 4096 as illegal.

A) The air-chiller:

The entire instrument enclosure is equipped with a circulating air-chiller (miniature air conditioner). Under all but very cold conditions this runs continuously. These devices do not like to be cycled, nor do we want so many transients from doing so. Instead an internal air heater adds heat (controlled by one of the four PID channels) to maintain the internal air temperature at the desired set point.

Only at temperatures sufficiently cold that the air-chiller is clearly unnecessary do we want it to turn off. The control logic for the air-chiller is driven from the outside air temperature (OAT) sensor. There are two thresholds (which are user-settable through the Z command (see shortly below)): ACLOTHR and ACHITHR. These thresholds are set in mV (A/D counts) for the thermistor bridge voltage.

These threshold values and the state of the AC control are displayed by the status command, see example below.

There is a hidden byte-variable which is a counter. Once per second the system task will sample the OAT and compare to the thresholds.

If greater than ACHITHR the counter is incremented

If less than ACLOTHR the counter is decremented

If ACLOTHR <= OAT signal <= ACHITHR the counter is set to 0

If the counter saturates to 127 the air-chiller is turned on. If the counter saturates to -128 the air-chiller is turned off.

This algorithm ensures that the air-chiller doesn't cycle rapidly. At present on reset ACLOTHR is set to approximately -5 °C, ACHITHR is set to 0 °C, and the counter pre-loaded to 124 with the air-chiller off. If the ambient air is warmer than ACHITHR then it will turn on in three seconds.

To set these parameters simply do

Z (RHLIM) (ACLOTHR) (ACHITHR)

where (RHLIM) (ACLOTHR) (ACHITHR) are numeric values., e.g. The function of RHLIM is discussed above in the section on the Peltier watchdog. Warning: the Z input command detects any Peltier threshold below the value 256 or above the value 4096 as illegal.

Z 1250 1435 1685 \ this sets the default state

Here is an example of the status command more than 3 seconds after reset in "warm" conditions. The fourth line shows the two air-chiller control thresholds (corresponding to approximately -5 and 0 °C), followed by $0F, a control flag for the PID heaters discussed below. The 1 at the end of the fourth line indicates that the air-chiller is enabled; if this is zero then it has been commanded off.

The next four lines are all zeros: the heaters are not running because time has not been set.

```
?>S 0
$0000  $0000 0.0000 0.0000
00:00:00   12-32-0  0.00000 (0)
3600 3600
1434 1684 $0F 1
0     0     0     0     { Prism/Optics }
0     0     0     0     { Enclosure Air}
0     0     0     0     { Fore-Optic ('snorkle')}
0     0     0     0     { Entrance Slit }
```

B) The PID heaters:

This is a bit more complicated. At present the PID heaters are disabled

(and their outputs forced off) until the time is set. This is intended to be conservative until we have more experience. After time is set the background PID function starts running. Values are computed and controls updated every 4 seconds; the actual duty-cycle modulation is done by the NMI routine and is fixed 1 Hz frequency, variable duty cycle, sliced in mS. The control numeric range is thus 0 ... 1000, zero being no power and 1000 full power.

The bit constant shown as $0F in the example status command output above (third item on the fourth line) is the PID control parameter. $0F is it's normal state. The four low-order bits are individual enables for the four control outputs. These should be on for normal instrument operation. The PID algorithm detects broken or not-connected control inputs (necessary because an open input appears very cold, and would otherwise cause the heater to "lock" at full power). If this occurs the algorithm will turn off ALL the heater power and disable it's own operation by setting the PID control parameter to zero. You can attempt a restart if this occurs as described below.

The high order four bits of the PID control parameter select/deselect continuous PID diagnostic outputs from the individual PID channels. This can be turned on to yield PID control data with every 4 second update (when not conflicting with data output from the CCD). Generally this is not wanted. See "Setting the PID" at the end of this note.

```
!>S 0
$0000  $0000 44.1375 72.5482
12:13:44   06-17-1999  36327.50953 (0)
3600 3600
1434 1684 $0F 1
-1      -1373   95      255
-4      -3473   876     138
-1      -816    206     152
-2      -148    28      57
```

The example status command here shows an output after time has set and the system has settled "near equilibrim" (there isn't any such thing as true equilibrium given the feedback controller and external temperature variations). The last four lines now show PID operating status for the 4 PID channels: in order from top to bottom Prism/Optics, Enclosure Air, Fore-Optic (aka 'Snorkle'), and Input Slit.

The columns from left to right are error from setpoint, integrator value, power output, and small-error lock counter. The error here is in digital counts at 4x the typical precision, achieved through 32x digital oversampling. A count in this scale is approximately 0.005 °C. All of the channels show

small errors, the enclosure air temperature being the worst: approx. 0.02 °C.

The critical temperatures for wavelength accuracy are the Prism/Optics (first line) and the Fore-Optic (third line). The design of the system isolates these to the greatest degree practical. Commonly the enclosure air temperature and the slit show larger errors, and respond to external transients to a greater degree than the more critical controlled values. The above example though is very typical.

The second column is the integrator value. Interpreting this requires understanding the specific control parameters set. It is primarily useful when "tuning" control-loop behavior.

The third column shows the heater power setting: from 0 to 1000 full scale. Note that the Slit heater is running at very low power (2.8%). The power settings fluctuate quite significantly from update to update as the algorithm tries to control the temperatures very closely; in order to form an estimate of the mean power you need to average several samples. (Remember the 4 sec update rate.)

The final column is a counter of the number of consecutive updates that the error has been within a specified (by the PID setup, see at bottom) error margin. The counter limits at 255. Interpreting this also depends on knowing the setup parameters. At present on reset this error margin is set at 5 for all channels (about 0.025 °C). When this counter is over 100 the control algorithm shifts to "fine control" parameters (which may not be different from the coarse ones). The Enclosure Air and Input Slit values hop outside this domain frequently. The Prism/Optics and Fore-Optics values are often "in small-error lock" for long periods.

********** SETTING THE PID PARAMETERS **********

The PID is controlled through the "Y" command. This command has three fundamentally different functions depending on whether the input parameter is negative, zero, or a positive number.

Zero is the easiest: it simply displays the PID control parameters for the four controlled items.

```
?>Y 0
#1 ____              { the Prism/Optics-Box}
12000
10 0 200 0 1 4
5 100 10 0 200 0
```

```
#2 ____          { the Enclosure Air heater}
10600
4 0 16 0 1 2
5 100 2 0 16 0
#3 ____          { Fore-optic Barrel (aka 'Snorkle')}
12384
4 0 16 0 1 2
5 100 2 0 16 0
#4 ____          { Entrance Slit }
13200
1 0 2 0 1 3
5 100 1 0 2 0
```

The other choices of the "Y" command have the potential of disrupting instrument performance if you goof them up (and perhaps even damaging the instrument).

Negative numbers allow you to set the PID control flags. You enter the negative of the number for the 8-bit set of flags:

```
  1 enables Prism/Optics temperature control
  2 enables Enclosure Air temperature control
  4 enables Fore-Optic temperature control
  8 enables Entranc Slit temperature control
 16 print diagnostics for Prism/Optics control
 32 print diagnostics for Enclosure Air control
 64 print diagnostics for Fore-Optic control
128 print diagnostics for Entrance Slit control
```

Only under very unusual conditions (something wrong) would you wish to disable any of the temperature control functions.

The most common thing to do is to turn on continuous diagnostic outputs. Y -255 will turn on outputs for all the channels, Y -15 returns to all controls enabled and all printed output off.

When printing is on you will see lines like the following printed every four seconds. These are automatically supressed when they would interfere with CCD data output, so you can turn this on and leave it running if you wish while the unit is operating in "GO" mode.

```
+20.72 +20.10 +07.40 +18.53 +05.21 +21.57 +28.22
-672 -2673 1000 0 -2279 -4201 1000 0 -168 -4201 1000 0 -585 -4509 434 0
```

These are shown here as two lines with a convenient break, but are printed as one, for easy ingest into data handling tools (typical use). The first numbers with + (or minus) signs and a decimal point are the standard temperatures

linearized by the internal firmware (note this has been upgraded so precision is now better than 0.01). The remaining numbers are the error, integrator, power, and small-error lock counters for the selected channels, in order. Note that if a channel is not enabled (bit in first block of 4), it won't be printed.

********** Changing PID Control-loop Parameters **********

Formally, changing the PID control parameters is easy. (The function of each parameter is explained below, but understanding the load and loop dynamics to arrive at "optimal" values is a question not addressed here!)

Y 1 12000 10 0 200 0 1 4 5 100 10 0 200 0

sets PID # 1 (the first parameter in the line) to the remainder of the control parameters shown. These parameters are (in order) the items from table 1 below. They are also the items in order shown by the Y 0 command, for a single PID, but entered on a single line and preceeded by the PID number to set.

If you err entering any of these parameters the command restores the on-reset default values for all the parameters of the individual PID. Thus to restore any PID to it's on-reset parameters just type

Y N

where N is the PID number, and nothing else. The absence of the remaining parameters is an error which forces the restoration.

Note that the setpoint is the thermistor bridge output in mV * 4. (Table of value vs/ temperature in appendix not yet provided)

The default values have been arrived at through a mix of some modeling and significant testing, and there aren't many circumstances where a user should alter these. Under extreme temperature conditions it may make sense to alter set points. (This will necessitate at least wavelength recalibration). In general this should not have much impact on the other control coefficients.

There is an enormous literature on PID control theory to which the interested reader should repair ... but be advised that it is hard to apply because the system here doesn't have linear control authority (no heater does), and the various loads act like heavily damped high order loads (in effect like a first-order load coupled to a transmission line), with inexactly known transfer functions. So theory and modeling only get you so far.

One case that may arise is operation under very hot conditions, where heaters, particularly the Slit heater, may have such low duty-cycles as to be not very

stable. Under these conditions reducing _all_ the gains may inprove stability at the expense of transient response capability.

More About the PID control algorithm:

The internal PID algorithms are implemented to be very general. The input signal scale can be anything represented as a 16-bit two's complement number, or any subrange thereof. This means you can handle unsigned number inputs up to 15 bits (0..32767)

The output signal scale can be anything up to 2's complement 16 bit range (unusual... much more typical is 8 bit to 12 bit ranges). The output range is set by the constants MINPOW and MAXPOW, set in this application as 0 and 1000 respectively. The PID arithmetic is very careful (provided you have parameters initialized properly!) to saturate and unsaturate properly.

This PID updates it's stored parameters, and solves the following standard PID equation to produce the control output:

Output = G * (a * error + b * derivative + c * integral)

G is the feedback gain polarity, + or - 1. When running heaters, that are driven "positively" from the control output (more positive number = more heat), and the input signal gets more positive for higher temperature, then G is negative 1. It is a hardware attribute, and in this RSS application there is no command at the user interface level to alter this. With G set properly for the hardware the remaining terms a, b, c are always positive for any "plausible" problem.

The a, b, c coefficients can be set by the user; for each you are permitted a coefficient of the form x / 2^y ... where x can range from 0 .. 255 and y from 0 .. 255 as far as storage, but large y values make no sense.

for example x = 3 and y = 4 yields 3/ 2^4 = 3/16

PID control algorithms in general do not show substantial change in control behavior for small variation in coefficients. The integerization of these algorithms and the coefficients as shown is not performance-limiting and speeds computation very substantially on a small microprocessor compared to floating point arithmetic.

These coefficients as entered are applied literally each time the control routine is called. The dynamics however do scale with the update rate. The error term is independent of the calling rate, the derivative coefficient (b) must be scaled UP proportional to the update frequency and the integral term scaled DOWN proportionally ditto, to maintain the same dynamic

behavior.... down to a low update rate where the loop is no longer fast enough to keep up with the load dynamics. A rough rule of thumb is call this routine at least twice per time-constant for "first-order" loads ... more frequently if tight control is needed or the load is higher order.

In addition you are permitted two sets of these parameters, one for "large error" conditions and another for "small-error" conditions. You set two parameters to control the transition: a small-error limit (obeyed symmetrically for +/- errors) and a number of cycles the system must be within the small error limit before the transition to the small-error coefficients. Both these parameters can be 0 ..255 Always set the small error parameters, even if you just set them equal to the large error values (common).

Table 1: user-setable PID control parameters

2 bytes = control set point (two bytes, signed)
error gain multiplier (one byte unsigned)
error gain divide-by shifter (ditto)
derivative gain multiplier (one byte unsigned)
derivative gain divide-by shifter (ditto)
integrator gain multiplier (one byte unsigned)
integrator gain divide-by shifter (ditto)
small-error limit (ditto)
small-error cycle counts (ditto)
small-error gain multiplier (ditto)
small-error gain divide-by shifter (ditto)
small-error derivative multiplier (ditto)
small-error derivative divide-by shifter (ditto)

Note that the integrator coefficient cannot be changed "on the fly" without renormalization or a major transient, Hence the large/small error options use the same integrator coefficient. Note also that when there is zero error & derivative it is the integrator coefficient that controls the output. Hence the following must be true, or you will never be able to achieve the specified output range at "equilibrium." Sometimes that is desirable, but not often!

32767 * N / 2^S >= MAX_OF(ABS(MAXPOW), ABS(MINPOW))

Where N is the integrator gain multiplier, S is the integrator shift divisor. If MAXPOW, MINPOW are numerically large this limits the shift divisor. These situations are unlikely as there is little need for a 16-bit output for a PID. (you'd want a 32-bit integrator in that case). Alternatively you want a slower update rate.

In addition to the control parameters the user selects above, the firmware establishes the following for you:

input signal source
feedback gain polarity; forced negative for all these heater channels
maximum integrator limit
minimum integrator limit

Limits on integrator range are desirable to cope with turn-on or other large transients. They avoid large integrator overshoots which otherwise can significantly extend settling times. Commonly a little extra margin from the equilibrium computation is desired, to allow the loop to run "in lock" right up to to the power output limit. Here an extra margin of 5% is provided for these integrator limits.

The max/min integrator limits are derived from MAXPOW and MINPOW, and the user's integrator divide-by multiplier. The input routine computes them for you to avoid user error and inconvenience, because it is just a bit tricky. In the codeflow Maxin is the _most-positive_ integrator limit, and Minin is the most negative. (Both can be positive or negative, or Maxin positive and Minin negative, but Maxin always > Minin !) Computing these depends on the feedback gain polarity. If the polarity is positive then they follow conventionally, e.g. with 5% extra margin

Maxin = (((5 * (MAXPOW - MINPOW)) / 100) + MAXPOW) * 2^S / N
Minin = (MINPOW - ((5 * (MAXPOW - MINPOW)) / 100)) * 2^S / N

where N is the integrator gain coefficient and S is the integrator gain shift divider. If the feedback gain polarity is negative then they "switch roles" as well as sign

Minin = -(((5 * (MAXPOW - MINPOW)) / 100) + MAXPOW) * 2^S / N
Maxin = -(MINPOW - ((5 * (MAXPOW - MINPOW)) / 100)) * 2^S / N

In the event that the user has selected an integrator coefficient not meeting the equilibrium condition (given 16 bit range) described above, the computation of Maxin, Minin saturates and doesn't make matters worse.

****
*
*
* Addendum 4: The "ZAP" command for output-port control *
* THIS COMMAND IS CCD-MICRO INTERFACE ONLY *
*
*
****

ZZAP creates a general PIA-driver interface to access all parallel I/O resources

of the CCD driver board EXCEPT the ADLATCH. However, you can fiddle with these ONLY if you know what you are doing, and the PIO interface has been turned off with the Q $ABCD command. This protects against the most egregious stupidities that could be done with these commands on a running RSS. (Which may not run afterwards!) The purpose of the ZZAP command is to allow the CCDmicro board to output/input digital state signals under RS-232 control when the spectrometer is being used in conjunction with other systems; it allows simple digital control for a variety of purposes.

Note that the USERLATCH (and also the ADLATCH) are write-only latches: all pins are outputs and no memory or state information is retained. The shutter control (and some additional lines on Yankee's systems) come from this port, and fiddling with this port's output state will generally cause CCD trouble unless you really know what you are doing and restore its state appropriately.

PORT5 and PORT6 are full PIA ports, controlled by their respective data-direction registers, which commands here allow you to set. On reset these ports get configured to the PIO interface states and you must not fiddle with these ports UNLESS the PIO interface has been disabled via the Q $ABCD command. To use these for your functions you must first set the input and output configuration(s) you want, using the "MAKE" option of the command. In general you do this only once. NOTE that following RESET, PORT 5 is set so that PIOBSY OR PIOHSS are OUTPUTS on PORT5
and all other pins (including all PORT6 pins) are set as inputs. PORT6 is all inputs in resting PIO state.

The ZZAP command works like this; you pass a 16bit number, the high-order byte of which is a function selector and the low-order byte is the data. Here is the function selector table, which sets what you can do with this command:

0 Store the data to the USERLATCH
1 Error, not valid command
2 Store the data to Port 5
3 Store the data to Port 6
4 SET the specified (1) data bits on Port 5 (i.e. OR the bits)
5 SET the specified (1) data bits on Port 6 (ditto)
6 CLEAR the specified (1) data bits on Port 5 (i.e. AND (NOT(the bits)))
7 CLEAR the specified (1) data bits on Port 6 (ditto)
8 MAKE the specified (1) bits of data as WRITE bits on Port 5
9 MAKE the specified (1) bits of data as WRITE bits on Port 6
A READ ports 5 & 6, depending on "data" byte for output format

The two MAKE commands which set the data-direction registers for PORT 5 and PORT 6 immediately (1 microsec) set all output pins LOW (zero) following the DDR strobe, so that they have a defined state as quickly as possible.

when you use the read command the "data" in the low-order byte sets the read command functionality. The bits are flags for the following:

bit7 (of 0..7) read and report the PORT5 data
bit6 report it in decimal (if not, then in HEX)
bit5 read and report the PORT6 data
bit4 report it in decimal (if not, then in HEX)

obviously with these independent flag fields there are combinations which might not make much sense from a user perspective, but they are legal. Here is the complete table of these options, with data byte in hex and low-order nibble set to zero. The "sensible" and preferred usage is shown with first words capitalized:

F0 READ AND REPORT BOTH IN DECIMAL (port 5 is first output, port 6 is second)
E0 read both, report port 5 first in decimal, port 6 second in hex
D0 read and report port 5 in decimal,
C0 READ AND REPORT PORT 5 IN DECIMAL
B0 read both, report port 5 first in hex, port 6 second in decimal
A0 READ AND REPORT BOTH IN IN HEX (port 5 is first output, port 6 is second)
90 read and report Port 5 in hex
80 READ AND REPORT PORT 5 IN HEX
70 read and report PORT 6 in decimal
60 read and report PORT 6 in hex
50 read nothing!
40 read nothing!
30 READ AND REPORT PORT 6 IN DECIMAL
20 READ AND REPORT PORT 6 IN HEX
10 read nothing!
00 read nothing!

****
*
*
* Addendum 5: Analog and Digital Channel definitions *
*
*
****

The following analog signals are used in the ASRC-built instrument #102 as deployed at SGP mid June, 2001. Analog Channel # are 0..31; the instrument is built with a standard MFRSR logger-board as the RSSmicro controller and the low-order 16 channels (0..15) are not used for RSS purposes.

********** On the ASRC RSS MICRO
****

16 nc
17 BAND-BOARD Electric box TEMP
18 OUT SIDE AIR TEMP
19 Optics Enclosure TEMP
20 Snorkel TEMP
21 THERMO-ELECTRIC COOLER TEMP
22 PRISM TEMP
23 SLIT TEMP
24 THERMISTOR EXITATION
25 INTERNAL OPTICS BOX PRESSURE
26 INTERNAL OPTICS BOX TEMP
27 INTERNAL OPTICS BOX HUMIDITY
28 nc
29 nc
30 nc
31 nc

The "INTERNAL" signals are the new signals added with this revision.

For the ASRC-built instrument output port bits are at a premium, and so the output pins for the Portable calibrator come from PORT5 and are the low-order four bits. Here are the individual bit definitions, which users should not need, for the Portable calibrator control lines:

```
PCLhton = $01        ! portable calibrator heat-on control output
PCLlpon = $02        ! portable calibrator lamp-on control output
PCLrfon = $04        ! portable calibrator reference-on output
PCLidck = $08        ! portable calibrator ID clock line.
```

Note that the other four pins of PORT5 and all of PORT6 are used for the PIO interface between the two CPUs so there are no free pins on these ports. On Yankee's "combo board" these bits have the same logical order and function, but are on the EXPANSION output port.

Similarly users should not need to know the portable calibrator inputs, but they are listed here for reference. These come to the DIGIN port (which also receives home-detecct signals). Note Yankee's are not quite the same.

```
H1bit  = $80    ! bitmask for home-detect 1 input
H2bit  = $40    ! bitmask for home-detect 2 input
```

```
PCLhere = $08           ! bitmask for port. cal. plug attached
PCLtpgd = $04           ! bitmask for portable calibrator Temp-good line
PCLlpok = $02     ! bitmask for portable calibrator Lamp-OK line
PCLidda = $01     ! bitmask for portable calibrator ID_data line
```

********** On the CCD MICRO **********

The following bits are used in USLATCH on the CCD micro. The user can directly set the "open shutter" state of the USLATCH (through either the RSS interface or the CCDmicro interface), the most common reason to do so is to do a "flat-field" test, but other uses include providing a separate drive line for cosine-bench signal acquisition, etc. Here are the pin/signal bitmasks. Note that the user command ALWAYs ORs your data with the NRCbit, the open-shutter state ALWAYs disables the shutter-driver recharge:

```
NRCbit  = $80  ! no-recharge -- this bit disables the switchmode
SHUTbit = $40  ! Shutter open control
FFLDbit = $02  ! flat-field bit illuminator on ASRC instrument
```

In addition YANKEE's DSI system requires that the CCD readout A/D calibration strobe be moved, and it is on USLATCH too as:

```
CALbit  = $20 ! this is the new CAL line, YANKEE ONLY
```