RCP02 V10 Release Notes

These notes cover changes made to the RCP02 code since release V09 of 23 June 1998. If you are upgrading from an earlier release, please read those notes also.

Bug Repairs

1. The printouts of ”Site Custom” and ”Help View” have been added to the overall listing from ”Help Listall”. These had been missing before.

New Features

1. The ”Help View” command now includes the version of code that last saved the nonvolatile setups.
2. The “Monitor Status” command has been improved so that the hardware input lines can be monitored independently of the actual internal status. Also, the condition of the auxiliary status input lines can also be viewed.

```
RCP> monitor status
Hardware Electrical Inputs
Locl Pw1 Pw0 Rad Srv T/R Stby Intr Mag Air WGp Res ElLO ElHI IRIS
---- --- --- --- --- --- ---- ---- --- --- --- --- ---- ---- ----
 --   --  --  --  --  --  --   --       On  --  --  --   --   ---
```

   The characters “—” will be printed under each status input that is not being used. For the used inputs, the word “ON” will be printed if the line is asserted, and blank space will appear if the line is not asserted.

   You may switch to the following alternate presentation by typing “alt”. Now, the internal status of each condition is displayed. This is different from the condition of the hardware input line in that the status may be coming from another source, or may be spoofed from the requested control.

```
RCP Internal Status
Locl Pw1 Pw0 Rad Srv T/R Stby Intr Mag Air WGp Res ElLO ElHI IRIS
---- --- --- --- --- --- ---- ---- --- --- --- --- ---- ---- ----
                  ON                                          000
```

   Lastly, if auxiliary status lines have been enabled, then you may switch to the following bit presentation by typing “alt”. In the following example, four bytes of optional status have been selected via the “site custom” menu. High inputs are shown as a “1”, and low inputs are shown as a “.” (rather than as “0”, to make the string more readable at a glance).

```
Auxiliary Status Inputs
S[31:24] S[23:16] S[15:08] S[07:00]
-------- -------- -------- --------
11111111 11111111 11111111 1111111.
```

3. The “Monitor Control” command has been improved so that the externally requested control functions can be viewed independently of the final logic-qualified control. Also, the auxiliary output control lines can now be monitored.

The control functions that have been externally requested (usually from the host computer) are shown in this display. If auxiliary control lines have been enabled in the "site custom" menu, then those numbered bits appear at the end of the printed line.

```
Externally Requested Control
Pw1 Pw0 Rad Srv T/R Res IRIS C[31:24] C[23:16] C[15:08] C[07:00]
--- --- --- --- --- --- ---- -------- -------- -------- --------
             ON          000 ........ ........ ........ ........
```

The qualified state of each control function can be viewed by typing "alt". The display now shows the actual control state, which may be different from the requested state if any internal logic equations are overriding the request. See the "Control Logic" command for a description of logic equations.

```
Qualified Control & Electrical Outputs
Pw1 Pw0 Rad Srv T/R Res IRIS C[31:24] C[23:16] C[15:08] C[07:00]
--- --- --- --- --- --- ---- -------- -------- -------- --------
             ON          000 ........ ........ ........ ........
```

Lastly, the sixteen local logic variables are shown in the following "alt" display.

```
Local Variables
V[15:08] V[07:00]
-------- --------
........ ........
```

4. Because of the improvements to the "Monitor Control" command, the control bit monitoring that used to appear in the "Monitor SIO" command has been removed.

Setup Changes

1. A major new setup section has been added in which custom logic equations can be defined to modify the RCP02 control bits. The "Control" command now takes an argument — use "Control Lines" to setup the hardware lines themselves (just like it used to), and "Control Logic" to access the new equation editor. The description of the equation editor is given below.

   The questions are actually a series of up to 32 logic equations that allow the user to qualify the RCP02 control functions in a very general way. Control bits can be modified according to any logical combination of control bits, status bits, and special internal local variables. The result is that the RCP02 can be programmed to perform custom safeguards and implement special feedbacks that are specific to each site.

   Each equation consists of a control variable on the left that is assigned to some logical combination of variables on the right. The syntax is that of an ordinary "C" language statement using the operators "&" for AND, "|" for OR, and "!" for NOT. Parentheses may be included anywhere, as needed. New equations and subcommands are typed following the special "—>" prompt that is printed below the current equation. An example of how a few logic equations might appear is shown below.

```
EQ00: v0 = airflow | wavegp | magcur
EQ01: cservo = cservo & !v0
EQ02: relay = relay | c10
EQ03: ctrpower = true
  -->
```

In the above example, EQ00 assigns the internal local variable "V0" to the logical OR of the "airflow", "waveguide pressure" and "magnetron current" status bits. V0 will be TRUE whenever any of those status lines are true. The second equation then uses a TRUE sense of V0 to force the "servo power" control line FALSE. Thus, servo power will only be on if it is requested to be on, and if none of the three control lines are asserted. Likewise, EQ02 allows the external local/remote relay to be forced on by the auxiliary control line C10. Finally, EQ03 forces "T/R power" to be on all the time, regardless of any other conditions.

The rules of constructing equations are as follows.

- The left side of the equation must consist of one of the control variables: cpw0, cpw1, cradiate, cservo, ctrpower, creset, ciris0, ciris1, ciris2, ngen_on, sgen_on, sgen_cw, relay, v[0:15], c[0:63]. Control variables that have status counterparts are prefixed with the letter "c". Thus, "cradiate" is the request to radiate, whereas "sradiate" is the detected radiate status. The numbered local variables V[0:15] can be used as temporary storage for subexpressions, and the numbered control lines C[0:63] refer to the optional auxiliary control outputs.
- The right side of the equation must consist of some logical combination of any of the above control variables, plus any of the status variables: spw0, spw1, sradiate, sservo, strpower, sreset, siris0, siris1, siris2, local, standby, ilock, magcur, airflow, wavegp, elimlo, elimhi, ngen_on, sgen_on, sgen_cw, sgen_flt, trlocal, s[0:63]. The numbered status variables S[0:63] refer to the optional auxiliary status input lines.
- The special symbols TRUE and FALSE may be used on the right side of an equation to indicate "always" and "never".
- The maximum number of different variables on the right side of an equation is four. If you need to combine more than four variables into an expression, then make use of multiple equations using local variables to combine all of the bits together. For example, to force servo power off whenever any of the first eight auxiliary status lines is high:

  EQ00 : V0 = S0 | S1 | S2 | S3
  EQ01 : V1 = S4 | S5 | S6 | S7
  EQ02 : Cservo = Cservo & !(V0 | V1)

- Control variables that appear on the right side of an equation refer to the requested control state, whereas those appearing on the left-hand-side reflect the final qualified state. Thus, the equation "cservo = cservo & !v0" causes the requested servo state to be AND'ed with the negation of local variable V0, and the result is the actual servo state. This convention is very handy, and allows you

to use control variables on the right without ambiguity. For example, the pair of equations: “EQ00: cpw0 = cpw1” and “EQ01: cpw1 = cpw0”, would swap the two pulsewidth control bits.

- A side effect of making assignments to local variables is that these bits are also reported to the host computer via the RCP02’s Internal BITE packet. This opens many possibilities for designing customized status bits that could be monitored and reported by the IRIS/Open BITEX utility.
- Order of evaluation of each equation is from right-to-left. There is no evaluation precedence among the “&”, “|”, and “!” operators. Thus, “!V0 & V1” means “!(V0 & V1)”, and not “(!V0) & V1”. Be sure to use parentheses as needed to express your equations properly.
- Order of evaluation of the set of equations is from EQ00 to EQ31. The order is important only when local variables are assigned in earlier equations so that they can be used in later equations.

When a new equation is entered, the RCP02 immediately parses the input text and converts the right-hand-side into an internal representation that can be evaluated efficiently at run time. The original line of text is discarded. Thus, when the equation is redisplayed, the RCP02 must recreate a printed line of text from this internal compiled representation. As a result, the equation that is echoed back will not always look the same as the equation that was originally typed. It will be logically equivalent, of course, but the exact syntax may be different. Some examples follow:

```
  --> v0 = v1 & v2 & !v3
EQ00: v0 = v1 & v2 & !v3
```

Here, the output representation happens to be identical to the original input.

```
  --> v0 = v1 & (v2 | v3)
EQ00: v0 = (v1 & v2) | (v1 & v3)
```

In this case, the equation is printed as an expanded version of the original. Note that the Boolean operators “&” and “|” both distribute symmetriclally over each other, so that logical expressions can be factored and expanded over either operator.

```
  --> v0 = (v1 & v2) | (v1 & !v2)
EQ00: v0 = v1
```

Here, what appears to be a function of two variables is really only dependent on one.

```
  --> v0 = (!v1) & (!v2)
EQ00: v0 = !(v1 | v2)
```

In this case, the DeMorgan equivalent of the “AND of two negations” is printed as the “negation of two OR’s”. This type of transformation will often be observed, since the RCP02 attempts to minimize the number of “!” operators in its synthesized expressions.

The following list of subcommands is printed at the beginning of the equation list.

```
Subcommands
   Del – Delete text of current equation
   Ins – Insert free slot before current equation
  Pack – Pack equations to consolidate free slots
     ! – Negate logic sense of equation
  ?<v> – Additional help
```

The first three subcommands are used to move and edit the list of equations. "Del" deletes the text of the current equation so that the line is blank. "Ins" inserts a blank equation at the current location by shifting the current equation plus all subsequent equations ahead by one. "Pack" removes blank equations and shifts all equations into the lower numbered slots.

The "!" subcommand replaces the current equation with its logical negation. Some examples are:

```
EQ00: v0 = v1 & v2
  --> !
EQ00: v0 = !(v1 & v2)


EQ00: v0 = v1 & !v2
  --> !
EQ00: v0 = (!v1) | v2
```

Note that DeMorgan's theorem was used to reprint the second of these two examples, because doing so removes an extra "!" from the equation. Perhaps the simplest equation to negate is:

```
EQ00: cservo = true
  --> !
EQ00: cservo = false
```

Here, the output variable is forced TRUE/FALSE each time the "!" subcommand is used. This can be very helpful when testing individual control lines for simple ON/OFF response.

Before the actual equation list appears (in response to the "Control Logic" command), the following initial question is asked:

Enable logic override of control lines: NO New Value:

This first question allows the entire set of equations to easily be switched ON/OFF, without having to change any of the equations themselves. Answering "NO" will leave the control functions unmodified (direct control from the host computer); whereas "YES" will apply all of the logical constraints.

Unlike all other RCP02 setup menus, the logic equation editor is a live menu. This means that each equation becomes active as soon as it is typed in. It then becomes easy to test individual control lines, and to edit the set of equations until the desired effects are obtained. Also, the "!" subcommand becomes a simple shortcut to perform a quick ON/OFF toggle test of any control line.