RCP02 V14 Release Notes

These notes cover changes made to the RCP02 code since release V13 of 2 September 1998. If you are upgrading from an earlier release, please read those notes also.

Bug Repairs

1. When an HPIB signal generator is not installed on the RCP02, the signal generator status sent back to the host computer will be spoofed from whatever siggen settings the host computer is currently requesting. Thus, the ”RF–Level”, ”On/Off”, and ”Cont/Pulse” status are all echoed back, and the ”Fault” status is FALSE (no fault). Previously, the HPIB signal generator status was always being reported, even when the RCP02 was not configured to use an HPIB siggen.

New Features

1. A new status variable ”antstop” has been added for use with logic equations. This variable is set TRUE whenever the antenna seems to be stopped, i.e., has been moving slower than a prescribed speed for more than a prescribed time. There are two new setup questions in the ”control variables” section to configure these speed and time thresholds. Both the AZ and EL axes must appear to be stopped in order for ”antstop” to be TRUE. Likewise, ”antstop” is set FALSE whenever the antenna seems to be moving, i.e., the speed on either axis has exceeded the threshold speed for more than the specified time.

2. A collection of software timer variables are now supported for use with logic equations. Eight new control variables have been added whose names have the generic form “`t`*`n`*`_mode_`*`time`*”, depending on how each timer has been configured. For example, if timer #3 is configured to be a retriggerable pulse generator with a period of 2.5 seconds, then the variable “`t3_retrig_2.5`” would appear in the control variable list. You could abbreviate the typed-in name to just “t3”, but the full mode and time will be echoed in each equation so that the exact behavior of the timer variables is clear at a glance.

   Timer variables can appear on both the left and right sides of logic equations. On the right they act as normal Boolean variables having TRUE/FALSE values that can be used in any logic equation. However, when they appear on the left, the value being assigned from the right-hand side acts as an input trigger to the timer. The timer’s response to this input can take several forms, depending on the mode that has been selected. The available modes are:

   - **Retriggerable Pulse Generator** (”retrig”)
     This timer generates a TRUE pulse whenever a FALSE→TRUE (rising edge) transition is applied to its input. Each rising edge continues to retrigger the output pulse, i.e., a fresh pulse period is begun each time. For example, if a rising edge were presented once per second to the timer “`t0_retrig_1.5`”, then the timer output would be a steady TRUE value. Since the 1.5-second

timeout begins anew once per second, the output pulse will never actually end. "Retrig" timers are handy for keeping track of whether any FALSE→TRUE transitions have occurred (perhaps irregularly) over a given period of time.

- **Single Pulse Generator** ("single")
  This timer generates a pulse similar to "retrig", except that an active pulse will not be retriggered by additional input transitions. For example, if a rising edge were presented once per second to the timer "`t0_single_1.5`", then the timer output would be a rectangular wave that is TRUE for 1.5 seconds and FALSE for 0.5 seconds. The 1.5-second TRUE pulse is first triggered by an input edge. One second later the timer is still active, so the next input edge is ignored. The pulse finally ends 1.5-seconds later, remains FALSE for 0.5 second, and then is triggered again by the next rising input edge.

  The active-low application of "single" is also useful, as in the following two equations which prevent "radiate" from being switched back on within 60 seconds of it being switched off. Note that if a "retrig" timer were used here, then repeated attempts at radiating would keep resetting the 60-second interval even though the transmitter had never actually turned back on.

```
EQ00: t0_single_60 = !cradiate
EQ01: cradiate = cradiate & !t0_single_60
```

- **Delay Line Filter/Follower** ("filter")
  This output of this timer attempts to follow its input, but with filtering and delay effects added. Whenever a TRUE input is presented, an internal counter begins counting up until the timer period is reached. At that point the timer's output is set TRUE. Likewise, a FALSE input causes the counter to decrement until reaching zero, at which point the timer's output is set FALSE. The net result is that the output follows the mean value of the input, and thus, a "filter" timer can be used to clean up a noisy logic signal, or combination of logic conditions.
- **Leading Edge Retard** ("retard")
  The output of this timer attempts to follow its input, except that rising input edges are delayed by the timer period, whereas falling input edges are passed through immediately. The result is that the leading edges of the input signal are delayed, but the falling edges are not. A "retard" timer is useful when one wants to delay only the onset portion of a signal, e.g., to holdoff transmitting for a few seconds after a radiate request has been made. It is also useful when filtering signals to remove short spurious TRUE inputs in which, contrasted with the "filter" timer, an instant-off effect is also required.
- **Trailing Edge Extend** ("extend")
  This timer is the counterpart to "retard", in that the falling input edge is extended by the timer period and the rising input edge is passed immediately. An "extend" timer forces a minimum time during which the timer output will be TRUE in response to any (possibly momentary) TRUE input. It is useful for stretching a

short input condition out to at least some minimum time, or for adding additional "hold time" to the end of a signal. Note that the output of an "extend" timer is logically equivalent to the negation of the output of a "retard" timer whose input is also negated.

- **Periodic Clock Oscillator** ("clock")
  This timer produces a free-running clock having a specified period. The length of the timer's TRUE interval (and hence, the duty cycle) is also adjustable. The "clock" timers usually appear on the right side of equations, where they can supply any periodic input that the logic might require, e.g., to make a light blink, or to perform a periodic reset. But their phase can also be resynchronized to the start of their TRUE output interval by the application of a rising input edge.

3. A printout of the eight timer states has been added to the "Local Variables" section of the "Monitor Control" command.

4. There is a new status variable "unsafe" for use with logic equations. It is TRUE if the RCP02 is in the temporary unsafe mode following a "reset" command.

5. There are two new status variables "ovel_az" and "ovel_el" for use with logic equations. These variables are TRUE whenever the velocity on the corresponding axis exceeds the maximum value setup from the "velocity" command.

6. The RCP02 "reset" command now places the controller into its momentary "unsafe" condition regardless of whether the RCP02 is shutdown at the time the command is received. Previously, "reset" would bring about "unsafe" only if a real shutdown was present. This change allows the reset command to be useful when attempting to exit from stuck conditions; including those times when the RCP02 has not actually shutdown.

7. When the optional Andrew–Kintec pedestal option is selected, the elevation low and high limit switch internal status variables are now set individually according to 1) the limit switch indication from the ACU, and 2) whether the elevation angle is below or above 45 degrees. Previously the two status variables were set the same, i.e., just from the ACU limit switch indication, which does not distinguish between the upper and lower limits.

Setup Changes

1. A new setup section "control variables" has been added. This collection of questions will be used to configure certain variables that can appear in Boolean logic equations.

2. Two new questions appear in the "control variables" section to configure the "antstop" variable (See New Feature #1.).

```
Minimum velocity for 'antstop':  0.50 deg/sec
Minimum time for 'antstop':  2.00 sec
```

3. Sixteen new questions appear in the "control variables" section to configure the eight internal timer variables (See New Feature #2.). The choices for timer mode are "retrig", "single", "filter", "retard", "extend", and "clock".

```
Timer #0 trigger mode: Retrig
Timer #0 period/delay:   1.0 sec
Timer #1 trigger mode: Retrig
Timer #1 period/delay:   1.0 sec
Timer #2 trigger mode: Retrig
Timer #2 period/delay:   1.0 sec
Timer #3 trigger mode: Retrig
Timer #3 period/delay:   1.0 sec
Timer #4 trigger mode: Retrig
Timer #4 period/delay:   1.0 sec
Timer #5 trigger mode: Retrig
Timer #5 period/delay:   1.0 sec
Timer #6 trigger mode: Retrig
Timer #6 period/delay:   1.0 sec
```

Note that when a timer is configured for periodic "clock" mode, an additional question will appear to set the duration of the TRUE time interval. The period is still set using the standard questions shown above.

4. The status and control variable lists that are printed by "?v" from within the "control logic" command now appear in alphabetical order. Making the lists easier to scan has become important as more and more variable names are being added.

Examples

1. The following example shows how to implement a custom logic requirement using timer variables. Suppose that an operational site requires that the radar transmitter be switched off whenever the antenna is not rotating. This is fine for normal operation, but during maintenance periods there must also be a procedure to allow transmitting while stopped. When such an override is requested, an audible warning and flashing light must first occur for 20 seconds; only then does the override actually take effect. At that point the horn becomes silent, but the warning light must continue to flash for the duration of the override.

   An RCP02 external status input line "S0" will be used to request the override. Assume that control line "C0" activates the horn, and that "C1" activates the warning light. The necessary equations are:

```
EQ00: v0 = cradiate & antstop & s0
EQ01: t0_retard_20 = v0
EQ02: c0 = v0 & !t0_retard_20
EQ03: c1 = v0 & t1_clock_1.5
EQ04: cradiate = (cradiate & !antstop) | t0_retard_20
```

   EQ00 assigns local variable "V0" as a qualified override request. V0 will be TRUE when there is a request to radiate while stopped, and when the external override request line is also TRUE. EQ03 combines this condition with a 1.5-second periodic clock to produce the flashing light. Meanwhile, EQ01 passes V0 through a 20-second "retard"

timer. When the timer output eventually becomes TRUE, EQ04 allows the transmitter to radiate even though the antenna is stopped. Meanwhile, EQ02 sounds the horn only during the timer's initial 20-second delay period.

As soon as the antenna starts moving, V0 and the timer output immediately become FALSE. The horn and light are extinguished right away, and the override input is ignored. The first "&" expression in EQ04 then allows the transmitter to be controlled in the normal On/Off manner.

2. Logic equations can help supply the necessary control signals for backing out of stuck situations. In this example, an antenna servo power unit requires override signals to move away from the low and high elevation physical limit switches. Assume that "C0" and "C1" enable motion in the up and down directions. The following equations will allow the "reset" command to activate these lines briefly:

```
EQ00: c0 = antstop & unsafe & elimlo
EQ01: c1 = antstop & unsafe & elimhi
```

The "unsafe" status variable is TRUE for a short interval of time following an RCP02 reset. Resets from the host computer serial port always give a 1.0-second unsafe interval, whereas those from the RCP02 command line take the number of seconds as an argument, e.g., "reset 2.5". The "antstop" test is added as an additional safeguard to insure that the antenna is motionless when the override is attempted.

3. Logic equations can be used to supply the host computer with BITE information that would nor ordinarily be available. The trick is to make an assignment to one of the first fourteen local variables, as those will then be transmitted via the RCP02's Internal BITE Packet. For example, adding the equation:

```
EQ00: v13 = ovel_az | ovel_el
```

will send a "velocity overspeed" bit to the host computer in Bit #6 of Byte #12 (See RCP02 User's Manual, Appendix C, Table C-10, for the mapping of the local variable bits).