NTF Beamline Controls

Micro-P Archaeology

Robert Goodwin

Mon, Apr 19, 2010

The Beamline Micro-P serves to monitor controls related to the Neutron Therapy Facility during treatment, including terminating operation when the prescribed radiation dose is reached. This note is meant to describe the logic of that system's operation, determined through archaeological study of the source code.

15 Hz Interrupt Activity, INT15

Almost all activity is executed as interrupt code. Each 15 Hz cycle, an interrupt occurs shortly (750 μs) before the 50 μs Linac 66 MeV beam pulse strikes the target to produce the neutrons that are used for treatment. Besides the work done within this interrupt code, smaller bits of logic occur at the three intermediate 60 Hz interrupts between successive 15 Hz interrupts.

In the 15 Hz interrupt code, the elapsed time since the previous interrupt is calculated and stored for diagnostic reference. It is in units of μs stored mod 2^{16}, so that the nominal value should be (66666 - 65536) = 1130 μs. (This is Acnet device `L:CDTIME`.)

A check is made that the 60 Hz interrupts are operating normally, meaning that a counter that is incremented in the 60 Hz interrupt code should be 3. If not, a LED is turned on.

The zero data (pedestal) is read for the ion chamber integrators and for the beam charge integrator. Note that this happens before the beam pulse. This is done by the routine `ICX1X2`, which calls `RDIFF` to read the 8-channel differential A/D and also digitizes the single channel (#11) from the 16-channel A/D that is connected to the Toroid #2 beam charge signal. It is summations of these three signals that determine `X1`, `X2` and `QP`.

The next job is to read the pressure and temperature signals via a special digital interface. But there is nothing connected via that interface, so the results are meaningless. The temperature and pressure are read via two A/D channels. Its readings are used only by humans; they are not needed by the Beamline μP to support patient treatment.

The digital multiplexer is read to access some gray code values that were once used for assessing the chair position. But these are no longer used; the chair position is obtained in a different way. (Again, the Beamline μP itself does not need to know chair positions.)

Next is a delay of 800 μs, in order to allow time for the beam pulse to occur and for the integrators to accumulate their signals. Then the 16-channel A/D is read via `RDAD`, and the 8-channel differential A/D is also read via `RDIFF` again. (Also read safety system status bits.)

Armed with values of the integrators both before and after the beam pulse, the differences are calculated by `DELTA` and accumulated into 16-bit sums. These mini-sums are then used by `ACC` to accumulate doses over multiple cycles according to the `#ON` pulses specified for the treatment. (Both `#ON` pulses and `#OFF` pulses can be specified to allow for control of the duty cycle, although in practice, they are nearly always set for `#ON` = 5 and `#OFF` = 0.) Because the accumulations of the delta values are done as 12 bits into a 16-bit sum, it is advisable to limit the `#ON` pulses to 16. The other significance of the `#ON` pulses is that only at the end of the `#ON` pulses is the 16-bit accumulation added into a wider accumulation and checked against the prescribed dose in order to stop the treatment when that dose is reached.

The next routine that is called is `ICHECK`, which checks for various problem conditions, any of which might stop the treatment. (More about this later.)

Call `COMP` to check the readings of the two bend magnets against their nominal values and specified tolerances.

Call `MPENBL` to operate the Micro-P Enable control bit logic. (More on this later.)

Copy the constants from PROM that are used by `ICHECK` as references for the `QP/X1` test and the `X2/X1` test, as well as those used for the power supply voltage checks.

Copy 256 bytes of variables data to shared memory so the other processor (`061B`) in the same VME crate can access it for passing to `node061C` via arcnet so that Acnet can see it.

This ends the 15 Hz interrupt routine. Its execution time is about 2.5 ms.

60 Hz Interrupt Activity SIXTY

This routine is made to run first after a delay of 16 ms from the 15 Hz interrupt code. When that interrupt occurs, a second interrupt is made to run after a further delay of 16 ms. Finally, a third interrupt is made to run after another 16 ms, or later when needed to verify that the bend magnets have reached full field before allowing NTF beam. The special case is specified via `C58DLY`, when its value is in the 40–64 range, which is interpreted as milliseconds since the 15 Hz interrupt to schedule the third 60 Hz interrupt. If it is not in that range, it is treated as a delay in 15 Hz cycles to allow for the 58 degree bend magnet to ramp up.

Interlocks check, ICHECK

Check that Treatment Enable status is true. If not, skip the following beam toroid checks.

Check the `TOR1` beam current, which is `TO4OUT`, to be above `0x0400`, which is 3.2 ma. (The comment says 14.5 ma, but this must be old scaling.) If it is, then use `TOR2`, the beam toroid before the target, to obtain the ratio `TOR2/TOR1`, and require it to be more than 0.68, in which case it is counted as a “good” beam pulse. After 15 beam pulses have been checked in this way, require that 8 of them have been “good” pulses; otherwise, stop the treatment.

But if `TOR1` was not large enough, then check `TOR2` to be above 3.2 ma. If it is, we have a strange situation whereby the second toroid shows something but the first does not, so consider this a “bad” pulse. If both toroids were small, however, simply ignore this beam pulse; pretend it never happened.

After dealing with the beam toroid readings, check whether this pulse is the last in a series of `ON` pulses. If it is, then there is more checking to do; otherwise, exit.

Check that there has been at least one `ON` pulse scheduled since the last NTF Reset. If not, exit.

Call `TCHK` to test dose thresholds to see whether treatment should be ended.

Call `RCHK` to check ratios `QP/X1` and `X2/X1` to see whether treatment should be stopped.

If any condition has occurred to warrant it, assert control line to stop treatment. After this, an NTF Reset is required to start treatment again.

ON/OFF Pulse sequencing MPENBL

What `MPENBL` does is to arrange for `#ON/#OFF` sequencing of beam pulses while PS Ramp Enable status is on. It starts with a `#OFF` cycles based on the `D58DLY` value, then schedules `#ON` cycles, followed by `#OFF` cycles, *etc*, until PS Ramp Enable status is off.

Two variables are used, the stage number `MPES` and the counter `MPEC`. The stage number can be 0, 1 (`OFF` cycles), or 2 (`ON` cycles). The counter is used to count down either `#ON` or `#OFF`.

Check the Power Supply Ramp enable status for change. If it just dropped off, call `MPECLR` to clear the Micro-P Enable control line, and then set `MPES` = 0, `MPEC` = 1. But if it just turned on, then examine `D58DLY`. (If it is zero or negative, or > 64, assume it is 64.) If it is >= 40, then set `MPES` = 2 and `MPEC` = `#ON`, except if `#ON` = 0, call `MPECLR` and set `MPES` = 0, `MPEC` = 1.

If the PS Ramp Enable status did `not` change from the last cycle, and it is still off, call `MPECLR`, set `MPES` = 0, `MPEC` = 1. But if the status is still on, examine `MPEC`. If it is zero, then call `MPECLR`. Then, if `MPES` = 2, then set `MPES` = 1, `MPEC` = `#OFF`, except that if `#OFF` = 0, set `MPES` = 2, `MPEC` = `#ON`. (If both `#ON` and `#OFF` = 0, then call `MPECLR`, set `MPEC` = 0 , `MPEC` = 1.)

Backing up a bit, if PS Ramp Enable status is still on, and `MPEC` is zero, but `MPES` is *not* 2, set `MPES` = 2, `MPEC` = `#ON`, except if `#ON` = 0, set `MPES` = 0, `MPEC` = 1.

Finally, before exit, decrement `MPEC`.

Message Queue Handler DOMSGQ

The Beamline Micro-P does not do networking in the normal sense, but it accepts commands via a message queue in shared memory. (The same shared memory is also used to pass a copy of the data pool to the coprocessor residing in the same VME crate.) The coprocessor `node061B` runs the normal Linac-style front end software, but it does not normally connect to the ethernet used by other front ends. It copies parts of the shared memory data pool into its own data pool every 15 Hz cycle. It runs a special local application called `SRMD` that emulates an SRM (Smart Rack Monitor) in supporting the SRM protocol used over arcnet. The NTF front end, `node061C`, sends out a request for SRM data via arcnet every 15 Hz cycle, the same as do most other Linac front ends. One SRM request actually reaches `node061B` and is passed to `SRMD` for handling. Then `SRMD` recognizes the request message, formats some data from its own data pool that includes data from the memory shared by the Beamline Micro-P, and returns it as a reply message via arcnet, just as do actual SRMs.

To support commands, SRM protocol setting messages sent via arcnet from `node061C` are interpreted and passed on via a message queue in the shared memory that is accessible by the Beamline Micro-P, which then interprets such commands that it finds in the message queue to accomplish the specified setting action. Such actions include establishing the `#ON`, `#OFF`, `C58DLY` parameters, the bend magnets nominal and tolerance values, the dose limits, a special integrator gate used to calibrate the NTF beam, and a special message to effect an NTF Reset.