LOOPPHAS Logic Flow

Linac local application

Mon, Dec 22, 2008

Here is part of the source code for LOOPPHAS.c, with a description of its meaning.

```
if (trig == cycle && par->SA != NULL)
     {
     WITH = par->SA; /*cycle*/
     WITH->gradient = ChanRaw(par->Grad_CN, C_read);
     WITH->phase = ChanRaw(par->Intph_CN, C_read);
     WITH->phase_nom = ChanRaw(par->Intph_CN, C_nominal);

     if (BitRaw(par->Beam_BN) == Bit_set)
          {   /*=1, NO BEAM*/
          WITH->Nobeam_cnt++;
          if (WITH->Nobeam_cnt >= 4500)
               {  /*over 300 sec. without beam - clear average calculations*/
               WITH->Nobeam_cnt = 0;
               WITH->Sum = 0;   /*=0.0*/
               WITH->Avr_cnt = 0;
               }
          }

     else if ((par->Inhib_BN == 0) | (BitRaw(par->Inhib_BN) == Bit_set))
          {  /*Beam was on!*/
          WITH->Nobeam_cnt = 0;
          if (WITH->gradient < WITH->Gradlow)
               {   /*low gradient, restart averaging*/
               WITH->Sum = 0;   /*=0.0*/
               WITH->Avr_cnt = 0;
               }
          else
               {
               WITH->Sum += WITH->phase;
               WITH->Avr_cnt++;
               if (WITH->Avr_cnt == Pul_in_avr)
                    {
                    WITH->err = WITH->phase_nom - WITH->Sum / WITH->Avr_cnt;

                    WITH->adj = Min_adj;
                    if (par->InvFlag & 1)
                         WITH->adj = -WITH->adj; /* inverse flag set */

                    if (WITH->err < 0)
                         {
                         WITH->adj = -WITH->adj;   /*reverse*/
                         WITH->err = -WITH->err;   /*abs (err)*/
                         }

                    if (WITH->err > WITH->Threshold)   /*adjustment required*/
                         {  /*SetD for the required adjustment*/
                         cni.node = ThisNode();
                         cni.chan = par->Phadj_CN;
                         ltype = noSw + 7;   /*relative setting*/
                         nbytes = 2;
                         idlength = 4;
```

```
                    SetD(ltype, nbytes, idlength, (USHORT *)&cni.node,
                                   (UBYTE *)&WITH->adj);

                    datePtr = (SByte *)DATETIME; /* record time of adjustment */
                    dest = WITH->adjTime[WITH->index];
                    for (i = 0; i < 7; i++)
                         *dest++ = *datePtr++;  /*yr-mo-da-hr-mn-sc-cy */
                    *dest = WITH->adj;  /* place #steps in last byte of time */
                    WITH->index += 1;
                    if (WITH->index >= maxTimes)
                         WITH->index = 0;

                    }
               WITH->Sum = 0;   /*start average count again*/
               WITH->Avr_cnt = 0;
               /*gradient within range, average phase*/
               }
          }
     }  /*beam on*/
}
```

The local application PHAS is called every 15 Hz cycle. Here is the logic flow:

Sample the current cycle values of gradient reading, intertank phase reading, and intertank phase nominal value.

If the beam status is “1”, there is no beam. If there has been no beam (determined by a clock event) for 5 minutes, then forget any average accumulation.

If the beam status is “0”, the present cycle is scheduled to accelerate beam. (Whether there is actual beam is not checked.) Proceed as follows:

If the optional inhibit regulation Bit# parameter is nonzero, and its “inhibit” status is “0”, it means that regulation of phase is to be inhibited. (This optional parameter may never have been used; i.e., its parameter field is zero in all nodes 0612-0615. The result is that regulation is never inhibited for this reason.)

If the gradient is less than 0.8 volts, forget about any average accumulation so that when normal gradient is restored, we will start over.

Add the current intertank phase reading into a sum. If this makes a total of 8 readings in the sum, compute the average as sum / 8. Calculate the current phase error, which is the nominal intertank phase minus the average.

Prepare to make a possible adjustment to the phase. The amount of adjustment is always 3 steps of stepping motor control. Normally, if the error is positive, the adjustment will be to increase the phase adjust; otherwise, it will decrease the phase adjust. (With the new parameter set to 0001, this logic is reversed.) In any case, the adjustment only takes place if the absolute value of the error is more than the threshold of 0.010 volts. Log any adjustment internally.