

# Parameter Page Notes

*Comments on source review*

Robert Goodwin

Mon, Jan 5, 2009

## CopySw

Monitors changes in console lights

`vMLites`, `vULites` are constants that indicate which lites are active in `PARM`.

`MMASK`, `UMASK` are globals the system monitors to light up lites as switches are pressed.

Switches without corresponding bits set do not activate lights.

`MLITES`, `ULITES` are globals that hold the bits of currently active lites.

Upon entry to the page, if no lites are on, `PARM` sets A/D mode, Eng units.

`dispMode`, `dispUnits` are enum variables that guide `PARM` screen update logic.

The bit assignments for the two bytes of lites are:

Bit	<code>dispMode</code>	<code>dispUnits</code>
7	n/a	n/a
6	A/D	Eng
5	D/A	Volts
4	Nom	Hex
3	Tol	Raise
2	Set	Lower
1	KeySwitch	Left/Right
0	Kbd Int	Up/Down

## LToI

Converts 32-bit long to 16-bit integer, clamping to -32767 to +32767 range.

## DefineLists

Issue data requests for ongoing data that `PARM` displays. Every time the list of up-to-14 channels changes, this function is called to update the requests. Three requests are made, all based upon the current list of channels. One is at 15 Hz, for readings and settings. A second is at a 13 cycle period, for alarm info (nominals, tolerances, flags) and the associated status bits. The third is a one-shot for the descriptors for each channel. Note that the underlying system code takes care of contacting the relevant nodes involved; each channel reference includes the node where it is based.

## SetSetLog

Build setting log record and write to `SETLOG` data stream via `DSWrite`. This record is of a special format that includes a device name and thereby provokes attention by local application `SLOG`, which delivers the info to Acnet as part of "settings accountability" support.

## SetChanData

Makes an integer setting via `setD`. Calls `SetSetLog` to record `SETLOG`.

## SetChanFData

Makes a float setting via `setD`. Calls `SetSetLog` to record in `SETLOG`.

## Update

Major function to update screen for a given row. Only pertains to rows that are not blank. If in plot mode, only pertains to rows `yRow` and `xRow`.

Do updating differently depending on dispMode.

dispMode ads:

```
If float channel, according to alarm flags bit #12,  
    If dispUnits not hex and in averaging mode,  
        fData = fsums/nSum  
    else  
        fData = freading.  
else (if not float)  
    If dispUnits not hex and in averaging mode,  
        Compute average = sums/nSum  
    else  
        If dispUnits = (volts or hex) and channel marked as nonlinear,  
            data = settng  
        else  
            data = readng.
```

Note that the above logic is used to show the raw values for those channels that are used for the nonlinear RF gradients and powers. These channels have no settings, but the settng field is used to hold the original digitized value, so that the readng can be replaced by a linearized value, one that can be linearly scaled to engineering units.

dispMode das:

```
If float channel,  
    fData = fsetting  
else  
    data = settng  
If not zerodata signal, according to conversion type byte bit #2,  
    arrange to use D/A scale factors. Show blank if no setting used.
```

Note that this logic shows the zerodata pedestal for channels that use it.

dispMode noms:

```
If float channel,  
    fData = fnominal  
else  
    data = nomnal  
Show blank if not in alarm scan, or dispUnits = volts and nonlinear.
```

dispMode tols:

```
If float channel,  
    fData = ftolrnce  
else  
    data = tolrnce.  
Show blank if not in alarm scan, or dispUnits = volts and nonlinear.
```

dispMode sets:

```
If float channel,  
    fData = fdASave.  
Show blanks if fData = fsetting or if no setting used.  
  
else  
    data = dASave.  
Show blanks if data = settng or if no setting used.  
Arrange to use D/A scale factors.
```

```

If blanks,
    Prepare to display blocks in number field.
    But if dispMode = sets,
        Show NODE:CHAN instead.
else
    If dispUnits = hex,
        If float channel,
            show float value as 8-digit hex
        else
            show integer value as 4-digit hex.
    else
        If float channel,
            show fData in 10-char field
        else
            If dispUnits = volts,
                fData = data / 3276.8
            else
                fData = data / 32768 * convFact[m]
            If dispMode != tols
                fData += convFact[m+1]
            else
                If data < 0,
                    data = -data.

```

    Show data in 6-char field via cvG.

    If channel in alarm scan and alarm status = BAD,  
    Show number field in inverse video.

Show contents of buffer on screen via TVM.

If associated status bits used for this channel,  
    Show appropriate associated status text.

If vDisplay not current state (dispUnits=volts or hex, or dispMode=sets),

    If integer channel,

        If dispUnits = volts or hex, or dispMode = sets,  
        Show " v. " as 4-char units text

    else

        Show descriptor 4-char units text for this channel.

ShowStat

    Update data request status if changed. If no error status, use blank. Note that only a single character position is allowed for this status indication.

Plot

    Plot dot on semigraphics screen using PBlock struct.

PlotData

    Call Plot to make the dot on screen. Arrange for plotting last point dim.

PlotInit

    Prepare and initialize screen for semigraphics mode.

#### GetTitle

Construct 14-character text field from 18-chars in descriptor. Try to omit first word of text, thinking it might indicate which tank#, say.

#### ParamLinesInit

Initializes each parameter via GetTitle. Call Update for the number field.

#### DoCycle

Perform 15 Hz activities. Collect beam status, derived from Bit 009F in local node. (Zero status indicates beam cycle.)

Process knob, raise/lower activity. First set knobLUpdate = knobLine, clear knobLine. Monitor global variable KNOBDIFF to detect knob adjustment. Assuming KNOBDIFF = 0, check raise/lower lights, and if either is set act as if KNOBDIFF = +-16. This indicates the coarse adjustment given via a rise/lower button, somewhat akin to clicking in the gray area of a modern computer scroll bar. If current line is a settable channel, set knobLine to indicate that a setting is to be performed to that channel. Clear KNOBDIFF.

Monitor changes in mode lites or unit lites, call CopySW to react to such.

If pStage < 0, initializing. Advance pStage. If pStage is now -1, and nameLine > 0, collect response to previously-issued request for name lookup. If valid, update chanId list and call DefineLists. Clear nameLine.

If pStage = 0, initialize all parameter lines. Collect descriptors for all channels as had been requested via DefineLists. Prepare text fields for each line by modifying the descriptor record. Part of this is to convert any lower case characters to upper case, a limitation of the little console display hardware. After all this is done, collect first reply data for both nominals and tolerances and for readings and settings. Then call ParamLinesInit. Call PlotInit if plotMode active. Signal that initializing is done.

Returning to the original check of pStage, if it had been >= 0,

If pStage = 0,

call CopySw, prepare for buidling averaging summations.

If pStage >= 1,

collect readings and settings. If knobLUpdate > 0, call Update for this line, specifying that an individual cycle value should be used in the number field. If plotMode active, call Plot to place the graphic dot accordingly. If knobLine > 0, call SetChanData to deliver setting.

If pStage = 1,

perform special beam-favored cycle average summation logic. This must cover both integer and raw floating point channels.

On the next to last of the 13 cycles over which the average is calculated, perform blinking logic for all channels that are currently in the alarm BAD state and that also are inhibiting beam. On the last cycle, set pStage = 2. Collect nominals and tolerances.

If pStage = 2,

Call update to update the average values. Only skip a channel that matches knobLine or a channel for which the cursor is located in the number field.

In case we got no descriptors (for at least one of the channels), but we have received at least one reply from the slow request (nominals and tolerances), call DefineLitss to start over, hopefully picking up descriptor(s) from a node that had been down.

## PlotKbInt

Initializes to support the plot mode. This is called when a keyboard interrupt ("click") is detected on display line 12, which is used while in that mode to enter plot ranges. Interrupt at the start of that line to return from plot mode to parameter mode. There is nothing to do if there is no valid channel on row 13, which is called yRow. Parse the plot range row to initialize the plotting variables.

## Digital

Handle digital control actions, relevant to what we call "associated digital control." It makes the appropriate setting using listype 22. The details are handled by the underlying system code that supports that listype.

## Numeric

Supports numeric settings. This includes ordinary D/A settings, but it also includes setting nominal or tolerance values. Besides expecting normal decimal values, the ability to enter hex values is also allowed, assuming first that we are in hex units mode. When entering a numeric value, all characters in that number field that are at or to the right of the cursor position are ignored. The idea is that one has just typed the value to be entered, and the cursor is left at the position immediately following the last character of the value. Support for both integer as well as raw floating point channels is included. For the integer case only, special support for ending a specified number of steps to a stepping motor channel is given. Merely suffix the number of steps with an S. Note that the S must be included within the number field.

Call `SetChanData` to execute setting. (Call `SetChanFData` for raw floating point case, in which only fields in the `FDATA` table are set. If hardware is to be touched by such a setting, a local applicaiton must be running to react to any change in raw floating point setting value and pass it to hardware.)

## DoKbInt

If `plotMode` is active, and if the cursor is on the special line 12 and not in column 1, then call `PlotKbInt` to handle it. If the cursor is on line 12 and on column 1, this is the means of establishing plot mode. There are two types allowed, indicated by a `T` (time plot) in column 0, or a `P` (parameter plot).

For a parameter line, which is lines 1-14 for parameter mode, or lines 13-14 for plot mode, act according to the cursor column:

If the cursor column is 0, when there is a valid channel assigned to the line, blank the line. If there is no valid channel on the line, but there is one on the line immediately above, then enter the line with the same node but with that channel plus 1, and advance the cursor position by one line. This facilitates entering a sequence of consecutive channels.

If the cursor column is 1, when there is a valid channel assigned to the line, replace the current channel with the same channel but with the node plus 1. If there is no valid channel on the current line, but there is a valid channel on the line immediately above, enter the same channel with the previous line node plus 1, and advance the cursor position by one line.

If the cursor column is less than 10, scan what was entered, up to the current cursor position, for a separator colon. If one is found, interpret what was entered as the `NODE:CHAN` format. If the `NODE` part is missing, assume the local node#. If the node# part is  $< 0x0100$ , assume the high byte of the node# to be the same as that of the local node#. If the `chan#` part is missing, so that what was entered had the format `NODE:`, then replace all remaining lines starting at the current line with the entered node#, except for those having a different node# from the one that

was on the current line. It is confusing to describe here, but take a simple example of a sequence of lines with channels all from the same node#. Entering `NODE:` on the first of these lines will replace all in the sequence so that the channel numbers match what is already there, but the nodes are set to the node# entered. This is especially useful for a case in which multiple nodes use the same channels for similar signals, such as exists for the low energy Linac RF stations, for example. After doing this, replace the cursor at the start of the line, so it is easy to enter another node# in the same way.

For the same case of the column less than 10, when scanning for a separator colon, if none was found, and the cursor column is  $\leq 6$ , then assume a 6-character channel name was entered. If the name entered was blank, then blank the current line and all remaining lines below it. If the name entered was not blank, make a data request for the node:chan matching the name, and set `pStage = -2`, set `nameLine =` the current line. This should restart things so that the node:chan that comes back from the data request for name translation is used to replace the current line. Note that the translation of name to node:chan is supported by the underlying system software, which for a nonlocal name actually multicasts the data request so that all nodes can see it. Only the node that has that name will respond, and the system will capture that response to provide as reply data to the name lookup request.

If no separator colon was found, and the cursor column was between 8 and 20, then it may have been associated digital control, so call `Digital` to handle it.

If the cursor is  $\geq 23$ , it may be within the number field, so call `Numeric` to handle it. For the integer case, check that the cursor is between 23 and 28; otherwise, it is in the units field, so that it means to update the nominal value to the current reading, called "renominalizing" the channel. For the raw floating point case, only check that the column is  $\geq 23$ , in which case there is no units field; thus renominalizing is not supported for the floating point case.

#### `DoMain`

Main entry to page application. In initialization, allocate and initialize static memory block. Copy list of channels from page memory table (`PAGEM`) entry. Call `copysw` to set up the switch lites. Make data request for beam status, from local Bit `0x9F`. On termination, cancel all data requests. If `x` in page home position, implying that we are returning to the index page, save the state of the current page (channel list, etc), and recall this page. In any case, release static memory.