

FDATA Support

Parameter page addition

Fri, Apr 26, 2002

The basic system code for IRMs and PowerPC nodes supports raw floating point data channels. But this support has yet to be added to `PAGEPARM`, the Parameter page application. This note explores how to add such support.

The `PARM` application uses one-shot requests to obtain analog descriptor information to format a line for each of up to 14 devices on the little console display. It issues requests to obtain 2-byte readings and settings from the `ADATA` table at 15 Hz, and it also requests alarm information to be returned every 2 seconds. This information includes the 2-byte nominal and tolerance words, plus the 2-byte alarm flags and trip count, all from `ADATA`.

Raw floating point data

The raw floating point data is housed in the `FDATA` table, which includes 4-byte floating point reading, setting, nominal, and tolerance values. A bit in the alarms flag word (mask `0x1000`) is set to designate a channel whose data is raw floating point. For such channels, the scale factors in the analog descriptor are not used; instead, the values of such channels only exist as raw floating point. When there are engineering units, such channel's floating point values are understood to be in those units.

Floating point channel data is generated by local applications; there is no standard Data Access Table support that produces such raw floating point readings. In addition, raw floating point setting support must be handled by local applications; there is no specific setting action that results from a raw floating point setting. When such a setting is made, the only direct result is that the setting value is placed into the `FDATA` raw floating point setting field. If other action should result, a local application must undertake to make it happen.

The alarm system includes support for these raw floating point channels. During the alarm scan, when such a channel is encountered, the raw floating point nominal and tolerance values from the `FDATA` table are used instead of the usual 2-byte values in the `ADATA` table.

*Changes to **PAGEPARM***

Returning to the issue at hand, what changes should be made to `PAGEPARM` to support such channels? In addition to requesting 15 Hz readings and settings from the `ADATA` table, we can increase that request so that 15 Hz raw floating point readings and settings are also returned. This means that data will be collected that is not needed, but it makes the implementation easier. Note that the list of up to 14 channels may include an arbitrary mix of 2-byte integer and 4-byte floating point channels. Likewise, the request for 2-second nominal and tolerance values can also be increased to include the raw floating point nominal and tolerance fields from the `FDATA` table. In this way, all necessary data will be available, independent of whether or not the channels are designated floating point. When displaying values in the "number field," the integer channels will have to be scaled using the descriptor scale factors. For raw floating point channels, the values need no scaling.

When performing settings to integer channels, the value entered in the number field has to be reverse-scaled to get the raw 2-byte setting value, whereas for raw floating point channels, the value can be used directly. Of course, the listype specified, as well as the data size, must be chosen correctly. It may be helpful to eliminate the reverse-scaling step for the integer

case, instead using the listype that causes the scaling to be done in the target system. Such changes can simplify the application code, once considered the most complex of all page applications. (Historical note: This assessment was made at a time when for many years only 4 page applications existed: parameter, analog descriptor, binary descriptor, and memory dump.) But in order to detect when a setting reaches an extreme range for 16-bit values, it may still be necessary to perform the reverse scaling. This argument may not apply if the reverse-scaling done in the target station clamps to the extreme value, which it does.

One could consider doing all data collection in floating point, by using the listypes suited to that purpose. But what would be lost is the ability to view the data values in unscaled units, either raw volts or raw hexadecimal. (When the parameter page displays raw hexadecimal value readings, the average values are ignored, so that only sampled values are shown. This ensures that the readings viewed are actual bit values, in case one is looking for “stuck bits.”)

Displaying raw floating point data brings up another problem, in that only 6 characters are made available for the number field on the 32-character display line of the little consoles. How can we show full precision floating point values, which should include at least 6 significant figures? The easiest approach for this is to ignore it and not show the increased precision available. Another approach is to leave out display of the units field for floating point channels. This would free up 4 more characters to make a 10-character floating point field, which is enough for a sign, a decimal point, and 8 digits. If the number is less than 1.0, an initial 0 is traditional. If the value is less than 0.1, then its value starts out with “0.0”. If it were desired to show the floating point value in hexadecimal, 8 characters would be needed just for that.

Is there a need to indicate on the display whether a channel is raw floating point or not? If we allow the hexadecimal option to show 8 hexadecimal digits, the user could push the Hex button to obtain this information. Perhaps this is good enough for the purpose.

Support for the informal plotting (64 X 33 pixels) on the little console hardware displays may be more difficult for such channels. The current support is strongly based upon 16-bit integer. But this plotting capability is not used much, so it may be easiest to skip raw floating point support for it.

What about knob control for floating point channels? This should probably not be supported at all, as it would be difficult to assign suitable scaling between knob clicks and changes in the floating point setting value.

Post-implementation notes

The PAGEPARM program was modified along the lines suggested above. In order to get enough room to display the increased precision available with floating point, the units text is not shown. No special indication is made on the line that identifies raw floating point channels, but one can use the Hex key to determine which channels are of this type. Informal plotting is not supported for such channels. Knob control is not supported, either. While these modifications were made, the support for DZero 16-character names was removed, as the DZero protocol is no longer supported by the underlying system code. The total size of the resulting executable code changed from 10.8K to 11.6K bytes.