ECool KHz Monitoring

KHZM local application

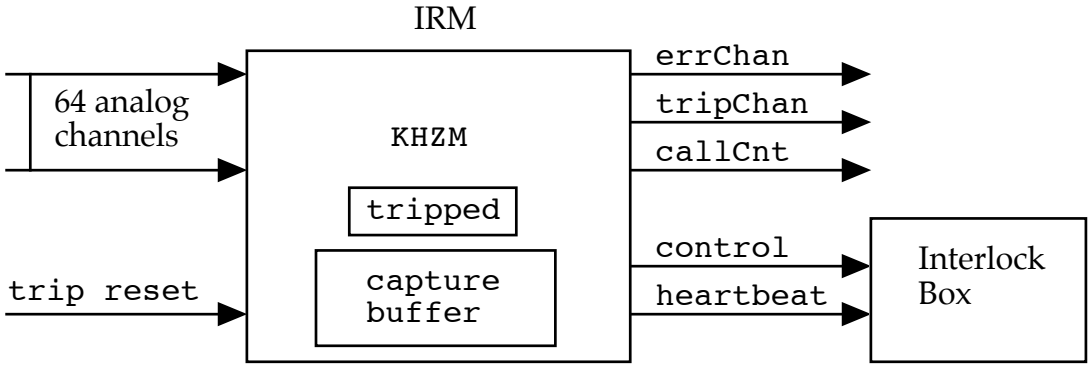
Robert Goodwin

Mon, Jan 24, 2005

Introduction

The ECool project requires monitoring several dozen signals at a rate of about 1 KHz in order to ensure reliable operation while the electron beam is on. An IRM interfaces to analog signals with digitizations of all 64 done by hardware at 1 KHz. The hardware also automatically stores the results into a 64K circular buffer that has room for 512 sets of such data, so that it "wraps" about every half second. A local application called `KHZM` operates at 15 Hz and supplies an interrupt routine that the underlying system code invokes from the digitizer interrupt, which occurs at the end of digitizing the 64 channels over a 800 μs period. Thus, the interrupt code is invoked when the latest set of 1 KHz data just been stored into the buffer. The system passes to the interrupt code an indication of where this latest data set is within the circular buffer. This note describes the functionality of the `KHZM` local application.

The following figure shows some of the components of interest:



The KHz routine supplied by `KHZM` checks up to 64 channels against upper and lower limits every time it is called by the KHz interrupt routine. If all channels scanned are within limits, the output control line is set low, assuming that the internal `tripped` flag is not set. If any channel is outside its specified pair of limits, the `control` line is set high.

Whenever the `control` line transitions from low to high, the `tripped` flag is set, and the offending channel# is latched and stored into the `tripChan` variable. At the same time, the `KHZM` 15 Hz execution is notified that it should copy the current circular buffer into a capture buffer for leisurely external examination, which it does over the following 8 cycles. The result is a set of 64 waveforms of 512 points each, sampled from 400 ms before the trip until 112 ms after the trip, that can be examined to analyze the behavior of all of the scanned signals.

As long as the `tripped` flag is set, the `control` line remains high, and the `errChan` variable always shows the first channel that is out of limits. To return the `control` line low, one must first reset the `tripped` flag, which is done by setting the `trip reset` bit. The `KHZM` 15 Hz execution notices this bit set and clears both the `tripped` flag and the `trip reset` bit itself. Note that this does not pull the `control` line low yet, but the next time the KHz scanning finishes with all channels found to be within limits, the `control` line will be set low. Only the interrupt routine changes the state of the `control` line. The `callCnt` variable is the number of calls made to the KHz routine every 15 Hz cycle; its value should be about 66.

Since continued execution of the interrupt routine is so important, a `heartbeat` output line is provided that is driven by the KHz interrupt routine logic. It is driven high at the start of the interrupt routine execution and driven low at the end, about 100 μs later, if the maximum of

64 channels is being scanned. Its minimum time can be about 6–10 μs if the first channel reading is out of limits. If the interlock box fails to see this `heartbeat` activity, it should assume there is no active interrupt monitoring going on.

Acnet interface

Several Acnet devices besides those of the signals scanned relate to this `KHZM` support. One allows the client application to establish the set of limits for each channel that is scanned. Another allows the client to collect the captured 512-point waveforms of any of the signals being scanned.

Additional devices allow access to the variables `errChan`, `tripChan`, and `callCnt`. The `errChan` variable is zero only when all channels are within limits; otherwise, it indicates the channel# (range `0x0100–0x013F`) of the first channel that is out of limits. The `tripChan` is set to the same value as `errChan` only when a trip occurs and the `control` line is set high. It is cleared when all channels are found within limits and the `control` line is set low. So the `tripChan` variable is changed only on transitions of the `control` line.

There is also a device that allows the client to set the `trip reset` bit that is noticed by the 15 Hz execution of `KHZM`, which clears both the `tripped` flag and the `trip reset` bit itself. It can only be effective in pulling the `control` line low if all signals are subsequently found to be within limits by the next KHz routine scan. If desired, this `trip reset` bit may be associated with either the `errChan` device or the `tripChan` device.

Although the `tripped` flag is an internal variable, its value can be inferred by monitoring the value of the `tripChan` variable. Only if the `tripChan` variable is nonzero is the `tripped` flag set; otherwise it is clear.

Recent changes

When the `KHZM` local application is first initialized, the `control` line is set high, the `tripped` flag is set, and the `tripChan` is set to an invalid nonzero value. The limits are set to impossible values, where the lower limits are higher than the upper limits. This will force the download of all the limits before it is possible to allow normal operation of ECool. When interpreting a `trip reset`, the `tripped` flag will only be cleared if `errChan` is zero.

Some of the analog input channels may be readings of latched conditions from the interlock box. To reset such latched conditions, the `trip reset` bit can be a hardware line that feeds into the interlock box. As soon as it is set, the box will clear its latches, and very soon, the `errChan` should not see these same analog channels in error, so that the reset can succeed.

The KHz interrupt code monitors the normal 15 Hz activity of `KHZM` by requiring the 15 Hz logic to copy the `nCalls` field to another field. If the KHz routine finds that this new field lags too far behind `nCalls`, say by 200 ms, it will not set its heartbeat line high. This change ensures that the 15 Hz activity is timely, even though it is unlikely this would ever occur.

Current status

The logic described here for `KHZM` has been installed in test `node0571`, and scope waveforms of some key signals have been captured by Mike Kucera. In the actual ECool installation, the `KHZM` local application will be installed in `node05E8`.