CAMAC - Support #15854

CAMAC front-ends need to use 720Hz clock delivery support

03/13/2017 04:41 PM - Richard Neswold

Status:	Assigned	**Start date:**	03/13/2017
Priority:	Normal	**Due date:**	
Assignee:	Richard Neswold	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Dennis needs to read an MADC channel on eight different clock events. The complication is that these clock events are separated by 10ms so sometimes the readings are returned out of order. We assume this happens because, by default, MOOC delivers clock events at 60Hz, which is too slow for this application. **Charlie** has added 720Hz clock delivery to MOOC so we'll try to use this feature to see if it fixes the problem. Otherwise a custom driver may have to be written.

History

#1 - 03/13/2017 05:09 PM - Dennis Nicklaus

So here's some typical data from my new muoncollect test program:

Collecting two devices M:OUTTMP and M:TOR109 on events 9a-9e and 96-98. (You can distinguish between the two devices by the values). The "final event" message is triggered by registering for sync:on event 99. It's OK if I get reading 3 of M:outtmp before reading 2 of M:tor109. It is *not* OK to get reading 8 of M:outtmp before reading 7 of M:outtmp (my collection software can easily sort it out, but it means it is not being sampled at the correct time on the camac frontend.)

```
got 2ndphase data {dpm_device_data,1,{1489,441960,87000},28.420800505878177}
got 2ndphase data {dpm_device_data,1,{1489,441960,87000},-0.03827250011265266}
got 2ndphase data {dpm_device_data,2,{1489,441960,97000},28.420800505878177}
got 2ndphase data {dpm_device_data,3,{1489,441960,107000},28.420800505878177}
got 2ndphase data {dpm_device_data,2,{1489,441960,97000},-0.034627500101923836}
got 2ndphase data {dpm_device_data,3,{1489,441960,107000},-0.03705750010907639}
got 2ndphase data {dpm_device_data,4,{1489,441960,117000},28.49056675553483}
got 2ndphase data {dpm_device_data,4,{1489,441960,117000},-0.03705750010907639}
got 2ndphase data {dpm_device_data,5,{1489,441960,127000},28.420800505878177}
got 2ndphase data {dpm_device_data,5,{1489,441960,127000},-0.03827250011265266}
got 2ndphase data {dpm_device_data,6,{1489,441960,137000},28.420800505878177}
got 2ndphase data {dpm_device_data,6,{1489,441960,137000},-0.03827250011265266}
got 2ndphase data {dpm_device_data,8,{1489,441960,157000},28.420800505878177}
got 2ndphase data {dpm_device_data,7,{1489,441960,147000},28.420800505878177}
got 2ndphase data {dpm_device_data,8,{1489,441960,157000},-0.034627500101923836}
got 2ndphase data {dpm_device_data,7,{1489,441960,147000},-0.03827250011265266}
Final event {sync_event,{1489,441961,439525},{clock,153,26337}}
Final event {sync_event,{1489,441961,439525},{clock,153,26337}}
```

#2 - 03/14/2017 09:33 AM - Richard Neswold

Dennis Nicklaus wrote:

> It is *not* OK to get reading 8 of M:outtmp before reading 7 of M:outtmp.
>
> [...]

Both samples, 7 and 8, of M:OUTTMP are exactly the same value, indicating that the two 10ms events got delivered in the same 60 Hz tick.

How does the trigger library deliver clock events? Doesn't it use bit masks to keep track of which events fired (throwing away the timestamp)? If that's the case, then the events should be deliver in ascending order. However, multiple processes taking the MOOC semaphore will block and, if they're at the same priority level, get unblocked nondeterministically (because the mutex is priority order, not FIFO.)

The solution still seems we need to turn on the 720Hz delivery.