

Timer Tracking

Local application
Robert Goodwin
Tue, Oct 19, 2010

There is a need in HINS and HTS for timer tracking, allowing a user, when setting one timer, to have another timer track that setting. This note presents a new LA called `TTRK` that is designed to provide such support.

The parameter layout is as follows:

<i>Param</i>		<i>Size</i>	<i>Meaning</i>
ENABLE	B	2	Usual LA enable Bit#
(SPARE)		2	(n.u.)
RFG DLY	C	2	Timer delay, width Chan#s set by user
RFG WID	C	2	
THRESH	C	2	Threshold to distinguish large timer width, Chan#
HI DLY	C	2	Tracking timer additional delay for width \geq THRESH, Chan#
LO DLY	C	2	Tracking timer additional delay for width $<$ THRESH, Chan#
END PRE	C	2	Amount to cut short tracking timer width, Chan#
TRK DLY	C	2	Tracking timer delay, width Chan#s
TRK WID	C	2	

The timer that the user sets, both its delay and width, are specified as RFG channel #s. To be sure that the tracking logic takes place seamlessly, so that no untracked cycles occur, these channels are dummy channels; that is, they are software channels, for which a setting does not target the actual hardware timers. When the LA notices a change in either of these software channels, it makes the appropriate change to the target hardware channels. If more hardware timer channels must track, they are specified by subsequent instances of this same LA. All adjustments are then made at once, with no chance of any setting being interrupted. As a result, all tracking timers are modified as appropriate to track the software channels set by the user.

As per the original request for this feature, the tracking timer is to be delayed by one of two values, depending on whether the width of the RFG timer is long or short. The `THRESH` parameter specifies what width is considered long. The `HI DLY` parameter reading is the delay to be used for long pulses; the `LO DLY` reading is used for short pulses. The `END PRE` reading specifies the time that the tracking timer width should end before the user-set timer ends.

Here is the specific requirement originally requested:

THRESH	100
HI DLY	10
LO DLY	1
END PRE	1

Consider two examples. If the RF gate timer is set for 1000 μ s delay and 300 μ s width, then the tracking timer will be set for 1010 μ s delay and 289 μ s width. If the RF gate timer is set for 1000 μ s delay and 50 μ s width, then the tracking timer will be set for 1001 μ s delay and 48 μ s width.

In order to make the changes appear seamless, it is necessary that the user change software channels, allowing the LA to set the hardware channels with the proper specified relationship all at once. If two timers must track the software channels, then two instances of `TTRK` are used. The first instance should specify zeros for the 4 parameters above, which will make the first timer track

without modifications. (It will not use the reading of channel #0.) The second timer can specify the necessary modifications; as a separate instance, it can specify its own parameters.

To let all instances act at once, the *setting* field is monitored by this LA, not the *reading* field. This is because the setting field is updated immediately when a setting is made, whereas the reading field is only updated at the start of the next cycle after the setting is made. This scheme will allow any number of timers to track all at once; thus, the hardware sees only a clean switch.

Because the front end is synchronized with the accelerator hardware operation, no hardware adjustments are made during timer activity. The "Micro-P Start" timing trigger is used to signal to the front end to perform its normal cycle activity, including running all active local applications. If the timers drive an RF pulse, for example, Micro-P Start is set to follow the end of the RF pulse.