

NAME

ni660x – National Instruments 6601/6602 PCI counter/timer

DESCRIPTION

The National Instruments models 6601 and 6602 PCI counter/timers provide four and eight 32-bit counter channels respectively. The 6601 specifies a maximum unprescaled count rate of 20 MHz while the 6602 specifies 80 MHz. However, all inputs may be prescaled by a factor of eight, extending the usable maximum count rates to 60 MHz for the 6601 and 125 MHz for the 6602. (The prescaling also gives three more bits of effective counter size.)

When counting to time, for shorter count times, `spec` uses the 20 MHz built-in clock. If prescale is on for the time channel (which it is by default), the effective clock rate is 2.5 MHz. With a 32-bit counter, that allows for a maximum count time of just under 28.6 minutes (1717.9 seconds). If prescale is off, the maximum count time is 214.7 seconds (3.6 minutes). For longer count times, `spec` uses the 100 KHz clock with prescaling off, allowing for a maximum count time of nearly 12 hours (as of `spec` release 5.06.05-3).

`spec` will keep track of overflows to the channel counting time (when not counting to a time preset), but does not currently correct for counter overflows in the other channels.

`spec` also provides commands for selecting the hardware's digital debouncing filter for external inputs and for accessing the eight digital I/O pins available on these cards. See the *FUNCTIONS* section below for details.

SOFTWARE CONFIGURATION

Use `spec`'s configuration editor (*edconf*, normally invoked by the `config` macro) to select the timer and to configure the scaler channels. On the **Devices** screen, you need only enter the the number of counters to be used on the board. The base address of the PCI cards is determined automatically.

```
SCALERS  DEV  ADDR  <>MODE  NUM                                <>TYPE
        YES                                4  NI PCI 6601/6602 as Counter/Timer
        YES                                8      NI PCI 6601/6602 as Counters
```

On the **Scaler** screen, choose `NI660x` as the device type. The unit numbers are assigned consecutively to the NI PCI counter/timers according to the order they are listed on the **Devices** screen, starting with unit zero.

```
NUMBER    NAME    MNEMONIC  <>DEVICE  UNIT  CHAN  <>USE AS  SCALE
    0  Seconds    sec      NI660x    0    0    timebase  1
    1  Monitor    mon      NI660x    0    1    monitor   1
    2  Detector    det      NI660x    0    2    counter   1
    3  Counter 2    c2       NI660x    0    3    counter   1
```

3 Counter 3	c3	NI660x	1	0	counter	1
3 Counter 4	c4	NI660x	1	1	counter	1

Any channel can be used as the time base. Also, any channel can be used as the monitor-preset channel. In addition, the monitor channel assignment can be changed while running with `counter_par()` (see the *FUNCTIONS* section below).

The scale factor is ignored for the time-base channel, since the time base is set and accounted for internally in the C code.

THEORY OF OPERATION

One of the key requirements for counter/timers in `spec` is the capability to have a master channel that will gate the slave counter channels by way of a hardware signal. A second requirement is that at least two channels (one for time and one for monitor) can be used as the master channel. Even better is hardware that allows any channel to be used as the master channel. The NI 6601 and 6602 meet these requirements.

The 660x card architecture allows any channel to be configured to use any other channel's associated gate input pin as its own gate. `spec` requires each possible master counter's output connected to its associated gate via a jumper. That counter's output will then serve as gate for all the counters. See the *Hardware Configuration* section below.

Normally all outputs are disabled. When a particular counter is the master, `spec` configures just that channel to have its output activated during counting to serve as gate for all channels. When the master counter counts down from its preset and reaches zero, the last count changes the state of the output, which disables the slave counters. This achieves a reliable hardware gating.

If the card is set up as only counters, `spec` configures the card so that all channels will gate off the gate pin associated with the lowest numbered channel included in the `config` file. The external gate from the master timer device can be connected to that pin.

If the 660x is to be used as master to gate other cards, one must connect the various designated output pins of the possible master counter channels together and use that signal. `spec` will only activate one output pin at a time. All the other pins will be in the default power-on state of high-impedance inputs, so there is no danger of overloading the outputs as long as `spec` is used to control the card.

By default, `spec` programs each channel for an active-high gate. That is, during counting the output signal from the master channel will be nominally at +5V. The `counter_par()` command can be used to configure the gate

signal as active low (see below), if that makes gating other devices more convenient. Note, the gate mode is reset to the default after reading the hardware *config* file. If using active-low gates, add the appropriate command for setting active-low gates to a `config_mac` macro, which will be called automatically during hardware configuration.

Although hardware gating is preferred for counters, `spec` does in any case send appropriate commands to arm and disarm the counters in the absence of hardware gating. The arm command occurs before the master timer is started. The disarm command occurs after the master timer has stopped. If no hardware gate is used with NI PCI cards configured as just counters, the active-low gating mode needs to be set, otherwise the counters will stay inhibited.

HARDWARE CONFIGURATION

For each counter channel that will be used as a master timing channel (either with the internal time base or with an external monitor-count preset), you need to connect the associated output pin to the associated gate pin. It doesn't hurt to make all the connections, even if you don't anticipate the channel being used for a monitor preset channel. All models can have the following connections:

- Out(0) pin 5 to Gate(0) pin 3
- Out(1) pin 9 to Gate(1) pin 8
- Out(2) pin 32 to Gate(2) pin 67
- Out(3) pin 29 to Gate(3) pin 64

For the eight-channel 6602, these additional connections can be made:

- Out(4) pin 26 to Gate(4) pin 61
- Out(5) pin 23 to Gate(5) pin 58
- Out(6) pin 53 to Gate(6) pin 21
- Out(7) pin 16 to Gate(7) pin 51

A channel will not work properly until its corresponding output-to-gate jumper is installed. That includes the time-base channel.

Input signals connect to the following pins. Note, one channel will probably be used for the time base, as selected in the *config* file. There's no need to connect an input signal to that channel, since it will be programmed to use the internal clock.

- Source(0) pin 2
- Source(1) pin 7
- Source(2) pin 34
- Source(3) pin 31

For the 6602 model, the additional inputs are:

Source(4) pin 28

Source(5) pin 25

Source(6) pin 22

Source(7) pin 52

Choose any of the many ground pins as ground.

FUNCTIONS

The following `counter_par()` options are available. The first three commands are associated with particular counter channels:

`counter_par(mne, "monitor")` – Sets the channel assigned to mnemonic *mne* to be the monitor preset channel. The monitor channel gets reset when the *config* file is read on start up or on reconfig.

`counter_par(mne, "prescale" [, how])` – Without the optional argument, returns nonzero if the channel for *mne* has been configured with the divide-by-eight prescale. If called with *how* nonzero, turns on the divide-by-eight prescale. Rereading the *config* file resets the mode to the default, which is prescale on for the time-base channel, and prescale off for the rest.

`counter_par(mne, "filter" [, val])` – Without the optional argument, returns zero, 1e-6, 5e-6, 1e-7 or 5e-7 indicating which digital debouncing filter is associated with the input signal for the corresponding counter channel. The units are seconds. Otherwise, sets the filter according to *val*. A parameter of zero disables the filter. The code also accepts 1, 5, 100 or 500 as valid arguments. According to the NI documentation, input pulses with widths equal to or greater than the value indicated are guaranteed to be passed, and pulses with widths of half the value or less are guaranteed to be blocked. This parameter can also be set in the *config* file as optional scaler parameter "misc_par_1". (Type *s* from the main Scaler screen to access the optional parameters.) (Filter support added in spec release 5.07.03-1.)

The following options are associated with the operation of the card, not with a particular channel. The *mne* argument is necessary for the syntax of the command and should be the mnemonic or counter number of any of the configured channels associated with the card.

`counter_par(mne, "active_low" [, how])` – Without the optional argument, returns nonzero if the counting card associated with channel *mne* has been configured for active-low gates. If called with *how* nonzero, turns on the active-low mode for gating. Otherwise, sets the

default active-high gate mode. Rereading the *config* file resets the mode to active high. The mode only needs to be set for one channel to affect all channels on the card.

`counter_par(mne, "io_config" [, val])` – Bits set in *val* will be configured as outputs on the digital I/O lines. On power up, all lines are configured as inputs. If the optional argument is missing, *spec* will return the current value as set by the user. (The value can't be read from the card itself.)

`counter_par(mne, "set_bit", val)` – Set output lines corresponding to bit number *val* to high, where bits and lines are numbered from 0 to 7.

`counter_par(mne, "clr_bit", val)` – Set output line corresponding to bit number *val* to low, where bits and lines are numbered from 0 to 7.

`counter_par(mne, "put_byte", val)` – Sets output lines corresponding to the bits set in *val* to high and bits not set in *val* to low.

`counter_par(mne, "get_bit", val)` – Returns 0 or 1 based on whether the input line corresponding to bit number *val* is high or low, where lines and bits are numbered from 0 to 7.

`counter_par(mne, "get_byte")` – Returns an 8-bit value that reflects the setting of the input lines.