

**NAME**

am9513 – PC board and USB counter/timers based on the AMD Am9513 chip

**DESCRIPTION**

spec supports counter/timer PC cards and USB devices based on the Advanced Micro Devices Am9513 chip. Although production of the original AMD chip ended in 1995 (after twenty years), compatible chips from other manufactures are still being made. Counter/timer models supported by spec include: Measurement Computing (formerly ComputerBoards) model CIO-CTR05/10/20 ISA cards, PCI-CTR05/10/20HD PCI cards, and the USB-4300 series; Keithley-Metrabyte Model CTM-05/10 ISA cards, Scientific Solutions Labmaster series ISA and PCI cards, Kontron's PCIDCC5/20-P PCI card.

The Am9513 chip contains five 16-bit counters that can be programmed in a wide range of configurations. Although boards may contain more than one of the chips, the first chip on the timer/counter board is used as the master timer. spec's programming uses two of the counters of the master timer for a 32-bit detector counter, two for a 32-bit monitor counter and one for a 16-bit elapsed time counter. On boards with two or four chips, the additional chips are each programmed for two more 32-bit detector counters. spec supports the Am9513 chip as both a time preset counter using the `tcount()` function or a monitor-count preset using `mcount()`.

Note, spec release 5.06.03 includes several updates to the Am9513 support. With the update, more than one Am9513-type card can be configured at a time, where the additional cards must be configured as counters only. Also, the code was changed to improve the timer gating, although an additional jumper is required to make the change effective. Previously, the timer was started in one instruction, followed by another instruction that released the gate signal that inhibited the counters. In the new code, the timer will also be gated by the gate signal, but only if the additional jumper from *output 1 to gate 1* is installed. (See the **HARDWARE CONFIGURATION** section below). spec will test for the presence of the additional jumper. If it is missing, spec will program the timer for no gate, as before, and print a message suggesting the jumper be added.

Note also, spec release 5.09.02 includes several significant updates to Am9513 support. The Am9513-compatible counter/timer chips used by Measurement Computing in PCI cards and USB devices behave slightly differently than the original chips. This difference prevented the chips working in counting-to-monitor mode using spec's original code. This release includes a work around in the spec support, and counting-to-monitor mode now works.

In addition, spec release 5.09.02 includes `counter_par()` commands to access the digital I/O ports on all versions of the device. See the **FUNCTIONS** section below.

**SOFTWARE CONFIGURATION**

Use spec's configuration editor (*edconf*, normally invoked by the `config` macro) to select the timer and to configure the scaler channels. For ISA cards, enter the base address on the *Device* screen. For all cards, enter the number of counters to be used on the board:

SCALERS	DEV	ADDR	<>MODE	NUM	<>TYPE
YES		0x340		3	Am9513 Counter/Timer (ISA)
YES				3	Am9513 Counter/Timer (PCI)
YES		0		3	Am9513 Counter/Timer (USB)
YES		0x340		2	Am9513 Counter-Only (ISA)
YES				2	Am9513 Counter-Only (PCI)
YES		0		2	Am9513 Counter-Only (USB)

Note that for the Labmaster board, the base address of the counter chip is eight plus the base address of the board itself. For PCI cards, the address is determined automatically. The address field can be used to enter the serial number of USB devices if more than one is to be configured. If the address is zero, the USB devices will be assigned in the order found.

For each additional chip on a board, add 2 to the value in the NUM field.

If more than one PCI Am9513 board is installed, the unit numbers are assigned to the boards, starting with unit 0, in the same order as the PCI bus/slot/function IDs are assigned by the computer. (The *Linux* command `/sbin/lspci` lists the PCI identifiers.)

As of spec release 5.06.03, the command

```
counter_par(mne, "device_id")
```

where *mne* is the mnemonic of a counter associated with a PCI Am9513 card, will return a string containing the I/O base address of the PCI counter/timer cards, which may be useful in writing macros that access other features of the card, such as digital I/O, A/D converters, etc. (With spec release 5.09.02, the digital I/O is accessible via `counter_par()`. See below.)

On the *Scaler* screen, the channel assignments must have the timebase as channel 0 and the monitor as channel 1, as shown below.

NUMBER	NAME	MNEMONIC	<>DEVICE	UNIT	CHAN	<>USE AS	SCALE
0	Seconds	sec	AM9513	0	0	timebase	1
1	Monitor	mon	AM9513	0	1	monitor	1
2	Detector	det	AM9513	0	2	counter	1
3	Counter 2	cnt2	AM9513	0	3	counter	1
4	Counter 3	cnt3	AM9513	0	4	counter	1
5	Counter 4	cnt4	AM9513	1	0	counter	1
6	Counter 5	cnt5	AM9513	1	1	counter	1

The timebase scale factor is ignored. Counters on unit 0, channels 3 and 4 would only be available if there are at least two Am9513 chips on the card. Counters on unit 1 correspond to a second Am9513 card.

#### HARDWARE CONFIGURATION

Connect the detector to the input connector pin labeled *source 3*. Counts received from the monitor go to the pin labeled *source 5*. In addition, wire the connector pin labeled *output 1* to the pins *gate 1*, *gate 2*, *gate 4* and *gate 5*. (In the more recent Keithley-Metrabyte CTM-05A manual, the *source* pins are now labeled *ACLKIN*, the *output* pins are now labeled *ATIMER-OUT*, and the *gate* pins are now labeled *AGATE*.)

For a two- or four-chip board, or for additional counter-only boards, additional detectors can be connected to the *source 3* and *source 5* pins of the chips. In addition, the *output 1* from the first chip of the timer/counter board must be connected to *gate 2* and *gate 4* of the additional chips.

The counter boards are accessed from user level and are polled to determine when count intervals have elapsed. Thus, interrupts should be disabled on the boards.

When counting to time, the resolution of the clock depends on the length of the count interval. The maximum count time is 71.5 minutes. The time base resolution (in seconds) is set according to the following table:

0.00001	for $t < 0.6$ sec
0.0001	for $t < 6$ sec
0.001	for $t < 60$ sec
0.01	for $t < 655.35$ sec (10.9 min)
0.0655	for $t < 71.5$ min

When counting to monitor counts, the 0.01 second time base is used, and the value returned for the time channel will be corrected to account for the rollovers that occur every 655.36 seconds.

## FUNCTIONS

The following `counter_par()` options are available. Note, some models have 8 bits each of input and output and some models have 16 bits each. Also, some models allow the output bit settings to be read, while some models do not. For models that do not allow reading of the output registers, `spec` keeps track of the values that have been written, so that it is possible to set and clear individual bits without disturbing settings on other bits. Note, though, that after reinitializing the hardware during startup or `reconfig`, the previous state of read-only output registers will be lost.

The digital I/O is associated with a counter card or device, not with a counter channel. However, the `spec counter_par()` functions requires a counter-channel argument. In the calls below, *mne* is any counter channel associated with the particular card or device having the digital I/O ports being addressed.

`counter_par(mne, "set_bit", val)` – Set output line corresponding to bit number *val* to high, where bits and lines are numbered from 0 to 7 or 0 to 15.

`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 or 0 to 15.

`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, "put_word", val)` – On devices with 16 bits of output, 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 or 0 to 15.

`counter_par(mne, "get_byte")` – Returns an 8-bit value that reflects the setting of the input lines 0 to 7.

`counter_par(mne, "get_word")` – On devices with 16 bits of input, returns a 16-bit value that reflects the setting of the input lines 0 to 15.