1 TIMING GATE GENERATOR (V124S)

1.1 HARDWARE

The V124S card provides eight independently configurable timing channels which can be triggered from:

- An event on the SNS event link,
- An external TTL signal,
- A software trigger from the IOC.

Channels may also be combined to provide extra long delays or widths. The board can also interrupt the IOC on error conditions, channel termination conditions, and timestamp conditions.

For specific details about the V124S hardware, register map, and programming, see reference [3][*].

1.1.1 Board Configuration Properties

Each V124S card has several "Board Configuration" properties – properties apply to the card in general rather than individual timing channels. These properties include the card-wide delay, the revolution counter re-synch event, the timestamp re-synch event, the event link error counters, and the interrupt controls.

Figure 6 in section 1.2 shows the diagnostic EDM screen that displays the card-specific properties of the V124S module.

1.1.1.1 Board Command Register

The card as a whole can be enabled or disabled. When disabled, no gates are generated. For testing purposes, the card can also be configured to use an internal clock instead of the event link clock.

1.1.1.2 Interrupt Sources

The card can be configured to interrupt on any of the following sources:

- Event Link carrier error (loss of carrier signal)
- Event Link frame error
- Event Link parity error
- Timestamp re-synch event
- Individual channel termination
- Individual channel timestamp.

The SNS V124S driver software normally enables only the three event link error interrupts. To prevent a bad event link from flooding the system with interrupts, the interrupt rate is throttled down to 4 Hz maximum.

Timestamp re-synch and individual channel timestamp interrupts are not currently implemented. The timing master sequencer task uses the individual channel termination interrupt. The timing master IOC defines an "End-of-Cycle" gate (not an event) that occurs 100

[*] The documentation in [3] may still contain some early design information that may be either superceded or not fully implemented. Where information in [3] conflicts with information contained in this document, this document will take precedence (except in those instances where I made a mistake).

turns after the Extract event. The timing master sequencer task waits on the termination interrupt of this gate to set up the RTDL and the variable rep-rate gates for the next cycle.

1.1.1.3 Error Counters

Error counters are maintained for each of the three event link error conditions. These counters are useful for determining whether the event link has had problems in the past or is currently having problems. Since the error interrupt is throttled to 4 Hz maximum, these counters will only increment at a maximum rate of 4 Hz. The error counters are reset by an IOC reboot and by a "Clear Errors" call to the V124S driver.

1.1.1.4 Revolution Frequency Re-Synch Event

As mentioned in section **Error! Reference source not found.**, the event link clock is derived from the ring RF signal multiplied by 32. The timing system hardware needs to know which of the 32 × Frev clock pulses corresponds to the start of a ring revolution. This is the purpose of the "Revolution Frequency Re-Synch Event" register. When the V124S detects the event number contained in this register, it resets its revolution counter and declares the corresponding clock cycle (and every 32nd clock cycle afterward) to be the start of a revolution.

For all SNS applications, the value in this register should always be "1" (the Cycle-Start event).

1.1.1.5 Timestamp Reset Event

The value in this register should also always be "1" (the Cycle-Start event). This allows the V124S hardware to reset the timestamp counters at the start of each machine cycle. For more on time-stamping, see section 1.1.2.9 below.

1.1.1.6 Clock Delay

This register allows you to tweak the timing of the entire card by up to 127.5 nanoseconds, in increments of one half nanosecond (500 picoseconds).

1.1.2 Channel Configuration Properties

Each individual timing gate has its own associated delay, width, trigger, and timestamp. The delay contains coarse, medium, and fine controls. “Coarse control” is the “Revolution Delay Count”, which aligns the gate with the start of a revolution and increments in units of “Turns” (about 945 nanoseconds at 1 GeV). “Medium control” is the “Sub-Revolution Delay Count”, which increments in units of “sub-revolutions” (about 29.5 nanoseconds at 1 GeV). “Fine control” is in absolute units of 500 picoseconds.

The gate delay and width countdowns are written to buffer registers. The actual counters are loaded from these buffer registers on the gate’s “Halt Trigger”. The halt trigger occurs when the last output pulse from a triggered gate completes. The counters are also loaded immediately after the buffer registers are written so that the new values will be there the next time the gate is triggered. As a result of this, if the buffer registers are written while one of the counters is running, the counter register will be reloaded and the actual delay or width for that pulse will be unpredictable. If this is a problem, you should ensure that the delay and/or width buffer registers are not written while the gate is active.

Figure 7 in section 1.2 shows the diagnostic EDM screen from which a gate’s individual properties can be displayed and adjusted.

1.1.2.1 Revolution Delay Count

This value specifies the number of revolutions (or “Turns”) to delay between the time the gate is triggered and the first output pulse is produced. Recall that one revolution is 32 sub-revolutions. This does not mean, however, that a revolution delay count of 1 is always 945 nanoseconds. The counter is decremented on every “Revolution” clock pulse – which is determined by the Revolution Frequency Re-Synch event as described above in section 1.1.1.4. A non-zero revolution delay count will therefore cause the output gate to be synchronized with the start of a ring revolution – regardless of when the gate was triggered.

The revolution delay count register may have any value between 0 and 65535. When running with a revolution delay count of 0, however, care must be taken to ensure that the “Delay Control Register” specifies “Manual” for the revolution delay and either “Manual” or “Event” for the sub-revolution delay. Normally, the V124S driver software will set these values correctly for you (see section 1.1.2.8 below for a discussion of the delay control register).

1.1.2.2 Sub-Revolution Delay Count

This value specifies the number of sub-revolutions to delay after the revolution delay count expires. The sub-revolution delay must be between 1 and 63 (just under two revolutions). The V124S will not support a sub-revolution delay of zero. If you need to start a gate just at the start of a revolution, you will need to adjust the revolution delay counter to be one less than the delay you want and then set the sub-revolution delay counter to 32.

1.1.2.3 Fine Delay Count

This register allows you to do fine adjustments on the gate delay. The fine delay is not tied to the ring RF. The adjustment is in fixed time units of 500 picoseconds. The fine delay register can be set to any value between 0 and 127.5 nanoseconds.

1.1.2.4 Pulse Width

The pulse width is given in sub-revolutions. The value may be anywhere from 0 to 65535 sub-revolutions for a maximum gate width of about 1.9 milliseconds at 1 GeV. Note that a gate width of 0 will not produce an output pulse.

1.1.2.5 Trigger Count

This register determines how many pulses will be generated each time the gate is triggered. This is a 32-bit register, so the value can be anywhere from 1 to 4,294,967,295. The V124S driver does not allow a trigger count of 0 (which is really 4,294,967,296).

If the trigger count is greater than one, the revolution, sub-revolution, and fine delays are re-applied and another pulse is output. This continues until the trigger count is exhausted. Note that multiple pulses will only be generated if the revolution delay counter (section 1.1.2.1) is greater than zero.

It is important to remember that the revolution delay counter decrements on every "Revolution Clock" pulse – which is determined by the Revolution Frequency Re-Synch event (see section 1.1.1.4). Consequently, a "delay" of one revolution will not always translate into a delay of 945 nanoseconds. For example, Figure 1 below shows a trigger count of two with a revolution delay of 1 and a sub-revolution delay of 1. The second pulse occurs exactly 945 nanoseconds after the start of the first pulse, and one sub-revolution after the start of the next turn.

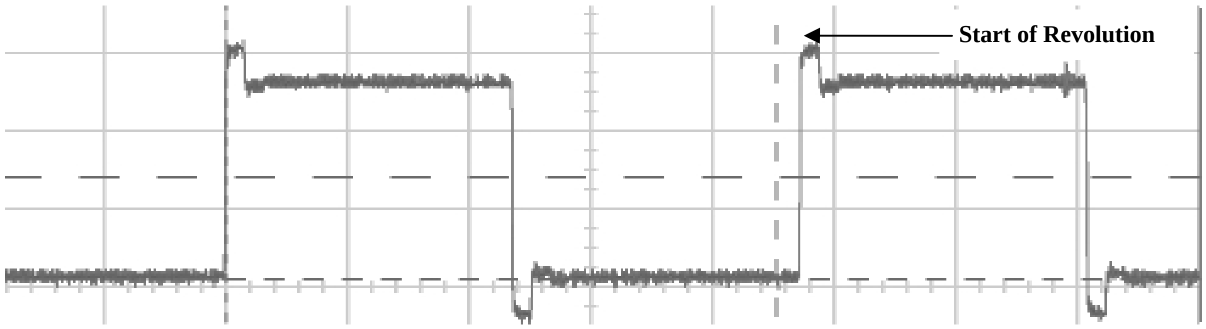


Figure 1 Multiple Triggers
Delay = 1 Revolution + 1 Sub-Revolution, Width = 16 Sub-Revolutions

In Figure 2, we see the same setup with a width of 24 sub-revolutions. Note that even though the width of the pulse has increased, the starting point of the second pulse stays the same.

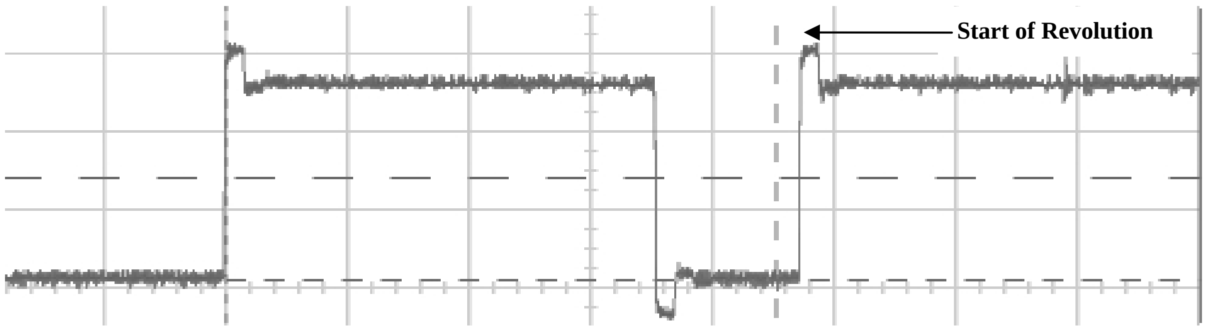


Figure 2
Width Increased to 24 Sub-Revolutions

There is a discontinuity when the width is 32. Instead of waiting until the start of the next revolution after the end of the first pulse (i.e. every other pulse), the second pulse begins after a delay of only 1 sub-revolution after the end of the first pulse.

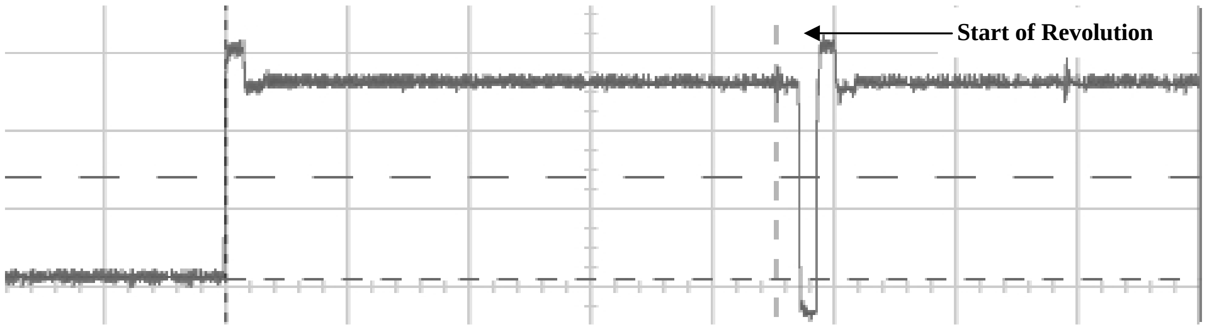


Figure 3
Width Increased to 32 Sub-Revolutions

If the trigger count is greater than 2, you will see an oscillating pattern of short and long delays between the pulses.

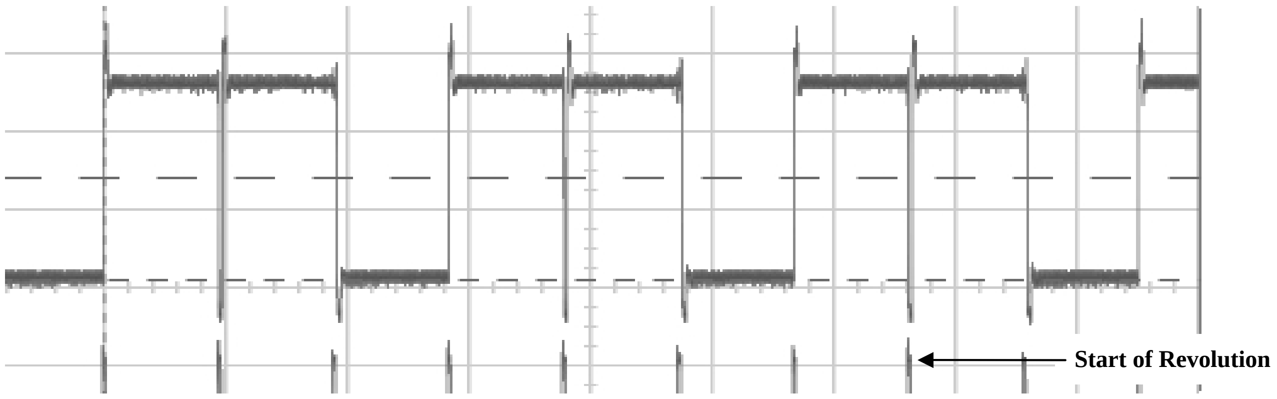


Figure 4
Multiple Trigger Pattern Anomaly When Width is 32 Sub-Revolutions

One final point to make about the trigger count is that because of the way revolutions are counted, each of the pulses will occur at the same point relative to the start of the turn (i.e. start plus one sub-revolution, start plus two sub-revolutions, etc.) This implies that if the revolution delay is one, and the sub-revolution delay and width are less than 32, there will always be exactly 32 sub-revolutions between the start of each pulse (excepting the "Width=32" anomaly). If the width or sub-revolution delay is between 33 and 63, there will always be 64 sub-revolutions (two turns) between each pulse.

1.1.2.6 Trigger Event

This register contains the number of the event that will trigger this gate. A value of zero means that the gate is not triggered by an event. When a non-zero number is written to this register, the V124S driver software automatically changes the values in the "Delay Control Register" (see section 1.1.2.8 below) to reflect the fact that the gate is triggered by an event.

It should be noted here that the V124S hardware is capable of triggering a gate from several different events at once. The SNS V124S driver software does not support this option.

1.1.2.7 Channel Control Register

This register allows you to set the output polarity of the pulse, stop the pulse from counting, reset the gate to a default state, trigger the pulse manually, and set it up for “Single-Shot”. The “Trigger” button on the diagnostic screen shown in Figure 7 (section 1.2) will cause the gate to fire immediately. “Single-Shot” mode is achieved by setting the “Reload Counters” bit to “Manual”. A single-shot is triggered by writing the “Trigger Count” register.

1.1.2.8 Delay Control Register

The “Delay Control Register” determines when the revolution and sub-revolution delays get applied, and when the gate “Halts” (stop’s producing output pulses). The register has three sections:

Options for the revolution delay count are:

Manual:	Wait for a software trigger to start the revolution delay counter. This is the option used for variable rep-rate and single-shot gates. This option is also used when the revolution delay count is zero.
Event:	Wait for the specified event to start the revolution delay counter.
External:	Wait for an external TTL signal to start the revolution delay counter.
Previous:	Wait until the previous channel’s revolution delay has completed before starting this channel’s revolution delay counter. This feature can be used to gang two channels together to produce very long delays.

Options for the sub-revolution delay count are:

Manual:	Wait for a software trigger to start the sub-revolution delay counter. Do not wait for the revolution counter to finish. This option is normally only used when the revolution delay counter is 0.
Event:	Wait for the specified event to start the sub-revolution delay counter. Do not wait for the revolution counter to finish. This option is normally only used when the revolution delay counter is 0.
Rev Done:	Wait for the revolution counter to finish before starting. This is the option that should be used when there is a non-zero revolution delay.
Rev Start:	Pretty much the same thing as “Rev Done”. The V124S driver software uses “Rev Done” by default.

Options for the halt delay selection are:

No Halt:	Channel never “Halts”. Trigger count is ignored. Triggers are repeated *ad infinitum.*
Fine Delay:	Gate “Halts” after the fine delay of the last pulse.
Last Pulse:	Used to gang gates together for long delays

1.1.2.9 Time-Stamping

A V124S timing channel can be time-stamped either by the event link clock, or by specific events. The “Timestamp Configuration Control Register” controls how channel time-stamping is performed. This register has two sections. The first section controls when the timestamp is applied and the second section controls the clock used to measure the timestamp.

Four options are possible for when the timestamp is applied:

None:	Time-stamping is not performed for this channel.
Event:	Timestamp is applied when a specified event is detected. The number of the event that will trigger the timestamp is given in the “Timestamp Trigger Event” register.
First Pulse:	Timestamp is applied at the start of the gate’s first output pulse. This is the default value for channel time-stamping.
Last Pulse:	Timestamp is applied at the start of the gate’s last output pulse. If the trigger count is set to 1 (i.e. there only is one output pulse per trigger), the applied timestamp will be the same as in the “First Pulse” case.

There are two options for the timestamp clock:

Carrier:	The 32 × Frev clock from the event link. When this option is selected, the timestamp value will be the number of subrevolutions from the last Cycle-Start event until the timestamp trigger.
Event:	When this option is selected, the event number specified in the “Timestamp Clock Event” register provides the timestamp clock. The timestamp value will be the number of times the specified event occurred after the last Cycle-Start event. The timing master sequencer program uses this feature on the “End-of-Cycle” gate. By setting the timestamp clock event to “3” (MPS Auto Reset Event), the timing master can count the number of “Fast Protect” trips occurred during the last cycle and set the appropriate bit in the “Last Cycle Veto” RTDL frame.

When the event link clock is used to timestamp a channel, the V124S hardware will always return a value that is slightly larger than the actual timestamp. The exact amount is determined by whether you trigger on the first or last output pulse.

- If you trigger on the first pulse, the timestamp value will be two subrevolutions too large.
- If you trigger on the last pulse, the timestamp value will be three subrevolutions too large.

The V124S driver software compensates for these offsets, so that the timestamp value will always appear correct.

1.1.3 Jumper Settings

The VME A16 base address is set via five jumpers located in the upper middle of the card (see Figure 5 below). The jumpers set base address lines A11 - A15 (2K boundary). With the board standing upright (as shown below), the high-order bit is at the top and the low order bit is toward the bottom. If a jumper is present, it represents a logical "0". If the jumper is removed, it represents a logical "1".

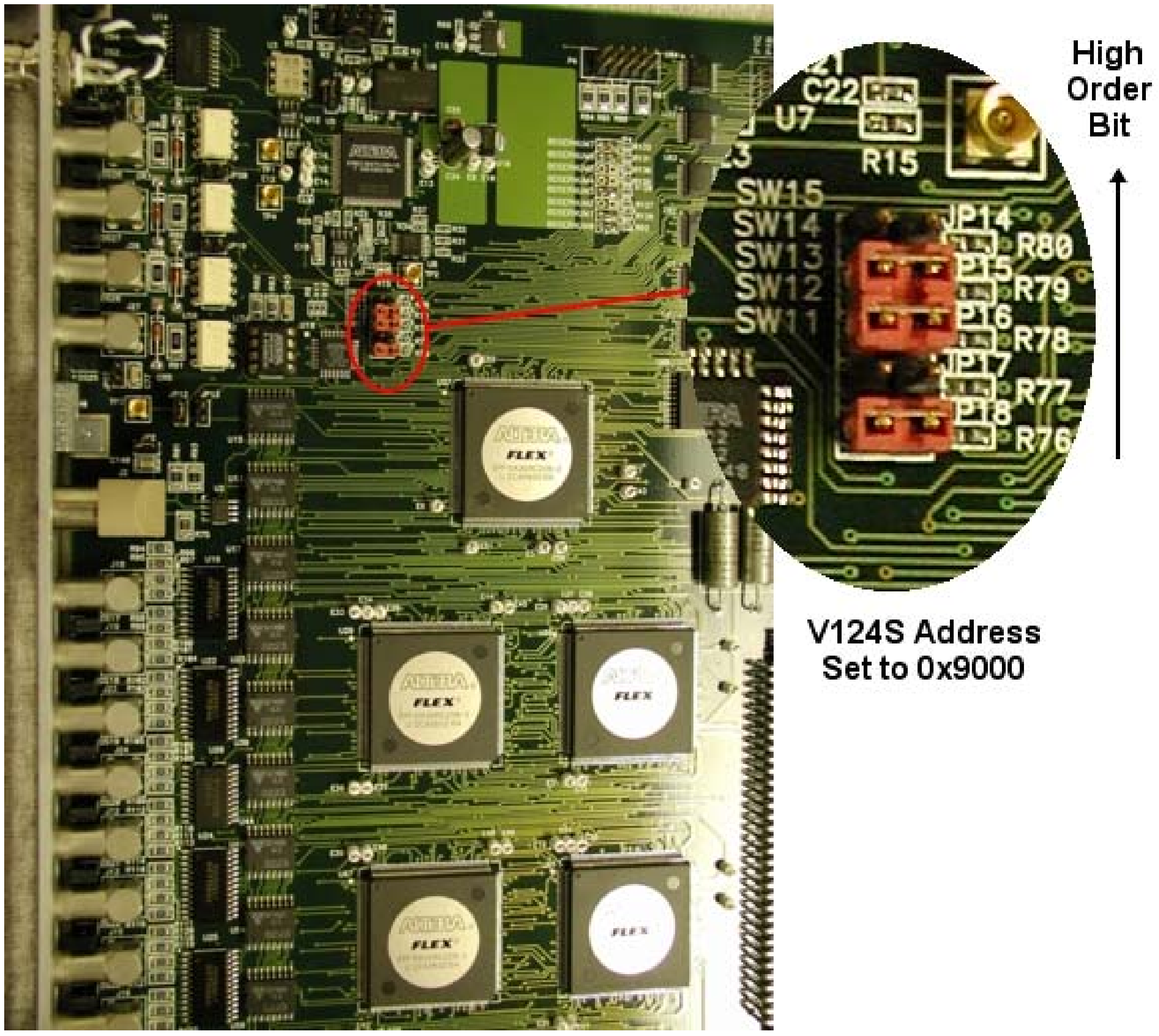
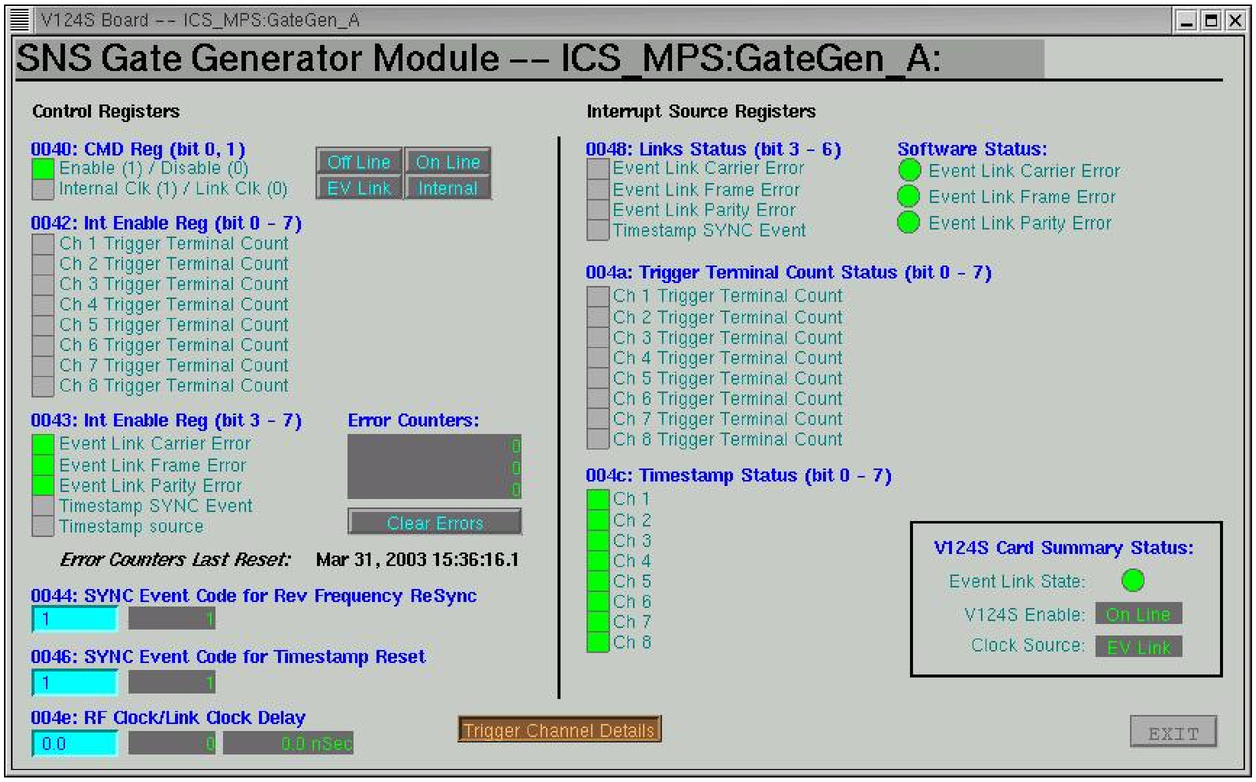


Figure 5
V124S Address Selection

1.2 DIAGNOSTIC SCREENS

The top level V124S diagnostic screens can be obtained by starting at the Timing Master EDM screen and selecting the “Timing System Diagnostics” related display button. From the Timing System Diagnostic screen, select the “Event Trigger (V124S) Screens” related display button. This will bring up a menu from which you can select the top-level screen for the V124S card you wish to observe:



V124S Board -- ICS_MPS:GateGen_A

SNS Gate Generator Module -- ICS_MPS:GateGen_A:

Control Registers

0040: CMD Reg (bit 0, 1)
- 🟩 Enable (1) / Disable (0)
- ▢ Internal Clk (1) / Link Clk (0)

Off Line | On Line | EV Link | Internal

0042: Int Enable Reg (bit 0 - 7)
- ▢ Ch 1 Trigger Terminal Count
- ▢ Ch 2 Trigger Terminal Count
- ▢ Ch 3 Trigger Terminal Count
- ▢ Ch 4 Trigger Terminal Count
- ▢ Ch 5 Trigger Terminal Count
- ▢ Ch 6 Trigger Terminal Count
- ▢ Ch 7 Trigger Terminal Count
- ▢ Ch 8 Trigger Terminal Count

0043: Int Enable Reg (bit 3 - 7)
- 🟩 Event Link Carrier Error
- 🟩 Event Link Frame Error
- 🟩 Event Link Parity Error
- ▢ Timestamp SYNC Event
- ▢ Timestamp source

Error Counters: 0 0 0

Clear Errors

Error Counters Last Reset: Mar 31, 2003 15:36:16.1

0044: SYNC Event Code for Rev Frequency ReSync: 1 1

0046: SYNC Event Code for Timestamp Reset: 1 1

004e: RF Clock/Link Clock Delay: 0.0 0 0.0 nSec

Trigger Channel Details

Interrupt Source Registers

0048: Links Status (bit 3 - 6)
- ▢ Event Link Carrier Error
- ▢ Event Link Frame Error
- ▢ Event Link Parity Error
- ▢ Timestamp SYNC Event

Software Status:
- ● Event Link Carrier Error
- ● Event Link Frame Error
- ● Event Link Parity Error

004a: Trigger Terminal Count Status (bit 0 - 7)
- ▢ Ch 1 Trigger Terminal Count
- ▢ Ch 2 Trigger Terminal Count
- ▢ Ch 3 Trigger Terminal Count
- ▢ Ch 4 Trigger Terminal Count
- ▢ Ch 5 Trigger Terminal Count
- ▢ Ch 6 Trigger Terminal Count
- ▢ Ch 7 Trigger Terminal Count
- ▢ Ch 8 Trigger Terminal Count

004c: Timestamp Status (bit 0 - 7)
- 🟩 Ch 1
- 🟩 Ch 2
- 🟩 Ch 3
- 🟩 Ch 4
- 🟩 Ch 5
- 🟩 Ch 6
- 🟩 Ch 7
- 🟩 Ch 8

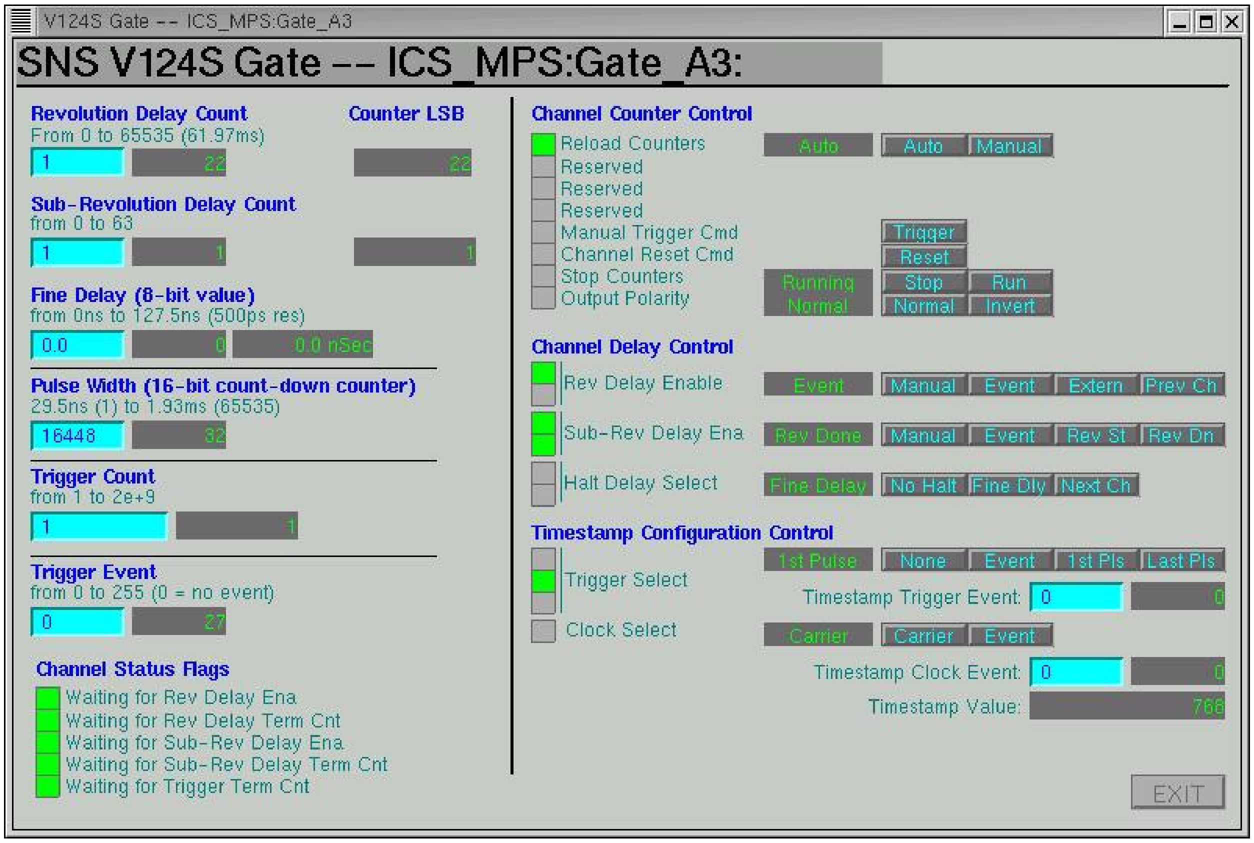
V124S Card Summary Status:
- Event Link State: ●
- V124S Enable: On Line
- Clock Source: EV Link

EXIT

Figure 6
Top Level V124S Diagnostic Screen

In a normally operating V124S card, the CMD register (upper left corner) should indicate “Enabled” (bit on) and “Link Clk” (bit off). The “Trigger Terminal Count” interrupts are normally not enabled. The “Event Link” error interrupts should be enabled and the related error counters should not be incrementing. The “SYNC Event Codes” for both the “Rev Frequency ReSync” and the “Timestamp Reset” should be 1.

To see the details about an individual V124S gate, select the “Trigger Channel Details” menu button on the bottom of the screen and chose the desired channel card you wish to view. This will bring up the following screen:



V124S Gate -- ICS_MPS:Gate_A3

SNS V124S Gate -- ICS_MPS:Gate_A3:

Revolution Delay Count — From 0 to 65535 (61.97ms): 1 / 22 — **Counter LSB**: 22

Sub-Revolution Delay Count — from 0 to 63: 1 / 1 — 1

Fine Delay (8-bit value) — from 0ns to 127.5ns (500ps res): 0.0 / 0 / 0.0 nSec

Pulse Width (16-bit count-down counter) — 29.5ns (1) to 1.93ms (65535): 16448 / 32

Trigger Count — from 1 to 2e+9: 1 / 1

Trigger Event — from 0 to 255 (0 = no event): 0 / 27

Channel Status Flags

- Waiting for Rev Delay Ena
- Waiting for Rev Delay Term Cnt
- Waiting for Sub-Rev Delay Ena
- Waiting for Sub-Rev Delay Term Cnt
- Waiting for Trigger Term Cnt

Channel Counter Control

- Reload Counters: Auto — Auto | Manual
- Reserved
- Reserved
- Reserved
- Manual Trigger Cmd — Trigger
- Channel Reset Cmd — Reset
- Stop Counters: Running — Stop | Run
- Output Polarity: Normal — Normal | Invert

Channel Delay Control

- Rev Delay Enable: Event — Manual | Event | Extern | Prev Ch
- Sub-Rev Delay Ena: Rev Done — Manual | Event | Rev St | Rev Dn
- Halt Delay Select: Fine Delay — No Halt | Fine Dly | Next Ch

Timestamp Configuration Control

- Trigger Select: 1st Pulse — None | Event | 1st Pls | Last Pls
- Timestamp Trigger Event: 0 / 0
- Clock Select: Carrier — Carrier | Event
- Timestamp Clock Event: 0 / 0
- Timestamp Value: 768

EXIT

Figure 7
V124S Individual Gate Diagnostic Screen

The values in the light blue boxes are setpoints. The values in the dark gray boxes are readbacks. As can be seen above, the setpoint values may not necessarily reflect the readback values – particularly if the IOC has been through a “power-on” reboot.

1.3 EPICS SOFTWARE

The EPICS software for the V124S module resides in the directory:

```
$SHARE/timing/production/
```

The “bin”, “db”, “dbd”, “include”, and “opi” directories all contain useful items. The “bin” directory contains architecture-specific sub-directories, each of which contain the file:

```
snsTimingLib
```

This file contains the EPICS driver and device support for the V124S module. It also contains the subroutines for implementing variable rep-rates and for translating between timing system units (turns, sub-revolutions, fine-delay) into clock time (microseconds).

The “dbd” directory contains the following .dbd files:

- **timingGateInclude.dbd:** This is an unexpanded dbd include file that references all the record types and all the driver and device definitions needed by the database templates in the “db” directory. This file is suitable for inclusion in an application’s “*xxx*AppInclude.dbd” file.
- **timingGate.dbd:** This file only includes the V124S driver and device definitions. This file is suitable for loading at boot time, via a “dbLoadDatabase” call, if your IOC does not build a single application-wide .dbd file.
- **timing.dbd:** This file is a fully expanded version of “timingGateInclude.dbd”. This file is primarily used by vdct as the reference .dbd file when constructing or editing the template files.

The “db” directory contains useful template files, including the diagnostic templates for individually inspecting the specifics of each gate. The diagnostic templates (and their macros) are:

v124s.template

Diagnostic database for reading and controlling all the "card-wide” registers of the V124S module. These include clock selection, card-wide delay, error counters, etc. This template should be instantiated once for each V124S card in the system.

Macros:

S	=	System Name
SS	=	Sub-System Name
N	=	System Instance
DI	=	Device Instance -- typically the board "letter" (A, B, C, ...)
CD	=	Card number (0, 1, 2, ...)

The above macros are used to construct the PV names, which are of the form:

$(S)_$(SS):GateGen$(N)_$(DI):<signal>

A typical PV name using this format might be:

ICS_Tim:GateGen_A:Enable

(note that in this case the system instance, $(N), is empty.)

timingGateDiag.template –

Diagnostic database for reading and controlling all the channel-specific registers of the V124S module. This template should be instantiated once for each of the eight channels on a V124S card

Macros:

S	=	System Name
SS	=	Sub-System Name
N	=	System Instance
DI	=	Device Instance -- typically the board "letter" followed by the gate “number” (e.g. A1, A2, A3, …)
CD	=	Card number (0, 1, 2, ...)
GATE	=	Gate number (1, 2, ... 8)

The above macros are used to construct the PV names, which are of the form:

```
$(S)_$(SS):Gate$(N)_$(DI):<signal>
```

A typical PV name using this format might be:

```
ICS_Tim:Gate_A1:FineDly
```

(note that in this case the system instance, $(N), is empty.)

Other useful template files are:

staticGate.template –

Database template for gates that are fixed in time and do not have operator-adjustable parameters (hence the term “static”). The gate's parameters are all specified as macros to the template. The generated records all have "PINI=YES", so that the gate parameters are written once at boot time.

Macros Defining the PV Names:

S	=	System Name
SS	=	Sub-System Name
N	=	System Instance
DI	=	Device Instance -- typically the gate's "name"

Macros Defining the Gate's VME Address

CD	=	Card number (0, 1, 2, ...)
GATE	=	Gate number (1, 2, ... 8)

Macros Defining the Gate's Parameters:

RevDly	=	Gate's revolution delay
SubRevDly	=	Gate's sub-revolution delay (0 - 63)
FineDly	=	Gate's fine delay (in nanoseconds)
Width	=	Gate width (in Sub-Revolutions)
Event	=	Event number to trigger the gate from. 0 = no event.

TrigCnt	=	Number of gates to generate per trigger (usually 1)
Polarity	=	0 means normal polarity (high true)
		1 means reverse polarity (low true)
AutoReload	=	1 means gate is automatically triggered
		0 means the gate is manually triggered by VME command or by "single-shot"

dataTrigger.template –

Database template for data acquisition trigger gates. These gates have operator-adjustable delays (turns, sub-revolutions, fine delay) which are referenced relative to the start of the beam gate. The operator may also select whether to trigger the gate on the "Fast Trigger" (6 Hz), the "Slow Trigger" (1 Hz), the "Snap Shot Trigger" (triggered manually), or on no trigger at all. The gate width, trigger count, polarity, and AutoReload flag are all configured by macros in the template.

Macros Defining the PV Names:

S	=	System Name
SS	=	Sub-System Name
N	=	System Instance
DI	=	Device Instance -- typically the name of the triggered device

Macros Defining the Gate's VME Address

CD	=	Card number (0, 1, 2, ...)
GATE	=	Gate number (1, 2, ... 8)

Macros Defining the Gate's Parameters:

Width	=	Gate width (in Sub-Revolutions)
TrigCnt	=	Number of gates to generate per trigger (usually 1)
Polarity	=	0 means normal polarity (high true)
		1 means reverse polarity (low true)
AutoReload	=	1 means gate is automatically triggered
		0 means the gate is manually triggered by VME command or by "single-shot"

variableRepRate.template –

Database template to set up a gate with a variable rep-rate. This template does not affect any of the gate's parameters other than rep-rate. The rest of the gate parameters will need to be set up via another method (such as the staticGate.template).

Macros Defining the PV Names:

S	=	System Name
SS	=	Sub-System Name
N	=	System Instance
DI	=	Device Instance -- typically the name of the triggered device

Macros Defining the Gate's VME Address

CD	=	Card number (0, 1, 2, ...)
GATE	=	Gate number (1, 2, ... 8)

Macros Defining the Gate's Parameters:

OFFSET	=	The offset (in cycles) from the constraint pattern. This is useful for creating "precursor" gates. (OFFSET = -1)
BASE	=	The constraining rep-rate pattern. This will typically point to the VALB field of a rep-rate gensub record (see section **Error! Reference source not found.**). Note that the macro should specify the both the record and field names. For example: "ICS_Tim:Gate_SourceOn:RRSub.VALB" If the BASE macro does not specify a valid PV name, then the rep-rate pattern will be unconstrained. Typically, an unconstrained rep-rate is specified by setting BASE = 60 (although any number would do).
MAXREP	=	Maximum rep-rate value.
SOFT	=	If 1 (TRUE), the gate should not be triggered on the cycles it is scheduled for. It should only be marked as triggered. If 0 (FALSE), trigger the gate on the cycles it is scheduled for (this feature is used by the timing master sequencer for beam-related gates that the sequencer wants closer control of).

1.3.1 Setting Up a Timing Client

The V124S software support depends on the gensub record and the SNS Utility Module support libraries. Client IOCs that use the V124S card must therefore link with and load the gensub record support, the utility module software, and the V124S software. The following areas in the client IOC application may need modification:

1.3.1.1 config Directory RELEASE File

The first step in adding a V124S card to your IOC is to go to the top level “config” directory, and edit the RELEASE file. Make sure the following symbols (or their equivalent) are defined:

```
GENSUB  = $(SHARE_RELEASE)/Gensub
UTILITY = $(SHARE_RELEASE)/utility/production
TIMING  = $(SHARE_RELEASE)/timing/production
```

This will make sure that the other application Makefiles have access to all the gensub, utility module, and V124S definitions and code they need.

1.3.1.2 src Directory Makefile.Vx File

The SNS standard is for an application to copy all the system and utility software it needs into its local “bin” directory and load it from the local directory at boot time. If your IOC implements this method, you will need to edit the “Makefile.Vx” file in your application’s “src” directory and add the following lines:

```
BIN_INSTALLS += $(GENSUB_BIN)/genSubRecord.o
BIN_INSTALLS += $(UTILITY_BIN)/snsUtilLib
BIN_INSTALLS += $(TIMING_BIN)/snsTimingLib
```

These lines will cause a “make” in the “src” directory to copy the latest versions of the gensub, utility module, and timing client libraries into your local “bin” directory.

1.3.1.3 src Directory *xxx*AppInclude.dbd File

A good practice is to create an application-wide ".dbd" file. The gensub, utility, and timing directories all contain "*xxx*Include.dbd" files that contain all the record, device, and driver definitions you need for each library. Any duplicate definitions will get sorted out when the *xxx*AppInclude.dbd file is expanded into the *xxx*App.dbd file during the make process (note that duplicate .dbd definitions are not sorted out when several .dbd files are loaded at boot time – this is why the *xxx*AppInclude.dbd method is preferred).

To make sure all the relevant timing, utility and gensub definitions are included in your application's .dbd file, insert the following lines into your *xxx*AppInclude.dbd file:

```
include "snsUtilInclude.dbd"
include "timingGateInclude.dbd"
```

1.3.1.4 src Directory base.dbd and baseLIBOBJS Files

If you are using the *xxx*AppInclude.dbd method to build an application-wide .dbd file, then the only thing you need to worry about including in the "base.dbd" file are the base record, device, and driver support routines that your own application uses.

The baseLIBOJBS file, on the other hand should be edited to include all record types used by the timing and utility module databases. These record types are:

```
aiRecord        aoRecord        biRecord        boRecord
calcRecord      calcoutRecord   eventRecord
longinRecord
longoutRecord   mbbiRecord      mbboRecord      selRecord
subRecord       stringinRecord
```

1.3.1.5 Db Directory *xxx*App.substitutions File

Each IOC with a V124S card should include the basic V124S diagnostic databases. This will allow for easier troubleshooting of timing system problems. The V124S diagnostic database consists of two parts, a card-specific part, and a channel-specific part. The card-specific part contains overall status information about the V124S card. It is represented by the template file, "v124s.template". The channel-specific part contains information about the individual timing channels. It is represented by the template file "timingGateDiag.template".

If you use a substitutions file to create your databases, you can define the V124S diagnostic database here with lines of the form:

```
#---------------------
# Diagnostic Records for the V124S card
#
file v124s.template {
    {S=<<system>> SS=<<sub-system>>, N=<<system-instance>>, DI=A, CD=0}
}

#---------------------
# Diagnostic Records for Each V124S Timing Gate
#
file timingGateDiag.template {
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A1, CD=0,
GATE=1}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A2, CD=0,
GATE=2}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A3, CD=0,
GATE=3}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A4, CD=0,
GATE=4}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A5, CD=0,
GATE=5}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A6, CD=0,
GATE=6}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A7, CD=0,
GATE=7}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A8, CD=0,
GATE=8}
}
```

1.3.1.6 iocBoot Directory st.cmd File

The IOC startup command file will require several modifications. First, the gensub record, utility module, and timing support libraries must be loaded. If you load all your software from your local "bin" directory, then the following lines should be added to your st.cmd file:

```
cd appbin
   :
<< Other System Libraries >>
   :
ld < gensubRecord.o
ld < snsUtilLib
ld < snsTimingLib
```

Note that it is important to load "snsTimingLib" after loading "gensubRecord.o" and "snsUtilLib".

Next you need to initialize the utility module and the V124S hardware. Before you call "iocInit", you should have the following lines:

```
#----------------------------------------------------
# Initialize the Utility Module
#----------------------------------------------------

snsUtilInit 0, 0x4000
snsUtilConfigEvent 0, 0x40, 5

#-----------------------------------------------------
# Initialize the V124S Timing Gate Generator
#-----------------------------------------------------

v124sConfig 0, 0x9000, 0x3A, 3, 0.0
```

These lines initialize the SNS utility module and V124S cards with their default addresses, interrupt vectors, and interrupt request priorities.

If your IOC does fragmented dbd loads, the file "$SHARE/timing/production/dbd/timingGate.dbd" contains just the V124S driver and device definitions. The template files in "$SHARE/timing /production/db" can also be loaded at boot time via "dbLoadRecords".

1.4 DRIVER SOFTWARE

The EPICS driver and device support software for the V124S card reside in the files:

```
$SHARE/timing/production/timingApp/src/drvV124S.c
$SHARE/timing/production/timingApp/src/devV124S.c
```

These files are built into the "snsTimingLib" library which needs to be loaded by any IOC that uses a V124S card. The header file "v124s.h" (in the same directory) contains the interface definitions and is installed in the directory:

```
$SHARE/timing/production/include
```

The routines below are implemented in the drvV124S.c module.

status = v124sConfig (card, base, vector, level, offset)

This routine is called from the vxWorks startup file to configure the V124S card's VME address, interrupt vector, and base time offset.

Arguments:

card	=	Card number (as specified in the EPICS record INP/OUT link fields).
base	=	Base address in A16 space.
vector	=	Interrupt vector.
level	=	Interrupt request level.
offset	=	(double) Time offset adjustment for this card (in nanoseconds).

Returns:

OK	=	V124S card was successfully configured.
ERROR	=	V124S card configuration had errors.

status = v124sInit ();

This routine is normally called only from the EPICS iocInit routine. It completes the V124S card initialization by connecting the interrupt vectors, and enabling card interrupts.

Returns:

OK	=	V124S card was successfully initialized.
ERROR	=	V124S card initialization had errors.

v124sReport (level)

This routine is called from the EPICS dbior routine to report the location and status of each V124S card configured for the system. It may also be called from the vxWorks shell.

Arguments:

level	=	Level of detail to report. This parameter is currently ignored.

V124S_HANDLE = v124sCardHandle (card)

Returns a driver handle for the specified V124S card.
Returns NULL if the specified card was not configured (via the v124sConfig function).

Arguments:

card	=	Card number to return the handle for.

status = v124sGet (handle, channel, tag, &value)

This is the generic driver routine for returning scalar information from the V124S driver and hardware.

Arguments:

handle	=	Driver handle to the V124S card we want to read from.
channel	=	Channel number (1 - 8) we want to read from. Channel 0 is used to return card-specific (as opposed to channel-specific) data.
tag	=	Function code describing what we want to read (tag values are defined in v124s.h).
value	=	Address of an unsigned longword to receive the data.

Returns:

OK	=	The read succeeded.
ERROR	=	The read failed.

status = v124sGetBytes (handle, channel, tag, size, &buffer)

This is the generic driver routine for returning byte array information from the V124S driver and hardware.

Arguments:

handle	=	Driver handle to the V124S card we want to read from.
channel	=	Channel number (1 - 8) we want to read from. Channel 0 is used to return card-specific (as opposed to channel-specific) data.
tag	=	Function code describing what we want to read (tag values are defined in v124s.h).
size	=	Size of return buffer (in bytes)

buffer	=	Address of buffer to receive the data.

Returns:

OK	=	The read succeeded.
ERROR	=	The read failed.

status = v124sPut (handle, channel, tag, value)

This is the generic driver routine for writing scalar information to the V124S driver and hardware.

Arguments:

handle	=	Driver handle to the V124S card we want to write to.
channel	=	Channel number (1 - 8) we want to write to. Channel 0 is used to write card-specific (as opposed to channel-specific) data.
tag	=	Function code describing what we want to write (tag values are defined in v124s.h).
value	=	Unsigned longword containing the data to write.

Returns:

OK	=	The write succeeded.
ERROR	=	The write failed.

status = v124sWaitTrigger (handle, channel)

Wait for a "Halt Trigger" interrupt from the requested channel.

Arguments:

Handle	=	Driver handle for the V124S card.
channel	=	Channel to wait on.