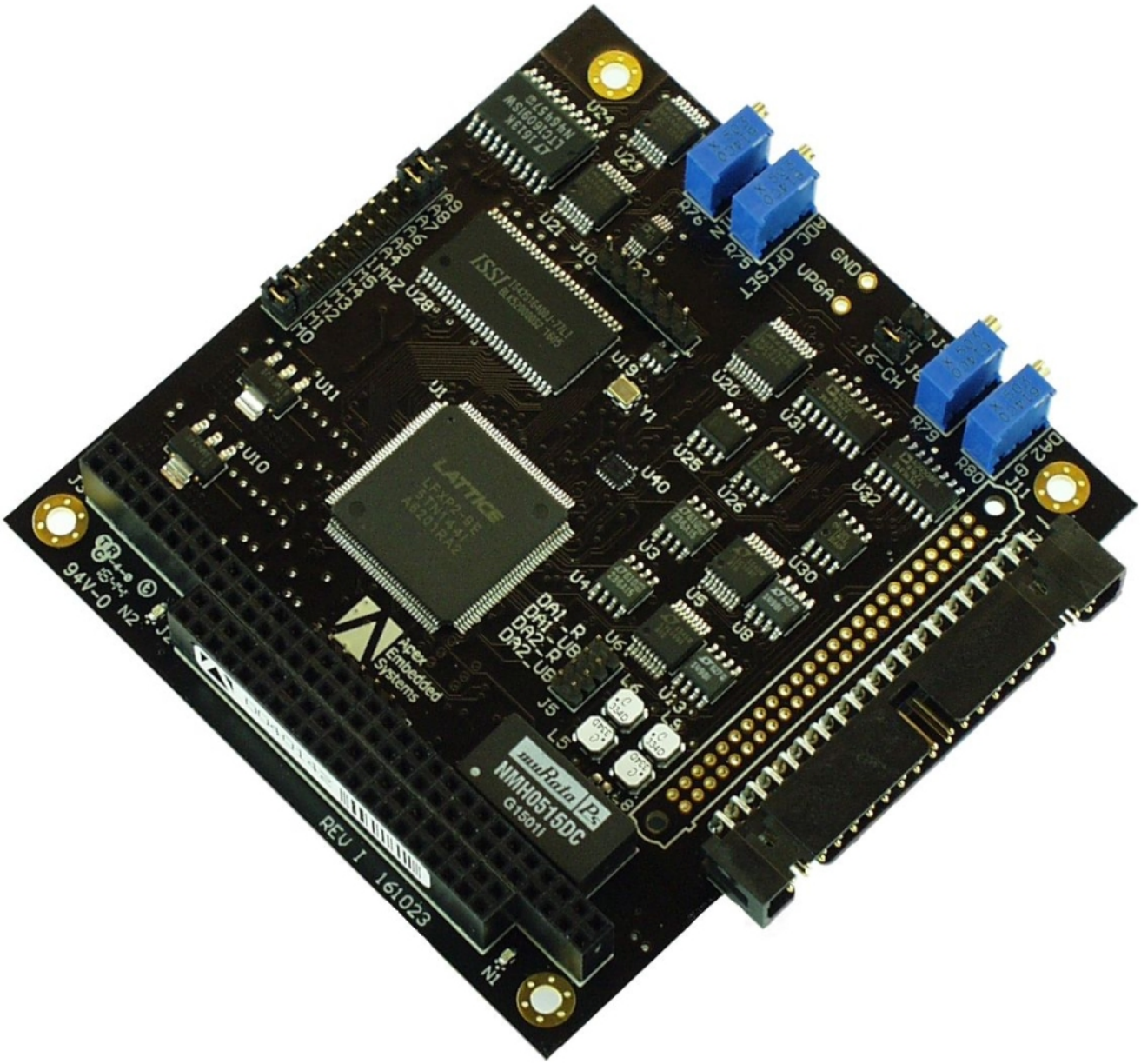
STX 104 Reference Manual



Apex Embedded Systems

STX104(-ND) Reference Manual

Analog I/O with 16-Bit Resolution

Apex Embedded Systems LLC

116 Owen Road

Monona, WI 53716

Phone 608.256.0767

www.apexembeddedsystems.com

customer.service@apexembeddedsystems.com

Table of Contents

STX104(-ND) Reference Manual

1 Welcome

Dear Valued Customer:

Thank you for your interest in our products and services.

Apex Embedded Systems "Continuous improvement" policy utilizes customer feedback to improve existing products and create new product offerings based on needs of our customers.

Continued Success,

Apex Embedded Systems

2 Legal Notice

Apex Embedded Systems' sole obligation for products that prove to be defective within 1 year from date of purchase will be for replacement or refund. Apex Embedded Systems gives no warranty, either expressed or implied, and specifically disclaims all other warranties, including warranties for merchantability and fitness. In no event shall Apex Embedded Systems' liability exceed the buyer's purchase price, nor shall Apex Embedded Systems be liable for any indirect or consequential damages.

This warranty does not apply to products which have been subject to misuse (including static discharge), neglect, accident or modification, or which have been soldered or altered during assembly and are not capable of being tested.

DO NOT USE PRODUCTS SOLD BY APEX EMBEDDED SYSTEMS AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS!

Products sold by Apex Embedded Systems are not authorized for use as critical components in life support devices or systems. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

3 Benefits and Features

3.1 Executive Summary

What is the STX104? The STX104 is a 16-channel 16-bit analog input and 2-channel 16-bit analog output card. The STX104 incorporates a large one mega-sample FIFO.

Description

What are the Benefits of using the STX104? The STX104 has the following benefits:

• One mega-sample FIFO data storage provides continuous data streaming from ADC to CPU with reduction in interrupt overhead and relaxation of interrupt latencies.

• 200,000 Samples per second aggregate for analog inputs

• Analog input 16-Sample moving average filters for data noise reduction

• Compatible with DAS16jr/16 and DAS1602 for using existing 3rd party drivers

• 16-bit data read (ADC data) operations double effective PC/104 bus bandwidth

• 16-bit data write (DAC data) operations reduce software overhead

• Use REP INSW (286 or higher CPU) to read-in blocks of ADC data from FIFO further increasing bandwidth and reducing complexity.

• Burst mode with only one interrupt generated per complete scan, thus reducing interrupt overhead and increasing effective throughput.

• One interrupt per 512-samples is possible in FIFO mode

• On-board LED to indicate that the STX104 is being addressed. By observing the LED you can quickly determine system activity.

• Polarized locking I/O connector. This eliminates board failures due to incorrect connector orientation.

• Extremely low DC drift

• External trigger input (DIN0) is deglitched preventing unwanted ADC triggers. Minimum valid pulse width required is 200nS.

• Industrial temperature range from -40^{O}C to +85^{O}C

• No tantalum capacitors or electrolytic capacitors used in the design

• Single +5V supply operation

New Features as of February 14, 2008 (Revision 080407H)

- Supports both 8- and 16-bit PC/104 data busses

- Supports 10-bit or 16-bit I/O addressing via jumper using jumper M4. DAS1602 register set located at base_address + 0x400 are also now relocated in an indexed register spaced within the 10-bit address space.
- Improved I/O wait states as well as wait state reduction supporting higher speed CPUs along with improved PC/104 bus throughput.
- New triggering subsystem (see page 97)
  - Trigger Start. 14 selectable events.
  - Trigger Stop. 15 selectable events.
  - Trigger Sync. Trigger Sync. 12 selectable events.
  - Synchronization via external signal sources (i.e. 60Hz)
  - Three possible triggering sequences.
  - Trigger delay timer.
- Interrupts (see page 90)
  - More IRQ lines supported including IRQ9, IRQ10, IRQ11, IRQ12, IRQ14 and IRQ15.
  - Two additional interrupt sources. Each has 13 selectable events.
  - Interrupt source status available at one location for faster interrupt service routines.
  - Interrupt threshold counter (see page 92) for multiple events per interrupt. 13 selectable events.
  - Interrupts can be synchronized to trigger start event.
  - Number of analog input data blocks per FIFO interrupt is now adjustable (two methods possible).
- Digital Outputs (see page 94)
  - Polarity control.
- Digital Inputs (see page 96)
  - Polarity control.
  - Long (200 nSec) or short (100 nSec) deglitch filter.
- Analog Inputs
  - Sampling sources (see page 100). 11 selectable source (all legacy functions have been preserved).
  - 32-bit intra-sample burst timer (see page 105) with resolution to 25 nanoseconds for improved timing between samples in ADC-burst mode.
  - Non-synchronization/synchronization with trigger start (see page 100). In other words, sample timing can remain at fixed intervals regardless of triggering start event or be synchronized to the triggering start event, respectively.
  - 32-bit frame timer (see page 104) with resolution 25 nanoseconds. ADC-Burst sample sequences or ADC-sampling can be controlled by this 32-bit timer.
  - 32-bit burst (intra-sample) timer (see page 105) with resolution to 25 nanoseconds. This timer is used to adjust the timing between ADC-samples during an ADC-Burst operation (i.e. one or more channels collected at a time).
  - 32-bit maximum frame counter (see page 108). This counter can be used to count the number of ADC-samples or ADC-bursts and when the count has reached a user defined limit, this event can be used to generate interrupts or trigger stop situations. Thus, it is now possible to collect N-number of samples into the large FIFO memory and stop collecting after a given interval of time with little software overhead.
- Other improvements
  - FIFO status (see page 69) values are now properly latched and in addition, the block count will not incorrectly report values.
  - DIN3 and DIN1 (see page 96) the user can reverse the swapped positions of these inputs.
  - An additional high speed FIFO can be enabled (see page 112) between STX104 main memory and the ISA bus, further

reducing and/or eliminating wait state (IOCHRDY) conditions.

- All status registers are properly latched as well to prevent change in values during a bus read cycle.
- I/O Read line is digitally filtered to support noisy bus problems and eliminate the possibility of dropped analog input data.
- Indexed register array banked over the 8254 registers. This opens the door for many new functions as described above.
- Moving average filter has been corrected to present channel data in the proper order.

Update as of January 15, 2008 (Revision 090115H)

- 8254 Counter/Timer is also available within the indexed register set. This allows software access to all registers when the indexed register set is enabled.

Update as of March 28, 2012 (Revision 120328H)

- Analog Inputs Bit pattern encoding (see page 103). This has been implemented to further reduce CPU post processing of data samples and provide maximum flexibility.
  - Two's complement or straight binary ADC output encoding.
  - Polarity (i.e. multiply by -1) of selected channels for both straight binary and two's complement encoding.
  - Absolute value is possible using two's complement encoding scheme.
- Analog output update latency has been reduced from 5 uS to 2.6 uS.

Update as of March 28, 2012 (Revision 120328H)

- Analog output update latency has been reduced from 5 uS to 2.6 uS.

Update as of October 23, 2016 (Revision 161023)

- DMA features have been removed
- Moving Average filter has been improved
- Status LED for CPLD

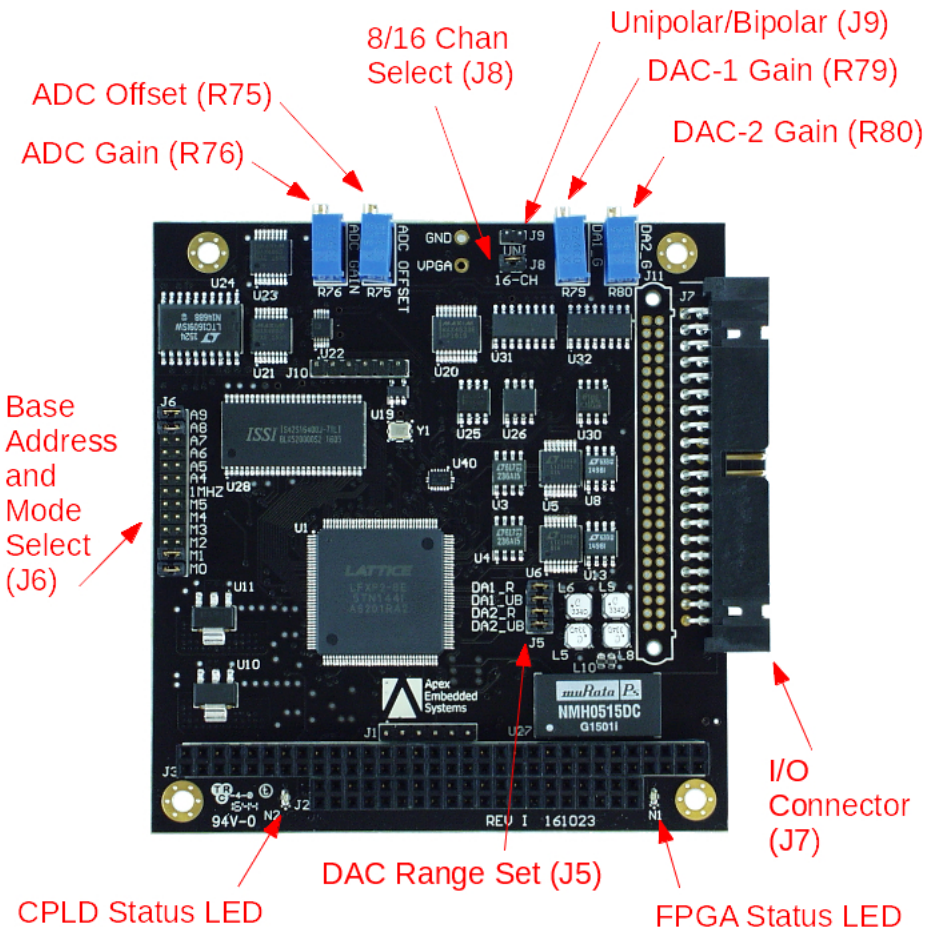
NEW REVISION INFORMATION (October 23, 2016 and beyond)

Date	PCB Date Code	CPLD Date Code	FPGA Date Code	Description
November 2, 2016	160923	161101	161102	Original release of firmware
December 21, 2016	160923	161221	161221	Recompiled revision
March 7, 2017	161023	161219	170307	FPGA has counter timer read back fixed
March 7, 2017	161023	161219	170307	FPGA has counter timer read back fixed
November 15, 2017	161023	161219	171115	FPGA has index pointer read back issue fixed and also has valid LED blink fixed

Please note that all of the new registers are designed such that if they are not configured, everything defaults to the classic modes of operation. Thus, existing software will function without modification. Writing values to the new registers enhances operations.

3

3.2 Photo



NOTES:

J1 is CPLD JTAG connector used during manufacturing only.

J10 is FPGA JTAG connector used during manufacturing only.

J11 is an optional alternate pin to wire I/O connector in 2mm format. Contact factory for further details.

4 Errata

Addresses known board related issues and includes methods to work around the issues.

Description

When the STX104 is operating in DAS16Jr/16 or DAS1602 mode the ADC Status Register bit 5 reports back a 0 for 16 single ended and a 1 for differential mode. The original DAS16Jr/16 or DAS1602 boards report a 1 for single ended and 0 for differential mode.

--- end

5 ESD Caution

A discharge of static electricity from your hands can seriously damage certain electrical components on any circuit board. Before handling any board, discharge static electricity from yourself by touching a grounded conductor such as your computer chassis (your computer must be turned off). Whenever you handle a board, hold it by the edges and avoid touching any board components or cable connectors.

6 PC/104 Insertion Caution

Before powering up the PC/104 stack... 

and look for proper PC/104 connector alignment.

This simple step will prevent permanent board damage.

Helpful hint: During system prototyping install the spacers to help guide installation and provide another means of checking board alignment. We recommend having the bolt end of the spacer facing up to act as a guide or alignment pin.

7 Definitions

Definition of terms.

7.1 ADC-Sample

Analog Input sample or data from an individual analog input channel.

7.2 ADC-Burst

A group of analog input samples defined by the number of channels for a burst operation. When an ADC-burst is started, the timing between samples is controlled by the Analog Input Burst Timer (see page 105) or in classic mode it is defined strictly as 5 microseconds.

7.3 Analog Input Sample Event

Analog Input Sample Event is a signal that causes either an analog input sample or analog input burst, depending on how the STX104 is set up.

7.4 Frame

A Frame is a generalized term. This is the record that is deposited per frame time into the FIFO memory. Currently, a frame is the equivalent to an ADC-Burst in terms of the amount of samples stored in the FIFO.

7.5 Intra-Sample

Intra-sample refers to time between analog input samples within an ADC-burst operation.

See Also

Analog Input Burst Timer (see page 105)

8 Analog Inputs

8.1 Calibration

Description

Assumption: Using default legacy Straight-Binary data encoding (data format).

Perform the following procedure to adjust ADC input. We recommend averaging ADC values over at least 256 samples in order to provide the most accurate calibration possible, reference the Voltage Input Conversion section under the ADC Data Register (see page 48) description. Full scale or gain calibration relies on the use of an external precision voltage source (or one can use a DAC output for this purpose as well).

1. Select the desired input mode by configuring J8 for differential or single-ended inputs.
2. Select the desired input range by configuring J9 for unipolar or bipolar inputs.
3. Calibration of potentiometer PGA_OFF (R3), if present. If R3 is not present, then skip to step #4 below.
    1. Short J7 pin 36 to J7 pin 37. In the case of differential inputs (J8 not stuffed), also short J7 pin 35 to J7 pin 37.
    2. Using a voltmeter connected to the two test points VPGA and AGND, adjust R3 to obtain zero volts (+/- 1 LSB).
4. Set the desired gain by writing to the ADC Configuration Register (see page 72) (base_address + 11) to the desired ADC input gain.
5. Set the ADC Channel Register (see page 54) (base_address + 2) to zero so that we are observing input channel zero.
6. Offset Adjustment.
    1. In the case of single-ended inputs (jumper J8 installed), short J7 pin 36 to J7 pin 37; effectively connecting the single-ended input to analog ground. In the case of differential inputs (jumper J8 not installed), short J7 pin 35 and pin 36 to J7 pin 37; effectively connecting the differential input to analog ground.
    2. While performing ADC conversions and reading the ADC value, adjust potentiometer ADC_OFFSET (R4) until the nearest zero value can be achieved, typically within one ADC step or LSB.
7. Remove all jumpers from the I/O connector at J7.
8. Gain Adjustment.
    1. In the case of single-ended inputs (jumper J8 installed), connect J7 pin 36 to the full scale voltage source. In the case of differential inputs (jumper J8 not installed), connect J7 pin 35 to J7 pin 37 and J7 pin 36 to the full scale voltage source. Adjust the DAC output for ADC full-scale input minus two LSB. Measure this value using a calibrated voltmeter to the accuracy you require.
    2. While performing ADC conversions and reading the ADC value, adjust potentiometer ADC_GAIN (R5) until full scale minus two LSB is achieved. This can be achieved through averaging ADC values. Please reference ADC Conversion Register for software example on how to average data.

9. Remove all jumpers from the I/O connector at J7.
10. Repeat steps 5 through 8 until the ADC input is calibration to within the accuracy you need or +/- 2 LSB.

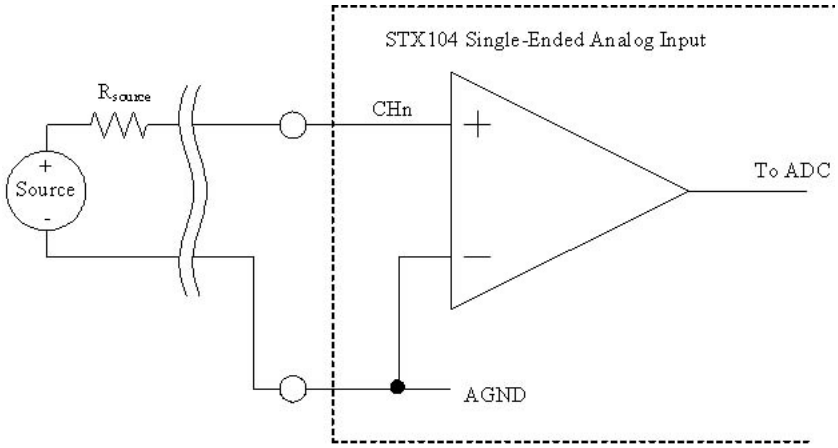
8.2 Connectivity

8.2.1 Single-Ended

Single-ended analog inputs.

Description

A single-ended input channel consists of one input. The output is the difference between the input and AGND (common).



Jumper Setup

J8 jumper must be installed for single-ended mode of operation. This also configures the STX104 card for sixteen analog input channels.

Amplifier Bias Currents

The primary path of amplifier bias currents is the AGND (or common) and since the source is tied to AGND amplifier bias currents flow through the sensor to AGND. Amplifier bias currents are typically on the order of tens of nano-amps.

Source Resistance

Source resistance Rsource along with cable and input parasitic capacitances must be taken into considered to determine minimum settling time. More details can be found here (see page 105).

Input Ranges

The input ranges for single-ended inputs.

SINGLE-ENDED INPUT RANGE (CHn)	RESOLUTION	UNIPOLAR/BIPOLAR (J9)	ACFG.ADBU	ACFG.G1	ACFG.G0
+/- 10 V	305 uV	J9 not installed	0	0	0
+/- 5 V	153 uV	J9 not installed	0	0	1
+/- 2.50 V	76 uV	J9 not installed	0	1	0
+/- 1.25 V	38 uV	J9 not installed	0	1	1
0 to +10 V	153 uV	J9 installed	1	0	0
0 to +5 V	76 uV	J9 installed	1	0	1
0 to +2.5 V	38 uV	J9 installed	1	1	0
0 to +1.25 V	19 uV	J9 installed	1	1	1

NOTE: Analog Configuration Register (ACFG) (see page 72)

Advantages

- More analog input channels available
- Less wiring and lower cost

Disadvantages

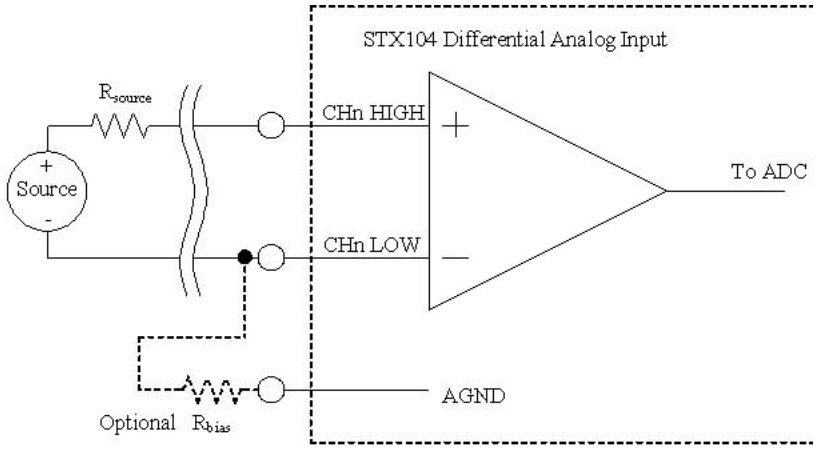
- Since all inputs are referred to AGND (i.e. common) the possibility exists for ground loops or voltage drops along common connections which will add to noise or measurement errors.

8.2.2 Differential

Full differential analog inputs.

Description

A differential analog input channel consists of two inputs, high side (positive) and low side (negative). The output measured is the difference between the two inputs, thus rejecting signals that are common to both inputs. In many situations this offers additional noise immunity to the signal of interest. The image below illustrates how to interface a sensor to a differential input channel.



Jumper Setup

J8 jumper must be removed for differential mode of operation. This also configures the STX104 card for eight analog input channels.

Optional Rbias

All analog inputs require a small amount of bias current, on the order of tens of nano-amps. If the sensor or signal source is not referenced to the STX104 AGND or system power supply then a resistor must be tied from one of the differential leads to AGND. The resistor value range is typically 100K ohms down to 100 ohms, depending on the application and the actual input bias current required. In most cases a 10K ohm resistor will do and will add less than +/-50uV of common-mode due to the bias current. Actually, the R_{bias} value could be as low as zero ohms, but using zero ohms you are reverting to single-ended behavior.

Source Resistance

Source resistance Rsource along with cable and input parasitic capacitances must be taken into considered to determine minimum settling time. More details can be found here (see page 105).

Input Ranges

The input ranges for each half of the differential input along with the full differential input range.

DIFFERENTIAL INPUT RANGE ((CHn High) - (CHn Low))	RESOLUTION	UNIPOLAR/BIPOLAR (J9)	ACFG.ADBU	ACFG.G1	ACFG.G0	CHn HIGH RANGE	CHn LOW RANGE
+/- 10 V	305 uV	J9 not installed (bipolar)	0	0	0	+/- 10 V	+/- 10 V
+/- 5 V	153 uV	J9 not installed (bipolar)	0	0	1	+/- 5 V	+/- 5 V
+/- 2.50 V	76 uV	J9 not installed (bipolar)	0	1	0	+/- 2.5 V	+/- 2.5 V

+/- 1.25 V	38 uV	J9 not installed (bipolar)	0	1	1	+/- 1.25 V	+/- 1.25 V
0 to +10 V	153 uV	J9 installed (unipolar)	1	0	0	0 - 10 V	0 - 10 V
0 to +5 V	76 uV	J9 installed (unipolar)	1	0	1	0 - 5 V	0 - 5 V
0 to +2.50 V	38 uV	J9 installed (unipolar)	1	1	0	0 - 2.5 V	0 - 2.5 V
0 to +1.25 V	19 uV	J9 installed (unipolar)	1	1	1	0 - 1.25 V	0 - 1.25 V

NOTE: Analog Configuration Register (ACFG) (see page 72)

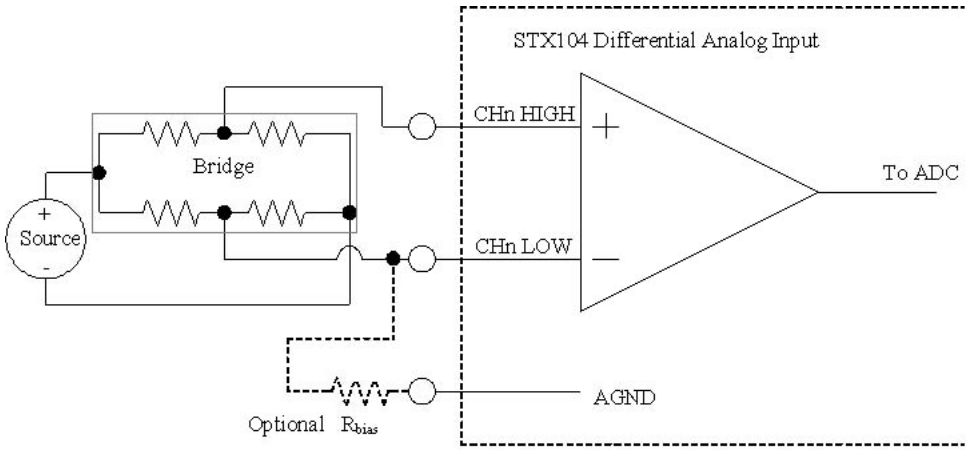
Input Total Range

Need some images or graph to illustrate more clearly what is happening. Need to illustrate how the unipolar 0-5V can slide up and down the -10V to +10V region and so on. How to illustrate this sliding around business. Maybe a CHn_High vs CHn_Low vs ADC value. Perhaps a 3-D plot where X=CHn_HIGH, Y=CHn_LOW, Z=ADC

ADC = (VH - VL);

Example Applications

To measure the output of a bridge network as shown in the diagram below. When the bridge is balanced, the common mode voltage at both the positive and negative inputs of the differential input are the same. Thus, the STX104 will report zero volts output (0x8000 in straight binary). Note that the voltage driving the bridge will be ratiometric at the outputs and thus nulled at the output.



If the negative side of the source driving the bridge is tied to the system supply that supplies power to the STX104, then the R_{bias} is not required because input bias currents have a complete DC path. If you are not sure whether your signal source is referenced to the STX104 AGND, power down the system, and using an ohmmeter measure the resistance between STX104 AGND and the common side of your signal source, you will want the resistance to be less than 10K ohms.

Advantages

- Common mode rejection
- Signals can be extracted even though there is a common mode voltage
- typically improves noise immunity
- Can perform ratiometric measurements

Disadvantages

- More wiring
- Fewer channels available

8

8.3 Data Acquisition Modes

Description

The following functions are available for each of the data acquisition modes:

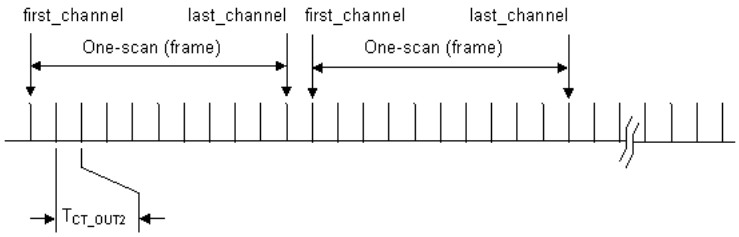
- The timing between samples and frames (i.e. ADC-burst) are adjustable to 25 nanosecond resolution via 32-bit timers.
- Analog Input Frame (see page 15) timing can be selected from a variety of sources including:
  - Internal Analog Input Frame (see page 15) Timer (AIFT) to 25ns resolution (50 ppm over temp).
  - External digital input (i.e. GPS pulses or other precision synchronizing timing sources)
- Triggering subsystem. Triggering starts and stops can originate from internal or external events.
- Interrupt subsystem. Variety of interrupts can be generated based on both internal and external events.
- Moving average filter.

8.3.1 Classic DAS16jr/16

- Single ADC-sample per analog input sample event.
- Analog input sampling events enhanced from 3 to 13 possible sources.
- Supports DMA (see note).

Description

The DAS16jr/16 mode is the simplest of all STX104 modes, and this mode is compatible with many other cards. In this case, as seen by the illustration below, sampling is equally spaced. If phase delay between channels is not important, this mode offers maximum settling time between channels.



NOTES:
(1) T_{CT_OUT2} = Intra-sample timing set by the 8254 Counter 1 and Counter 2. Minimum sample timing allowed is 5 microseconds.
(2) One scan or frame time will be the number of channels times T_{CT_OUT2}.
(3) Sample timing can be gated by GCTRL bit in the Pacer Clock Control Register along with the ALSS[1:0] bits found in the ADC Control Register.

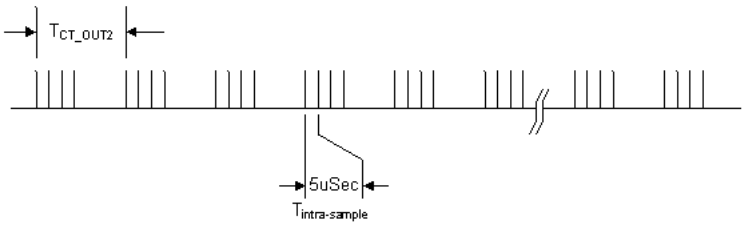
This mode typically uses the 8254 timer to set the sampling rate, however, other sampling sources can now be configured via the Analog Input General Configuration Register (see page 100). The frame counter (see page 109) will actually count samples rather than frames, and the Analog Input Frame Timer (see page 104) will produce the timing for each sample rather than each frame.

8.3.2 Classic DAS1602

- ADC-sample per analog input sample event.
- ADC-burst (one or more channels collected) per analog input sample event. Timing between ADC-samples within a burst are adjustable from 5 microseconds to 53 seconds to 25 nanosecond resolution.
- Analog input sampling events enhanced from 3 to 13 possible sources.
- Large FIFO depth allows for greater interrupt latency or readout latency.
- Supports DMA (see note).

Description

The DAS1602 mode is a step up from the DAS16jr/16 mode in that this mode additionally supports ADC-bursting. In other words, one sample strobe will produce and ADC-burst or one complete scan from the *first_channel (see page 54)* to the *last_channel (see page 54)*.



NOTES:
(1) T_{CT_OUT2} = frame (adc-burst) time set by the 8254 Counter 1 and Counter 2.
(2) $T_{intra\text{-}sample}$ = fixed to 5 microseconds only.
(3) One scan or frame time will be T_{CT_OUT2} or the number of channels times 5uSec which ever is largest.
(4) Sample timing can be gated by GCTRL bit in the Pacer Clock Control Register along with the ALSS[1:0] bits found in the ADC Control Register.

This mode typically uses the 8254 timer to set the sampling rate, however, other sampling sources can now be configured via the Analog Input General Configuration Register (see page 100). The frame counter (see page 109) will actually count samples rather than frames, and the Analog Input Frame Timer (see page 104) will produce the timing for each sample rather than each frame.

See Also

Conversion Disable (see page 84)

Burst Mode Enable (see page 85)

Burst Function Enable (see page 86)

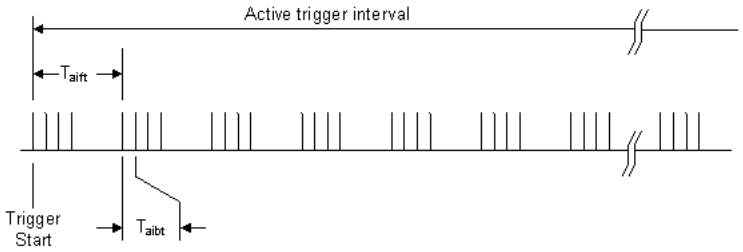
Extended Status (see page 86)

Analog Input Burst Timer (see page 105)

8.3.3 Continuous High Speed Sampling

- CPU read-out N-blocks of data per unit of time (i.e. CPU-burst reads)
- Sample timing and intra-sample timing in ADC-burst modes adjustable to 25 nanosecond resolution.
- FIFO interrupt generated interrupts and events are very flexible and support optimal CPU bursting (ultimately dependent on other CPU overhead or rhythmic behavior such as flash drives).
- Data fragment buffer reduces PC/104 generated IOCHRDY or wait conditions, further enhancing PC/104 data bus throughput.
- Large FIFO depth enhances the time decoupling between CPU activities and precise sample timing.

Description



NOTES:
(1) T_{aift} = frame (adc-burst) time set by the Analog Input Frame Timer Register
(2) T_{aibt} = intra-sample time set by the Analog Input Burst Timer Register
(3) In this illustration the Analog Input Frame Maximum is set to 4 frames (i.e. ADC-bursts).
(3) Trigger start selected by STS[3:0] bits in the Trigger Configuration Register.
(4) Trigger stop selected by ETS[3:0] bits in the Trigger Configuration Register.
In this illustration, trigger stop is selected to be the Analog Input Frame Maximum.
(5) It is possible to delay sampling after trigger start by introducing Trigger Delay.
(6) In this illustration the Analog Input Frame Timer and the Analog Input Frame Counter are synchronized to the trigger start (SAIFTTS='1' and SAIFCTS='1' in the Analog Input General Configuration Register).

See Also

FIFO Status MSB (see page 69)

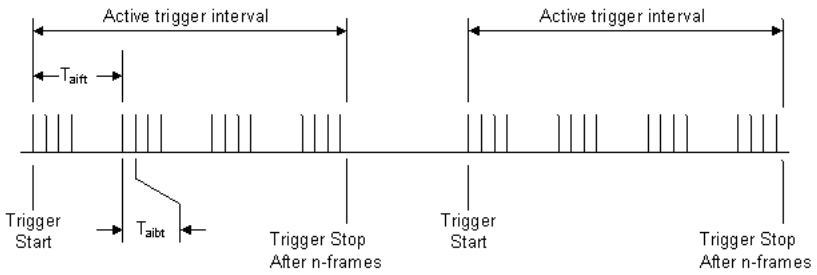
Interrupt Configuration (see page 90)

8.3.4 Start/Stop-Trigger Encased Frame Groups

- Start-Trigger to begin sampling, then Stop-Trigger to stop sampling. Repeat sequence.
- Large FIFO depth offers ability to collect large groups of data with a post-readout scheme.

Description

The most generalized sampling scheme is illustrated below.



NOTES:
(1) T_{aift} = frame (adc-burst) time set by the Analog Input Frame Timer Register
(2) T_{aibt} = intra-sample time set by the Analog Input Burst Timer Register
(3) In this illustration the Analog Input Frame Maximum is set to 4 frames (i.e. ADC-bursts).
(3) Trigger start selected by STS[3:0] bits in the Trigger Configuration Register.
(4) Trigger stop selected by ETS[3:0] bits in the Trigger Configuration Register.
In this illustration, trigger stop is selected to be the Analog Input Frame Maximum.
(5) It is possible to delay sampling after trigger start by introducing Trigger Delay.
(6) In this illustration the Analog Input Frame Timer and the Analog Input Frame Counter are synchronized to the trigger start (SAIFTTS='1' and SAIFCTS='1' in the Analog Input General Configuration Register).

See Also

Trigger Configuration (see page 97)

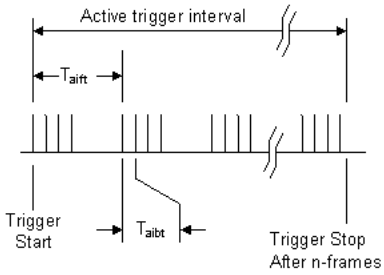
Analog Input Frame Maximum (see page 108)

8

8.3.5 N-Sample Collection

- Start-Trigger to begin sampling, stop sampling when N-samples/-Frames reached (i.e. Stop-Trigger event).
- Large FIFO depth offers set it and forget it methodology, thus drastically reducing CPU software monitoring overhead.

Description



NOTES:
(1) T_{aift} = frame (adc-burst) time set by the Analog Input Frame Timer Register
(2) T_{aibt} = intra-sample time set by the Analog Input Burst Timer Register
(3) In this illustration the Analog Input Frame Maximum is set to 4 frames (i.e. ADC-bursts).
(3) Trigger start selected by STS[3:0] bits in the Trigger Configuration Register.
(4) Trigger stop selected by ETS[3:0] bits in the Trigger Configuration Register.
In this illustration, trigger stop is selected to be the Analog Input Frame Maximum.
(5) It is possible to delay sampling after trigger start by introducing Trigger Delay.
(6) In this illustration the Analog Input Frame Timer and the Analog Input Frame Counter are synchronized to the trigger start (SAIFTTS='1' and SAIFCTS='1' in the Analog Input General Configuration Register).

See Also

Analog Input Frame Maximum (see page 108)

8

8.4 Analog Input Sample Timing

Description

The example below illustrates various methods of setting up analog input sample timing. There are two primary parameters needed to setup timing: time between frames, and time between samples (i.e. ADC-burst timing).

One must call the STX104_AI_Time_Settling_Minimum() function to determine the minimum timing possible between samples or the intra-sample timing or minimum ADC-burst timing.

The functions within the example code below will compute and configure the proper analog input timing registers based on firmware revision level. If only 8254 timers are used, then resolution is fixed to 1uSec. In the case of using the Analog Input Frame Timer (see page 104) and the Analog Input Burst Timer (see page 105), resolution is improved to 25 nanoseconds.

When using the triggering subsystem, to synchronize analog input sampling to the trigger-start event, set the SAIFTTS bit in the Analog Input General Configuration (see page 100) Register.

-

Method for configuring the Analog Input Frame Timer (see page 104) and the Analog Input Burst Timer (see page 105) registers.

a) Our assumption is that ADC-burst mode will always be enabled, and is acceptable for all situations.

b) First calculate the time between samples (i.e. intra-sample time) in the the ADC-burst mode. The timing can be as short as 5uSec or longer. The timing is based on multiplexer settling time.

If you have a sensor with a source resistance of 10Kohms and a cable with a total capacitance of 100pF, then the minimum time between samples will be approximately 11 * 10Kohms * 100pF = 11,000 nSec. In other words, the minimum settling time is eleven RC time constants.

ai_time_intra_sample_ns = AIBT_Register_Value * (25 nSec) + (5000 nSec).

AIBT_Register_Value = (ai_time_intra_sample_ns / (25 nSec)) - 200

c) The minimum frame time will be:

minimum_ai_time_frame_ns = (last_channel - first_channel + 1) * ai_time_intra_sample_ns.

d) The frame time can now be computed:

ai_time_frame_ns = (AIFT_Register_Value + 1) * (25 nSec)

AIFT_Register_Value = ai_time_frame_ns * (25 nSec) - 1

The above example is a summary of the example source code below.

Example

```
#define STX104_INTRA_SAMPLE_TIMING          5000  /* nanoseconds */
#define STX104_TIME_RESOLUTION              25    /* nanoseconds */


#define STX104_REVISION_080214   0x1008
#define STX104_REVISION_080407   0x1008
#define STX104_REVISION_090115   0x1009
```

```
/*##############################################################################
*                                                        ANALOG INPUT TIMING FUNCTIONS
*/

/*************************************************************
/                              ANALOG INPUT MINIMUM SETTLING TIME
*/
static unsigned long STX104_AI_Time_Settling_Minimum( float ai_capacitance_pf, float
ai_resistance_ohms )
{
    unsigned long settle_time_ns;

    //settle_time = ai_resistance_ohms * ai_capacitance_pf * pow10(-12) * 16 * ln(2) * pow10(9)
    settle_time_ns = (unsigned long) ( ai_resistance_ohms * ai_capacitance_pf * pow10(-3) * 16
* log(2) );

    return( settle_time_ns );
}
/*************************************************************
/                                       ANALOG INPUT 8254 TIMING
*/
static void STX104_AI_Timing_8254_Set( int board, long time_interval_ns )
{
    long high_count;
    long low_count;
    unsigned int octet;

    STX104_Set_Bank( board, 0 );

    /* assumes 10MHz clock (i.e. no 1MHz jumper) */
    low_count  = 10L; /* 1 microsecond intervals */

    high_count = time_interval_ns / 1000;
    while ( high_count > 65536L )
    {
        high_count = high_count >> 1;
        low_count  = low_count << 1;
    }
    while ( high_count < 2L )
    {
        high_count = high_count << 1;
        low_count  = low_count >> 1;
    }
    /* set counter/timer 2  */
    outportb( stx104_base_address[board] + STX104_CT_CONFIGURATION, 0xB4 );
    octet = ((unsigned int) high_count) & 0x00FF;
    outp( stx104_base_address[board] + STX104_CT2_DATA, octet );
    octet = ((unsigned int) high_count) >> 8;
    outp( stx104_base_address[board] + STX104_CT2_DATA, octet );
    /* set counter/timer 1  */
    outportb( stx104_base_address[board] + STX104_CT_CONFIGURATION, 0x74 );
    octet = ((unsigned int) low_count) & 0x00FF;
    outp( stx104_base_address[board] + STX104_CT1_DATA, octet );
    octet = ((unsigned int) low_count) >> 8;
    outp( stx104_base_address[board] + STX104_CT1_DATA, octet );
}

/*************************************************************
/                                    ANALOG INPUT GENERAL TIMING
*/
static void STX104_AI_Timing_Set( int board, unsigned long ai_time_frame_ns, unsigned long
ai_time_intra_sample_ns )
{
    unsigned int adc_burst_channel_count;
```

```
    switch ( stx104_revision[board] )
    {
       case STX104_REVISION_080214:  /* same as version 080407 */
       case STX104_REVISION_090115:
            STX104_Set_Bank( board, 1 );
            STX104_Write_Indexed_Data_Byte( board, STX104_BURST_FUNCTION_ENABLE_INDEXED, 0x40
);
            STX104_Write_Indexed_Data_Byte( board, STX104_BURST_MODE_ENABLE_INDEXED,     0x40
);
            STX104_Write_Indexed_Data_Byte( board, STX104_CONVERSION_DISABLE_INDEXED,    0x00
);
            adc_burst_channel_count = (unsigned char)( 0x0F & ( stx104_ai_channel_last[board]
- stx104_ai_channel_first[board] ) );
            adc_burst_channel_count = adc_burst_channel_count << 4;
            outp( stx104_base_address[board] + STX104_PACER_CLOCK_CONTROL,
adc_burst_channel_count );
            /* the following registers require the number of 25nSec intervals to achieve the
selected timing */
            STX104_Write_Indexed_Data_Dword( board, STX104_ANALOG_INPUT_FRAME_TIMER, (
ai_time_frame_ns/STX104_TIME_RESOLUTION ) );
            STX104_Write_Indexed_Data_Dword( board, STX104_ANALOG_INPUT_BURST_TIMER, (
(ai_time_intra_sample_ns - STX104_INTRA_SAMPLE_TIMING) / STX104_TIME_RESOLUTION ) );
            break;
        default:
            if ( ai_time_intra_sample_ns > STX104_INTRA_SAMPLE_TIMING /*nS*/ )
            { /* cannot use adc-burst mode */
                STX104_AI_Timing_8254_Set( board, ai_time_intra_sample_ns );
                outp( stx104_base_address[board] + STX104_BURST_FUNCTION,         0x00 );
                outp( stx104_base_address[board] + STX104_BURST_MODE_ENABLE,      0x00 );
                outp( stx104_base_address[board] + STX104_CONVERSION_DISABLE,     0x00 );
            }
            else
            { /* utilize adc-burst mode */
                STX104_AI_Timing_8254_Set( board, ai_time_frame_ns );
                outp( stx104_base_address[board] + STX104_BURST_FUNCTION,         0x40 );
                outp( stx104_base_address[board] + STX104_BURST_MODE_ENABLE,      0x40 );
                outp( stx104_base_address[board] + STX104_CONVERSION_DISABLE,     0x00 );
                adc_burst_channel_count = (unsigned char)( 0x0F & (
stx104_ai_channel_last[board] - stx104_ai_channel_first[board] ) );
                adc_burst_channel_count = adc_burst_channel_count << 4;
                outp( stx104_base_address[board] + STX104_PACER_CLOCK_CONTROL,
adc_burst_channel_count );
            }
    }
}
```

8.5 Triggering Subsystem

Description

The triggering subsystem is used to start and stop analog input sampling. This is a new generalized triggering system. Please refer to Trigger Configuration (see page 97) and Trigger Start Delay (see page 99) Registers for further details.

The Analog Input General Configuration (see page 100) Register bit nSGATE is used to enable triggering subsystem use. The

nSGATE bit is required in order to support classic modes without special configuration.

Further, the triggering subsystem must be enabled in order for it to function.

There are three start triggering sequences:

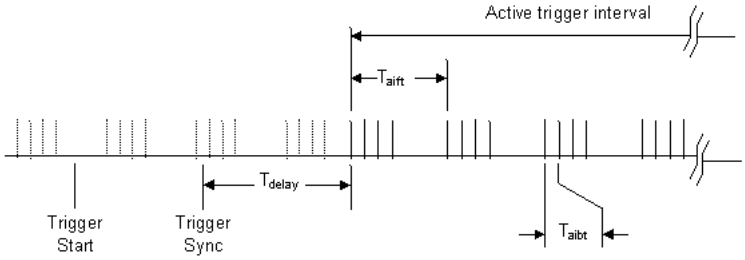
1. Start trigger source --> Analog Input Sampling Begins
2. Start trigger source --> Trigger Delay --> Analog Input Sampling Begins
3. Start trigger source --> Start Trigger Synchronization source --> Trigger Delay -- > Analog Input Sampling Begins.

Analog input sampling is stopped upon a valid End/Stop Trigger Source event.

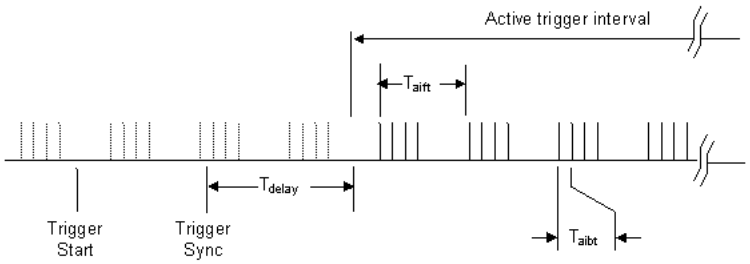
What is the purpose of the triggering subsystem? It allows precise control as to when analog input sampling can occur. It is possible to synchronize sampling to 60Hz line source or whatever is chosen. It is possible to configure the STX104 to trigger on an event and accumulate N-samples and then stop sampling.

Synchronization of Analog Input Frame Timer to Trigger Start (SAIFTTS)

The diagrams below illustrate how sample timing is affected by the SAIFTTS bit found in the Analog Input General Configuration Register.



NOTES:
(1) SAIFTTS='1' which means that the analog input frame timer is synchronized to the beginning of the active trigger. of triggering.
The triggering acts essentially as a mechanism to gate the analog input frame timer.
(2) The greyed sampling shows the timing from the previously triggered sampling interval and how the timing is affected.



NOTES:
(1) SAIFTTS='0' which means that the frame timer runs independent of triggering. The triggering acts essentially as a mechanism to gate the analog input frame timer.
(2) The grayed sampling shows the timing from the previously triggered sampling interval and how frame timing is maintained independent of the trigger.

See Also

Trigger Configuration (see page 97)

8

8.6 Moving Average Filter

Description

Installing jumper M3 enables the 16-sample moving average filter for all channels. The filter can be reset (or cleared) by writing to the Channel Register. The moving average filter is enabled for all channels and operates completely transparent to ADC acquisition modes. The ADC values read out will be the current sample plus the last fifteen samples summed together and divided by sixteen (average of sixteen ADC samples). It is important to recognize that after the filter is reset or at the beginning of data sampling that it may require at least 16-data samples per channel be taken until the data becomes current. In other words, there is an inherent 16-sample delay in the ADC data that is read out of the ADC Data Register.

8.7 CPU Readout Methods

8.7.1 DMA Read

Analog input data is read from the FIFO memory via Direct Memory Access (DMA). This section only applies to Revision 120328 and prior, Revision 161023 and after does NOT support DMA.

Description

We strongly encourage you to use the REP INSW instruction (or Insw() function in Linux) as it performs better than DMA and is substantially simpler to set up and use. Note that DMA performs I/O to memory transfers a byte at a time (DMA 1 and DMA 3 are byte wide transfers) while REP INSW performs I/O to memory transfers a word at a time. If you are designing a real-time system, you will have better timing control over your system by using REP INSW instruction over DMA. Since processor generated I/O cycles are faster than DMA generated cycles (typically 350ns versus 800ns), data transfer can take place faster than DMA. Note that 8-bit bus transactions (including DMA) are typically at least twice as long as 16-bit bus transactions; this alone is reason enough to avoid DMA.

When DMA is enabled, the 1 mega-sample FIFO is used.

In DMA mode, receiving a terminal count from the CPU will generate an interrupt, if the interrupt is enabled. If you are in DAS1602 mode (jumper M0 is installed), receiving a terminal count will set the Conversion Disable bit to false, thus disabling any additional ADC sampling. Writing 0x00 to the Conversion Disable Register (see page 84) will allow ADC sampling to continue.

In order to use DMA, you must set up the computer's DMA controller and page registers before enabling DMA on the STX104 board.

We consider DMA a deprecated function and in the future should be avoided; it just no longer makes any sense to use.

8.7.2 Burst Read

Reading multiple samples one after the other reading out blocks or frames (i.e. ADC-burst) of data in a short interval of time. This is typically done using the REP INSW or Insw() type functions. Refer to the ADC Data Register (see page 48) for an example.

8.7.3 Single Read

Reading an individual sample using either an 8- or 16-bit read cycle. Refer to the the ADC Data Register (see page 48) for examples.

9 Analog Outputs

9.1 Calibration

Description

Hardware configuration:

1. Select the desired output range by adjusting jumper settings at J5.
2. Install jumper M2 as required for the application (this jumper selects 16-bit versus 12-bit resolution).

Calibration of DAC Channel-A:

1. Write 65535 (0xFFFF) to Register DAC Channel-A (see page 58).
2. Adjust potentiometer DA1_G (R2) until the desired full scale is reached.

Calibration of DAC Channel-B:

1. Write 65535 (0xFFFF) to Register DAC Channel-B (see page 61).
2. Adjust potentiometer DA2_G (R1) until the desired full scale is reached.

9.2 Connectivity

Description

Each DAC channel is a single-ended type output. We suggest wiring the return path back to one of the AGND pins at connector J7.

10 Hardware Configuration

10.1 Base Address Table

Card Base Address set via installing jumpers at J6 positions A9, A8, A7, A6, A5 and A4. Installing jumper M4 will reduce address I/O decode from a full 16-bits to 10-bits, this provides compatibility with CPU cards that only offer 10-address bits for I/O transactions.

Description

BASE ADDRESS Full 16-bit Address Decode M4 Jumper *not* Installed	BASE ADDRESS 10-bit Address Decode M4 Jumper Installed	A9	A8	A7	A6	A5	A4
0x0220	0x220	1	0	0	0	1	0
0x0240	0x240	1	0	0	1	0	0
0x0250	0x250	1	0	0	1	0	1
0x0260	0x260	1	0	0	1	1	0
0x0280	0x280	1	0	1	0	0	0
0x0290	0x290	1	0	1	0	0	1
0x02A0	0x2A0	1	0	1	0	1	0
0x02B0	0x2B0	1	0	1	0	1	1
0x02C0	0x2C0	1	0	1	1	0	0
0x02D0	0x2D0	1	0	1	1	0	1
0x02E0	0x2E0	1	0	1	1	1	0
0x02F0	0x2F0	1	0	1	1	1	1
0x0300*	0x300*	1	1	0	0	0	0
0x0310	0x310	1	1	0	0	0	1
0x0320	0x320	1	1	0	0	1	0
0x0330	0x330	1	1	0	0	1	1
0x0340	0x340	1	1	0	1	0	0
0x0350	0x350	1	1	0	1	0	1
0x0360	0x360	1	1	0	1	1	0
0x0370	0x370	1	1	0	1	1	1
0x0380	0x380	1	1	1	0	0	0
0x0390	0x390	1	1	1	0	0	1

0x03A0	0x3A0	1	1	1	0	1	0
0x03B0	0x3B0	1	1	1	0	1	1
0x03C0	0x3C0	1	1	1	1	0	0
0x03D0	0x3D0	1	1	1	1	0	1
0x03E0	0x3E0	1	1	1	1	1	0
0x03F0	0x3F0	1	1	1	1	1	1

Notes: 1 = Jumper installed, 0 = Jumper not installed.

* Factory Default

10.2 Compatibility Selection and Extended Functions

Description

10-BIT ADDRESS DECODE

The STX104 now supports CPU cards which only present the first 10 address lines. Normally, the STX104 will decode all 11 address bits in order to fully decode an I/O command. For 10-bit address decode, the upper 6 address bits are ignored. A 10-bit address decode provides a typical I/O address space from 0x000 to 0x3FF. The DAS1602 burst registers are now alternatively available through indexed register array access.

DAS16jr/16 and DAS1602 LEGACY COMPATIBILITY

The STX104 is compatible with DAS16jr/16 and the DAS1602. The DAS16jr/16 mode supports 8-bit or 16-bit systems (XT or AT, respectively). The enhanced register set now supports redirection of the DAS1602 extended burst registers to the indexed array register set thus allowing more addressing and data width options including both XT and AT compatibility. Jumper position M0 selects one of these compatibility modes.

The DAS1602 compatibility offers ADC-sample (see page 15) and ADC-bursting (see page 15).

DAS16jr/12 LEGACY COMPATIBILITY

The STX104 is compatible with DAS16jr/12. The DAS16jr/12 mode supports 8-bit or 16-bit systems (XT or AT, respectively). To achieve this configuration place a Mode Jumper on M5 and remove mode jumpers from M1 and M0. Do not place a jumper on J9

FIFO SUPERSET

Data FIFO superset functionality can be selected for either the DAS16jr/16 or the DAS1602 compatibility modes. The FIFO superset functionality is very similar to other cards on the market. Stuffing jumper position M1 enables FIFO superset functionality. Note that for the DAS1602 compatibility mode, the 1 mega-sample FIFO is enabled, however to maintain register compatibility the FIFO status registers are only visible once the M1 jumper is installed.

The HSFIFOEN bit found in the FIFO Configuration (see page 112) register can be used to enable a 2048 sample high speed pre-queueing FIFO buffer between STX104 main memory and the ISA bus. In general, this will reduce/eliminate IOCHRDY wait states and improve average system throughput.

12-/16-BIT DAC MODE

The 16-bit DAC mode allows the DAC registers to provide 12-bit legacy functionality as well as full 16-bit DAC functionality. Installing jumper M2 allows for full 16-bit DAC operation.

ADC 16-SAMPLE MOVING AVERAGE FILTER

Installing jumper M3 enables the 16-sample moving average filter for all channels. The filter can be reset (or cleared) by writing to the Channel Register. The moving average filter is enabled for all channels and operates completely transparent to ADC acquisition modes. The ADC values read out will be the current sample plus the last fifteen samples summed together and divided by sixteen (average of sixteen ADC samples). It is important to recognize that after the filter is reset or at the beginning of data sampling that it may require at least 16-data samples per channel be taken until the data becomes current. In other words, there is an inherent 16-sample delay in the ADC data that is read out of the ADC Data Register.

MODE	M5	M4	M3	M2	M1	M0
10-Bit Address Decode (A0 through A9 are decoded while A10 to A15 ignored. DAS1602 burst registers available in the indexed register array). If jumper is removed A0 through A10 are decoded while A11 to A15 ignored.	0	1	X	X	X	X
DAS16jr/16 + DAC02 Compatibility	0	X	X	X	0	0
DAS1602 Compatibility w/1MegaSample FIFO (Single interrupt per sample during ADC-burst)	0	X	X	X	0	1
DAS1602 Compatibility and FIFO Status Registers Visible (Single interrupt per ADC-burst completion)	0	X	X	X	1	1
Enable FIFO Superset Functionality	0	X	X	X	1	X
16-bit DAC Registers	X	X	X	1	X	X
16-Sample moving average filter for all 8 or 16 ADC channels	X	X	1	X	X	X
DAS16jr/12	1	X	X	X	0	0

Note: 1 = Jumper installed, 0 = Jumper not installed.

10.3 CPU Limitation Accommodations

Description

The STX104 can support the following CPU card variations:

- 10- or 11-bit Address decoding
- 8- or 16-bit PC/104 data bus (without or with 40-pin connector, respectively).

10.4 PC104-DAS16jr/12

Description

Differences Between STX104 and PC104-DAS16Jr/12

1. The STX104 has no DMA functions.

2. The STX104 has the 40 pin connector rotated with pin 1 being away from the PC104 Connector and away from the card edge. The PC104-DAS16jr/12 cards have pin 1 being near the PC104 Connector and close to the card edge.

3. The STX104 supplies -5V on pin 15 of the 40 pin connector where the PC104-DAS16jr/12 has this pin as a no connect.

4. The STX104 allows 16 bit bus transactions

5. The STX104 channel register(offset 2) has the default value set as 0x00 and the current channel is set to 0 as the default. If the channel low bits are set to a higher value then the channel high bits, the STX104 does not wrap around. If set the channel register(offset 2) is set to 0x03, it will only sample channel 3.

The PC104-DAS16Jr/12 channel register(offset 2) has the default set as 0x03. When this register is written to, the value located in bits 0-3 is set as the current channel. The exception is the default value. Since this value is not set by the user, channel 3 is not set as the current channel and instead the current channel is 0. For this card, when the channel low is set to a lower channel higher then the channel high, it wraps around. For the default value, the channels sample in the following order: 0,3,4,5,6,7,8,9,10,11,12,13,14,15. If the user had written the value 0x03 to the channel register (offset 2), then the channels sample in the following order 3,4,5,6,7,8,9,10,11,12,13,14,15,0.

6. The PC104-DAS16Jr/12 has a 512 sample FIFO, while the STX104 has 1 Million sample FIFO.

7. The ADC sample limit of the STX104 200K samples per second. The PC104-DAS16Jr/12 can sample at 330K samples per second.

8. When changing the gains on the STX104 should be followed by rewriting or updating the channel register. This resets everything and allows the correct data to flow. If this is not done, the first sample of adc data might be sampled in the previous

gain setting. When setting gains on the STX104, we recommend allowing a few hundred mircoseconds for the board to complete this change, if this is not done then the first few samples might have skewed data.

9. On the STX104 the CNV goes active while the ADC is busy but it also goes active when writing to the channel register (offset 2). The PC104-DAS16Jr/12 only goes active while the ADC is busy.

10. On the STX104, the digital outputs and Counter Timer outputs only go to 3.3 volts. When unprogrammed, the Counter Timer outputs on the STX104 are low. The outputs on the PC104-DAS16Jr/12 are high.

11.Upon a bus reset the STX104 8254 timer will reset. The PC104-DAS16Jr/12 8254 does not reset.

11 Register Set

11.1 Summary

Overview of the STX104 register set.

Description

Register Name	NOTE	Mnemonic	Size	Direction	Offset	Index	Bank (acr.rb)
Software Strobe (see page 46)	(3)(4)	ssr	BYTE	w	0	-	X
ADC Data LSB (see page 47)		adl	BYTE	r	0	-	X
ADC Data MSB (see page 48)		adh	BYTE	r	1	-	X
ADC Data (see page 48)		ad	WORD	r	0	-	X
ADC Channel (see page 54)		achan	BYTE	rw	2	-	X
Digital Outputs (see page 56)		do	BYTE	w	3	-	X
Digital Inputs (see page 57)		di	BYTE	r	3	-	X
DAC Channel-A LSB (see page 58)		dacal	BYTE	w	4	-	X
DAC Channel-B MSB (see page 61)		dacah	BYTE	w	5	-	X
DAC Channel-A (see page 58)		daca	WORD	w	4	-	X
DAC Channel-B LSB (see page 60)		dacbl	BYTE	w	6	-	X
DAC Channel-B MSB (see page 61)		dacbh	BYTE	w	7	-	X
DAC Channel-B (see page 61)		dacb	WORD	w	6	-	X
Clear Interrupts (see page 63)		cir	BYTE	w	8	-	X
ADC Status (see page 64)	(4)	asr	BYTE	r	8	-	X
ADC Control (see page 66)	(4)	acr	BYTE	rw	9	-	X
Pacer Clock Control (see page 68)		pccr	BYTE	w	10	-	X
FIFO Status MSB (see page 69)	(2)(4)	fsh	BYTE	r	10	-	X
ADC Configuration (see page 72)	(4)	acfg	BYTE	rw	11	-	X
8254 CT0 Data (see page 77)		ct0d	BYTE	rw	12	-	0
8254 CT1 Data (see page 77)		ct1d	BYTE	rw	13	-	0
8254 CT2 Data (see page 77)		ct2d	BYTE	rw	14	-	0
8254 Configuration (see page 77)		ctcfg	BYTE	w	15	-	0
FIFO Status LSB (see page 80)	(2)	fsl	BYTE	r	15	-	X

Index Data LSB (see page 80)	(5)	idl	BYTE	rw	12	-	1
Index Data MSB (see page 80)	(5)	idh	BYTE	rw	13	-	1
Index Data (see page 81)	(5)	data	WORD	rw	12	-	1
Index Pointer (see page 81)	(5)	index	BYTE	rw	14	-	1
::::::::: DAS1602 Registers Section :::::::::							
Conversion Disable (see page 84)	(1)	cdr	BYTE	w	1028	-	X
Burst Mode Enable (see page 85)	(1)	bmer	BYTE	w	1029	-	X
Burst Function Enable (see page 86)	(1)	bfer	BYTE	w	1030	-	X
Extended Status (see page 86)	(1)	esr	BYTE	r	1031	-	X
:::::::::::::::::: General Section ::::::::::::::::::							
General Configuration (see page 87)	(5)	gcfg	BYTE	rw	-	0	1
Interrupt Source Select (see page 89)	(5)	iss	BYTE	rw	-	2	1
Interrupt Configuration (see page 90)	(5)	icfg	WORD	rw	-	4	1
Interrupt Threshold (see page 92)	(5)	ith	DWORD	rw	-	8	1
Digital Output Configuration (see page 94)	(5)(7)	doc	WORD	rw	-	12	1
Digital Input Configuration (see page 96)	(5)	dic	BYTE	rw	-	14	1
:::::::::::::::::: Triggering Section ::::::::::::::::::							
Trigger Configuration (see page 97)	(5)	tcfg	WORD	rw	-	16	1
Trigger Start Delay (see page 99)	(5)	tsd	DWORD	rw	-	20	1
:::::::::::::::::: Analog Input Section ::::::::::::::::::							
Analog Input General Configuration (see page 100)	(5)(7)	aigc	WORD	rw	-	32	1
Analog Input Polarity (see page 103)	(7)	aipol	WORD	rw	-	34	1
Analog Input Frame Timer (see page 104)	(5)	aift	DWORD	rw	-	36	1
Analog Input Burst Timer (see page 105)	(5)	aibt	DWORD	rw	-	40	1
Analog Input Frame Maximum (see page 108)	(5)	aifm	DWORD	rw	-	44	1
Analog Input Frame Counter (see page 109)	(5)	aifc	DWORD	r	-	48	1
::::: DAS1602 Redirected Registers Section :::::							
Conversion Disable (see page 84)	(1)(5)	icdr	BYTE	w	-	64	1
Burst Mode Enable (see page 85)	(1)(5)	ibmer	BYTE	w	-	65	1
Burst Function Enable (see page 86)	(1)(5)	ibfer	BYTE	w	-	66	1
Extended Status (see page 86)	(1)(5)	iesr	BYTE	r	-	67	1
::::::::: 8254 Redirected Registers Section :::::::::							
8254 CT0 Data (see page 77)	(6)	ict0d	BYTE	rw	-	68	1
8254 CT1 Data (see page 77)	(6)	ict1d	BYTE	rw	-	69	1
8254 CT2 Data (see page 77)	(6)	ict2d	BYTE	rw	-	70	1
8254 Configuration (see page 77)	(6)	ictcfg	BYTE	w	-	72	1
::::::::: SPI Master Module Section :::::::::							

SPI Configuration	(7)	spicfg	WORD	rw	-	80	1
SPI Half Clock Interval	(7)	spihclk	WORD	rw	-	82	1
SPI Receive FIFO Data	(7)	spirx	BYTE	r	-	84	1
SPI Transmit FIFO Data	(7)	spitx	BYTE	w	-	84	1
SPI Status	(7)	spistat	BYTE	r	-	85	1
SPI Software Chip Select (see page 43)	(7)	spics	BYTE	w	-	85	1
:::::::::::::::::: System Section ::::::::::::::::::							
Miscellaneous Output Configuration (see page 110)	(5)	mocr	BYTE	rw	-	208	1
FIFO Data Available (see page 111)	(5)	fda	DWORD	r	-	224	1
FIFO Configuration (see page 112)	(5)	fcfg	BYTE	rw	-	228	1
Reserved	-	-	BYTE	rw	-	240	1
Reserved	-	-	DWORD	rw	-	244	1
Reserved	-	-	BYTE	rw	-	242	1
Scratch Pad (see page 113)	(5)	scr	WORD	rw	-	248	1
Board ID (see page 114)	(5)	bid	WORD	r	-	250	1
Reserved	-	-	DWORD	-	-	252	1

(1) Available only when jumper M0 is installed.

(2) Available only when jumper M1 is installed.

(3) Renamed register to reflect generalizations.

(4) Additional functionality added

(5) New register as of February 14, 2008

(6) New register as of January 15, 2009

(7) New register or functionality added as of October 21, 2011

Example

```
/* STX104 Register Set Definitions */
#define STX104_SOFTWARE_STROBE                  0   /* offset (to be added to the
base address) */
#define STX104_ADC_DATA_LSB                     0
#define STX104_ADC_DATA_MSB                     1
#define STX104_ADC_DATA                         0
#define STX104_CHANNEL                          2
#define STX104_DIGITAL_OUTPUTS                  3
#define STX104_DIGITAL_INPUTS                   3
#define STX104_DAC_CHANA_LSB                    4
#define STX104_DAC_CHANA_MSB                    5
#define STX104_DAC_CHANA                        4
#define STX104_DAC_CHANB_LSB                    6
#define STX104_DAC_CHANB_MSB                    7
#define STX104_DAC_CHANB,DACB                   6
#define STX104_CLEAR_INTERRUPTS                 8
#define STX104_ADC_STATUS                       8
#define STX104_ADC_CONTROL                      9
#define STX104_PACER_CLOCK_CONTROL              10
```

```
#define STX104_FIFO_FLAGS                          10
#define STX104_ADC_CONFIGURATION                   11
#define STX104_CT0_DATA                            12
#define STX104_CT1_DATA                            13
#define STX104_CT2_DATA                            14
#define STX104_CT_CONFIGURATION                    15
#define STX104_FIFO_DATA_STATUS                    15
#define STX104_CONVERSION_DISABLE                  1028
#define STX104_BURST_MODE_ENABLE                   1029
#define STX104_BURST_FUNCTION                      1030
#define STX104_EXTENDED_STATUS                     1031
/* Applicable to Firmware Revision 080214H and beyond    */
#define STX104_INDEXED_DATA_LSB                    12  /* offset (to be added to the
base address) */
#define STX104_INDEXED_DATA_MSB                    13
#define STX104_INDEX_DATA                          12
#define STX104_INDEX_POINTER                       14
// indexed register array
#define STX104_GENERAL_CONFIGURATION               0   /* index */
#define STX104_INTERRUPT_SOURCE_SELECT             2
#define STX104_INTERRUPT_CONFIGURATION             4
#define STX104_INTERRUPT_THRESHOLD                 8
#define STX104_DIGITAL_OUTPUT_CONFIGURATION        12
#define STX104_DIGITAL_INPUT_CONFIGURATION         14
#define STX104_TRIGGER_CONFIGURATION               16
#define STX104_TRIGGER_START_DELAY                 20
#define STX104_ANALOG_INPUT_GENERAL_CONFIG         32
#define STX104_ANALOG_INPUT_FRAME_TIMER            36
#define STX104_ANALOG_INPUT_BURST_TIMER            40
#define STX104_ANALOG_INPUT_FRAME_MAX              44
#define STX104_ANALOG_INPUT_FRAME_COUNT            48
#define STX104_CONVERSION_DISABLE_INDEXED          64
#define STX104_BURST_MODE_ENABLE_INDEXED           65
#define STX104_BURST_FUNCTION_ENABLE_INDEXED       66
#define STX104_EXTENDED_STATUS_INDEXED             67
#define STX104_CT0_DATA_INDEXED                    68
#define STX104_CT1_DATA_INDEXED                    69
#define STX104_CT2_DATA_INDEXED                    70
#define STX104_CT_CONFIGURATION_INDEXED            71
#define STX104_MISCELLANEOUS_OUTPUT_CONFIG         208
#define STX104_FIFO_DATA_AVAILABLE                 224
#define STX104_FIFO_CONFIGURATION                  228
#define STX104_SCRATCH_PAD                         248
#define STX104_BOARD_ID                            250
```


11.2 Software Strobe (Offset=0)

Software Strobe Register. Expanded version of the ADC Software Trigger Register.

Register Layout

Offset = 0.

D7	D6	D5	D4	D3	D2	D1	D0
SSR7	SSR6	SSR5	SSR4	SSR3	SSR2	SSR1	SSR0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
SSR[7:0]	w	-	Software Strobe Register

Description

if triggering is disabled (TEN='0') then any value written to the Software Strobe Register will cause a ADC-sample or ADC-burst can occur depending on configuration.

if triggering is enabled (TEN='1') then specific values written to the Software Stobe Register will cause either an ADC-sample or ADC-burst or trigger signalling as shown below.

DATA VALUE WRITTEN	TEN	DESCRIPTION
0xXX (Don't care)	0	Software controlled ADC-Sample or ADC-Burst
0x55	1	Software generated trigger start
0xAA	1	Software generated trigger end (or stop)
0x5A	1	Software generated trigger sync
Any value not equal to 0x55 or 0xAA or 0x5A	1	Software controlled ADC-Sample or ADC-Burst

See Also

Register Summary (see page 43)

Example

11.3 ADC Data LSB (Offset=0)

ADC Data LSB. Please refer to ADC Data (see page 48) Register for further details.

See Also

Register Summary (see page 43)

11.4 ADC Data MSB (Offset=1)

ADC Data MSB. Please refer to ADC Data (see page 48) Register for further details.

See Also

Register Summary (see page 43)

11.5 ADC Data (Offset=0)

ADC Data Register.

Register Layout

ADC LSB. Offset=0x0, Byte 0. Offset=0, Word 0.

D7	D6	D5	D4	D3	D2	D1	D0
AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0

ADC MSB. Offset=0x1, Byte 1. Offset=0, Word 0.

D15	D14	D13	D12	D11	D10	D9	D8
AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
AD[15:0]	r	- na -	ADC data word. AD0 is the least significant bit. AD15 is the most significant data bit. In the Straight Binary Encoding (legacy default) a reading of 0_{10} (0x0000) represents a negative full scale input and a value of 65535_{10} (0xFFFF) represents an input of positive full scale. The two registers can both be read simultaneously by simply reading the data as a 16-bit I/O transaction (Examples: “in ax, dx” or “inpw(base_address+0)”).

Description

The ADC LSB register provides the least significant data byte and the ADC MSB register provides the most significant data byte. Each of the two registers can be read in any order (when FIFO Superset is not enabled). Writing any data to the ADC LSB register issues a software trigger.

Data bandwidth between the STX104 and CPU (PC/104 ISA bus) can be doubled by simply reading the ADC register as a 16-bit register. Software examples are shown below. Further improvement in bus bandwidth can be had by limiting the CPU-burst readouts (i.e. Insw() function) to the data fragment buffer size; thus eliminating I/O wait states.

When FIFO Superset is enabled (M1 jumper installed) the ADC data is read out of the FIFO as either a word or a byte at a time. To read out the data correctly as a byte at a time, you must first read the LSB and then the MSB. Once the MSB is read, the FIFO is considered to be read and FIFO is advanced to the next value to be read (if not empty). You may continue to read the data until the FIFO empty flag is set. It is possible to continuously sample analog inputs (continuous ADC sampling or triggering) and continuously read FIFO without any breaks in sampling.

It is the programmer's responsibility to be sure that data is read out in the correct sequence, as well as starting and stopping the acquisition as required.

For the following tables below, please refer to the Analog Input General Configuration (see page 100) Register and the Analog Input Polarity (see page 103) Register for details on ADC input bit pattern encoding schemes.

VOLTAGE INPUT RANGES --- STRAIGHT BINARY ENCODING (default legacy mode)

OUTPUT RANGE	UNIPOLAR JUMPER J9 *	ADC CFG REG G1	ADC CFG REG G0	RESOLUTION	NEG FULL SCALE VOLTAGE	POS FULL SCALE VOLTAGE	NEG FULL SCALE HEX	POS FULL SCALE HEX
+/- 10 Volts	0	0	0	305 uV	-10.00	+10.00	0x0000	0xFFFF
+/- 5 Volts	0	0	1	153 uV	-5.000	+5.000	0x0000	0xFFFF
+/- 2.5 Volts	0	1	0	76 uV	-2.500	+2.500	0x0000	0xFFFF
+/- 1.25 Volts	0	1	1	38 uV	-1.250	+1.250	0x0000	0xFFFF
0 - 10 Volts	1	0	0	153 uV	0.000	+10.00	0x0000	0xFFFF
0 - 5 Volts	1	0	1	76 uV	0.000	+5.000	0x0000	0xFFFF
0 - 2.5 Volts	1	1	0	38 uV	0.000	+2.500	0x0000	0xFFFF
0 - 1.25 Volts	1	1	1	19 uV	0.000	+1.250	0x0000	0xFFFF

* '1' = Jumper installed. '0' = Jumper not installed.

VOLTAGE INPUT RANGES --- BITWISE INVERTED STRAIGHT BINARY ENCODING

OUTPUT RANGE	UNIPOLAR JUMPER J9 *	ADC CFG REG G1	ADC CFG REG G0	RESOLUTION	NEG FULL SCALE VOLTAGE	POS FULL SCALE VOLTAGE	NEG FULL SCALE HEX	POS FULL SCALE HEX
+/- 10 Volts	0	0	0	305 uV	-10.00	+10.00	0xFFFF	0x0000
+/- 5 Volts	0	0	1	153 uV	-5.000	+5.000	0xFFFF	0x0000
+/- 2.5 Volts	0	1	0	76 uV	-2.500	+2.500	0xFFFF	0x0000
+/- 1.25 Volts	0	1	1	38 uV	-1.250	+1.250	0xFFFF	0x0000

0 - 10 Volts	1	0	0	153 uV	0.000	+10.00	0xFFFF	0x0000
0 - 5 Volts	1	0	1	76 uV	0.000	+5.000	0xFFFF	0x0000
0 - 2.5 Volts	1	1	0	38 uV	0.000	+2.500	0xFFFF	0x0000
0 - 1.25 Volts	1	1	1	19 uV	0.000	+1.250	0xFFFF	0x0000

* '1' = Jumper installed. '0' = Jumper not installed.

VOLTAGE INPUT RANGES --- TWOs COMPLEMENT ENCODING

OUTPUT RANGE	UNIPOLAR JUMPER J9 *	ADC CFG REG G1	ADC CFG REG G0	RESOLUTION	NEG FULL SCALE VOLTAGE	POS FULL SCALE VOLTAGE	NEG FULL SCALE HEX	POS FULL SCALE HEX
+/- 10 Volts	0	0	0	305 uV	-10.00	+10.00	0x8000	0x7FFF
+/- 5 Volts	0	0	1	153 uV	-5.000	+5.000	0x8000	0x7FFF
+/- 2.5 Volts	0	1	0	76 uV	-2.500	+2.500	0x8000	0x7FFF
+/- 1.25 Volts	0	1	1	38 uV	-1.250	+1.250	0x8000	0x7FFF
0 - 10 Volts	1	0	0	153 uV	0.000	+10.00	0x8000	0x7FFF
0 - 5 Volts	1	0	1	76 uV	0.000	+5.000	0x8000	0x7FFF
0 - 2.5 Volts	1	1	0	38 uV	0.000	+2.500	0x8000	0x7FFF
0 - 1.25 Volts	1	1	1	19 uV	0.000	+1.250	0x8000	0x7FFF

* '1' = Jumper installed. '0' = Jumper not installed.

VOLTAGE INPUT RANGES --- TWOs COMPLEMENT ENCODING MULTIPLIED BY -1 with CLAMPING ENABLED

OUTPUT RANGE	UNIPOLAR JUMPER J9 *	ADC CFG REG G1	ADC CFG REG G0	RESOLUTION	NEG FULL SCALE VOLTAGE	POS FULL SCALE VOLTAGE	NEG FULL SCALE HEX	POS FULL SCALE HEX
+/- 10 Volts	0	0	0	305 uV	-10.00	+10.00	0x7FFF	0x8001
+/- 5 Volts	0	0	1	153 uV	-5.000	+5.000	0x7FFF	0x8001
+/- 2.5 Volts	0	1	0	76 uV	-2.500	+2.500	0x7FFF	0x8001
+/- 1.25 Volts	0	1	1	38 uV	-1.250	+1.250	0x7FFF	0x8001
0 - 10 Volts	1	0	0	153 uV	0.000	+10.00	0x7FFF	0x8001
0 - 5 Volts	1	0	1	76 uV	0.000	+5.000	0x7FFF	0x8001
0 - 2.5 Volts	1	1	0	38 uV	0.000	+2.500	0x7FFF	0x8001
0 - 1.25 Volts	1	1	1	19 uV	0.000	+1.250	0x7FFF	0x8001

* '1' = Jumper installed. '0' = Jumper not installed.

USING THE INSW() FUNCTION or THE REP INSW INSTRUCTION

Using the REP INSW instruction (or Insw() function in Linux) provides a very high speed mechanism for reading out ADC data from the FIFO. This instruction is available on 286 CPUs and above. In fact, this is faster and easier to use than DMA; both in implementation and execution. Using REP INSW alleviates video problems related with DMA as well as improved CPU timing management (i.e. DMA adds timing holes in the system that are not easily managed by simple real-time kernels). The speed of any ISA bus I/O transaction is dependent on the CPU ISA bus speed which is usually set in the BIOS setup; use this setting with caution as it may affect performance of other cards and/or can cause data loss. This is usually available on many CPU cards.

Limiting the size of the sample count to the STX104 data fragment buffer in the Insw() function call will eliminate I/O bus wait states and further enhance overall throughput. In most cases this will improve throughput by another factor of two.

Note: DMA is not available on Revisions 160923 and 161023.

See Also

Register Summary (see page 43)

Example

Examples of how to read the ADC data. All of the following examples assume that the we are using Straight Binary Data Encoding which is the legacy default mode.

8-Bit Read in C/C++:

```
union { unsigned int word; unsigned char byte[2]; } ad_value[16];
...
adc_value(channel).byte[0] = inp(base_address+0);
adc_value(channel).byte[1] = inp(base_address+1);
/* adc_value(channel).word now contains the full ADC word value */
...
```

16-Bit Read in C/C++:

```
unsigned int ad_value[16];
...
adc_value(channel) = inpw(base_address+0);
...
```

8-Bit Read and FIFO enabled in C/C++:

```
union { unsigned int word; unsigned char byte[2]; } ad_value[16];
...
if ( fifo_not_empty() == true ) /* function to check fifo status */
{
 adc_value(channel).byte[0] = inp(base_address+0);
 adc_value(channel).byte[1] = inp(base_address+1);
 /* adc_value(channel).word now contains the full ADC word value */
}
...
```

16-Bit Read and FIFO enabled in C/C++:

```
unsigned ad_value[16];
...
if ( fifo_not_empty() == true ) /* function to check fifo status */
{
adc_value(channel) = inpw(base_address+0);
}
...
```

16-Bit Read with FIFO enabled in C/C++ using REP INSW:

```
typedef unsigned int WORD;

void insw(WORD port, void *buf, int count)
{
 _ES = FP_SEG(buf); /* Segment of buf */
 _DI = FP_OFF(buf); /* Offset of buf */
 _CX = count; /* Number to read */
 _DX = port; /* Port */
 asm REP INSW;
}

void main()
{
 WORD *data[512];
 ...
 insw(0x300, data, 512); /* assumes 512 samples (or two blocks) in FIFO */
 ...
}
```

An example shown below illustrating how to set the gain and use it to determine the actual input voltage.

```
/*****************************************************************************
   Apex Embedded Systems
   Revised: 03MAR08
   STX104 Analog Input Demo
*****************************************************************************/

#include <conio.h>
#include <stdio.h>
#include <dos.h>

/*****************************************************************************/
float adc_gain[8] = { 20.0/65536.0, 10.0/65536.0, 5.0/65536.0, 2.5/65536.0,
              10.0/65536.0,  5.0/65536.0, 2.5/65536.0, 1.25/65536.0 };
float adc_offset[8] = { -10.0, -5.0, -2.5, -1.25, 0.0, 0.0, 0.0, 0.0 };

/*****************************************************************************/
float Adc_Voltage ( void )
{
    long sum;
    int index;
    float voltage;

    /* take several samples and average */
    sum = 0;
    for ( index=0; index <256; index++ )
    {
       outportb(0x300, 0x00); /* start a sample */
```

```
        while ( (inportb(0x308) & 0x80) != 0x00 ); /* wait for sample */
        sum = sum + ((long)inpw(0x300));
    }
    sum = sum / 256;
    index = inportb( 0x30B ) & 0x07;
    voltage = ( (float) sum ) * adc_gain[index] + adc_offset[index];
    return voltage;
}

/*************************************************************************/
void main()
{
      int x,y;
      unsigned char channels;

      channels = 0x00; /* scan channel zero only */
      /* base address set to factory default at 0x300 */
      /* initialize STX104 */
      outportb(0x309, 0x00); /* no interrupts, no DMA and s/w trigger */

      /* Applies to firmware revision 080214H only: for non-zero first_channel
         please write to channel register twice to correct for errata issue.  This
         is transparent for other firmware revisions (simply means that acquisition
         controller is reset twice).
      */
      outportb(0x302, channels);
      outportb(0x302, channels);
      while ((inportb(0x308) & 0x80 ) == 0x80 ); /* wait       */

      printf("0) +10.0V   input\n");
      printf("1)  +5.0V   input\n");
      printf("2)  +2.5V   input\n");
      printf("3)  +1.25V  input\n");
      printf("9) Quit this test\n");

      do {
           y = inportb( 0x30B ) & 0x07;
           if ( (y & 0x04) == 0x00 ) printf("Bipolar:  ");
           else printf("Unipolar: ");
           y = y & 0x03;
           printf(" Gain=");
           if ( y == 0 )      printf("10V,   ");
           else if ( y == 1 ) printf(" 5V,   ");
           else if ( y == 2 ) printf(" 2.5V, ");
           else if ( y == 3 ) printf(" 1.25V,");
           printf(" Voltage = %+10.6f        \r", Adc_Voltage());

           if ( kbhit() )
           {
              y = getch();
              /* set gain */
              if ( y == '0' )      { x = 0;    outportb( 0x30B, x ); }
              else if ( y == '1' ) { x = 1;    outportb( 0x30B, x ); }
              else if ( y == '2' ) { x = 2;    outportb( 0x30B, x ); }
              else if ( y == '3' ) { x = 3;    outportb( 0x30B, x ); }
              else if ( y == '9' ) x = 9;
           }
       } while ( x != 9 );
       printf("\n\n");
}
```

11.6 ADC Channel (Offset=2)

ADC Channel Scan Register

Register Layout

Offset=0x2, Byte 0.

D7	D6	D5	D4	D3	D2	D1	D0
LC3	LC2	LC1	LC0	FC3	FC2	FC1	FC0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
CH[3:0]	r	0000	Current Channel to be sampled. See ADC Status Register (see page 64). This value is sent to the analog multiplexers to set the current analog channel to be sampled. This value can be read in the ADC Status Register. Shown here for completeness only.
FC[3:0]	rw	0000	First Channel. This is the first analog channel that is sampled. When this value is written, it will also set the current-channel (CH = FC).
LC[3:0]	rw	0000	Last Channel. This is the last analog channel that is sampled. Once this channel has been sampled, the current-channel value will wrap to the first-channel value.

Description

The channel register contains *first-channel* and the *last-channel* in the scanning range of the analog multiplexer. When the channel register is written the first-channel value is also written to the *current-channel* which sets the analog channel to be sampled. Upon an ADC trigger, the ADC samples the *current-channel*, and the multiplexer is advanced to the next channel. The current-channel value will wrap from the *last-channel* to the to the *first-channel* once the *last-channel* has been sampled.

The *current-channel* value is incremented only when the ADC is sampled or triggered. For every ADC trigger, the *current-channel* is incremented. The *current-channel* is presented as CH[3:0] in the ADC Status Register (see page 64).

If the STX104 is configured for differential input mode, the most significant bit of the *current-channel* (and therefore *first-* and *last-channel*) is ignored.

If the FIFO function is enabled, the user software must track the channel being read out of the FIFO. In other words, it is the software which must maintain synchronization.

If the *first-channel* is the same as the *last-channel*, then the analog channel (multiplexer) is not changed. This can be useful for sampling the same channel continuously.

It is recommended to have *first-channel* < *last-channel* to prevent confusion of the channel sequencing as illustrated in examples (C) and (D) below.

FIFO RESET

Writing to the channel register will reset the FIFO when the FIFO Superset mode is enabled (jumper M1 is installed) and/or DAS1602 mode enabled (jumper M0 is installed).

ACQUISITION CONTROLLER RESET

Writing to the Channel Register resets the internal acquisition controller in all modes.

MOVING AVERAGE FILTER RESET

Writing to the Channel Register also resets the internal moving average filter in all modes when jumper M3 is installed. The CNV bit (see ADC Status Register (see page 64)) will become active for approximately six microseconds while the moving average filter is reset.

SSH OUTPUT

The Start Sample and Hold (SSH) signal, at pin 14 of the I/O connector, is a TTL output used to drive the sample and holds line of external simultaneous sample and hold cards. The behavior of the SSH output signal is related to the Channel Register. Writing any value to the Channel Register will bring the SSH line high. The SSH line will go low, indicating a hold, when the *first_channel* sampling has completed (i.e. the input multiplexers now looking at the next channel). The SSH line will return high when *last_channel* has been sampled and the multiplexers wrap back to the *first_channel*.

See Also

Register Summary (see page 43)

Example

Example channel sequencing (all in hexadecimal values):

a) LC[3:0] = D, FC[3:0] = 3

16-Channel, Single Ended: 3,4,5,6,7,8,9,A,B,C,D,3,4,5,...

8-Channel, Differential (see page 19): 3,4,5,3,4,5,...

b) LC[3:0] = 1, FC[3:0] = 9

16-Channel, Single Ended: 9,A,B,C,D,E,F,0,1,9,A,B,C,...

8-Channel, Differential (see page 19): 1,1,1,1,...

c) LC[3:0] = 6, FC[3:0] = 5

16-Channel, Single Ended: 5,6,5,6,...

8-Channel, Differential (see page 19): 5,6,5,6,...

d) LC[3:0] = 5, FC[3:0] = 6

16-Channel, Single Ended: 5,6,7,8,9,A,B,C,D,E,F,0,1,2,3,4,5,6,7,8,9,...

8-Channel, Differential (see page 19): 6,7,0,1,2,3,4,5,6,7,0,1,2,...

11.7 Digital Outputs (Offset=3)

Digital Output Register.

Register Layout

Offset=0x3, Byte 0.

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	DOUT3	DOUT2	DOUT1	DOUT0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
X	-	-	Don't Care
DOUT[3:0]	w	0000	Digital Outputs. These bits are write only. Non-inverted digital output bits. You must maintain a copy in software for bit manipulation.

BIT NAME	DIRECTION	CONNECTOR PIN POSITION	PHYSICAL I/O TYPE
DOUT3	--->	J7.5	LVTTL Output
DOUT2	--->	J7.6	LVTTL Output
DOUT1	--->	J7.7	LVTTL Output
DOUT0	--->	J7.8	LVTTL Output

Description

Digital TTL Outputs.

See Also

Register Summary (see page 43)

Digital Output Configuration (see page 94)

Example

11.8 Digital Inputs (Offset=3)

Digital Input Register.

Register Layout

Offset=0x3, Byte 0.

D7	D6	D5	D4	D3	D2	D1	D0
CM1	CM0	X	X	DIN3	DIN2 GATE0	DIN1	DIN0 TRIG

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
CM[1:0]	r	M0	Compatibility mode. These bits are set to zero for DAS16jr/16 mode. These bits are set to one for DAS1602 mode compatibility.
X	-	-	Don't Care
DIN[3:0]	r	0000	Digital Inputs. DIN0 also functions as an ADC external trigger input. As a trigger input, it is deglitched by an internal match-filter and requires a minimum pulse width of no less than 200nS. See the Interrupt Summary (see page 43) section for further functionality. DIN2 also functions as counter-zero gate input.

DIN1 and DIN3 swapped to their correct positions

BIT NAME	DIRECTION	CONNECTOR PIN POSITION	PHYSICAL I/O TYPE
DIN3	<---	J7.11	TTL Input
DIN2	<---	J7.10	TTL Input
DIN1	<---	J7.9	TTL Input
DIN0	<---	J7.12	TTL Input

Classic STX104 Configuration (supporting existing customers, Default mode)

BIT NAME	DIRECTION	CONNECTOR PIN POSITION	PHYSICAL I/O TYPE
DIN3	<---	J7.9	TTL Input
DIN2	<---	J7.10	TTL Input

DIN1	<---	J7.11	TTL Input
DIN0	<---	J7.12	TTL Input

Description

Digital TTL inputs.

See Also

Register Summary (see page 43)

Digital Input Configuration (see page 96)

Example

11.9 DAC Channel-A LSB (Offset=4)

DAC Channel-A LSB. Please Refer to DAC Channel-A (see page 58) Register Details.

See Also

Register Summary (see page 43)

11.10 DAC Channel-A MSB (Offset=5)

DAC Channel-A MSB. Please Refer to DAC Channel-A (see page 58) Register Details.

See Also

Register Summary (see page 43)

11.11 DAC Channel-A (Offset=4)

DAC Channel-A Register

Register Layout

DAC Channel-A LSB. Offset=0x4, Byte 0. Offset=0x4, Word 0.

D7	D6	D5	D4	D3	D2	D1	D0
DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0

DAC Channel-A MSB. Offset=0x5, Byte 1. Offset=0x4, Word 0.

D15	D14	D13	D12	D11	D10	D9	D8
DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
DA[15:0]	w	- na -	DAC Channel-A data word in 16-Bit mode. DA0 is the least significant bit and DA15 is the most significant data bit.

BIT STRING NAME	DIRECTION	CONNECTOR PIN POSITION	PHYSICAL I/O TYPE
DA[15:0]	--->	J7.17 (DAC_OUT_1)	Analog Output

Description

Each channel is updated once the MSB is written. Writing only the MSB will update the DAC channel output. The results of changing jumper settings at J5 will only take affect after writing the MSB on the DAC output. The two 8-bit DAC registers can be written simultaneously by writing the data as a 16-bit I/O transaction (Examples: “out dx, ax” or “outpw(base_address+4, dac_value)”).

DAC outputs are available in either DAS16jr/16 or DAS1602 modes. The DAC outputs are always enabled and available for use. The DAC output bit alignments can be adjusted for either 12-bit legacy operation or full 16-bit DAC mode.

POWER UP or RESET

At power-up or reset the DAC outputs are cleared to zero volts.

OUTPUT VOLTAGE CONVERSION

OUTPUT RANGE	DA1_UB J5 *	DA1_R J5 *	RESOLUTION	NEG FULL SCALE VOLTAGE	POS FULL SCALE VOLTAGE	NEG FULL SCALE HEX	POS FULL SCALE HEX
+/- 10 Volts	1	1	305 uV	-10.00	+10.00	0x0000	0xFFFF
+/- 5 Volts	1	0	153 uV	-5.000	+5.000	0x0000	0xFFFF
0 - 10 Volts	0	1	153 uV	0.000	+10.00	0x0000	0xFFFF
0 - 5 Volts	0	0	76 uV	0.000	+5.000	0x0000	0xFFFF

* '1' = Jumper installed. '0' = Jumper not installed.

See Also

Register Summary (see page 43)

Example

Examples of how to write to the DAC output register in 16-bit DAC mode.

8-Bit Writes in C/C++:

```
unsigned int dac_value;
...
outp( base_address+4, dac_value & 0xFF );
outp( base_address+5, dac_value >> 8 );
...

or

union { unsigned int word; unsigned char byte[2]; } dac_value;
...
outp( base_address+4, dac_value.byte[0] );
outp( base_address+5, dac_value.byte[1] );
...
```


16-Bit Write in C/C++:

```
unsigned int dac_value;
...
outpw( base_address+4, dac_value );
...
```

11.12 DAC Channel-B LSB (Offset=6)

DAC Channel-B LSB. Please Refer to DAC Channel-B (see page 61) Register Details.

See Also

Register Summary (see page 43)

11.13 DAC Channel-B MSB (Offset=7)

DAC Channel-B MSB. Please Refer to DAC Channel-B (see page 61) Register Details.

See Also

Register Summary (see page 43)

11.14 DAC Channel-B (Offset=6)

DAC Channel-B Register

Register Layout

DAC Channel-B LSB. Offset=0x6, Byte 0. Offset=0x6, Word 0.

D7	D6	D5	D4	D3	D2	D1	D0
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

DAC Channel-B MSB. Offset=0x7, Byte 0. Offset=0x6, Word 0.

D15	D14	D13	D12	D11	D10	D9	D8
DB15	DB14	DB13	DB12	DB11	DB10	DB9	DB8

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
DB[15:0]	w	- na -	DAC Channel-A data word in 16-Bit mode. DA0 is the least significant bit and DB15 is the most significant data bit.

BIT STRING NAME	DIRECTION	CONNECTOR PIN POSITION	PHYSICAL I/O TYPE
DB[15:0]	--->	J7.16 (DAC_OUT_2)	Analog Output

Description

Each channel is updated once the MSB is written. Writing only the MSB will update the DAC channel output. The results of changing jumper settings at J5 will only take affect after writing the MSB on the DAC output. The two 8-bit DAC registers can be written simultaneously by writing the data as a 16-bit I/O transaction (Examples: “out dx, ax” or “outpw(base_address+4, dac_value)”).

DAC outputs are available in either DAS16jr/16 or DAS1602 modes. The DAC outputs are always enabled and available for use. The DAC output bit alignments can be adjusted for either 12-bit legacy operation or full 16-bit DAC mode.

POWER UP or RESET

At power-up or reset the DAC outputs are at set to zero volts.

OUTPUT VOLTAGE CONVERSION

OUTPUT RANGE	DB1_UB J5 *	DB1_R J5 *	RESOLUTION	NEG FULL SCALE VOLTAGE	POS FULL SCALE VOLTAGE	NEG FULL SCALE HEX	POS FULL SCALE HEX
+/- 10 Volts	1	1	305 uV	-10.00	+10.00	0x0000	0xFFFF
+/- 5 Volts	1	0	153 uV	-5.000	+5.000	0x0000	0xFFFF
0 - 10 Volts	0	1	153 uV	0.000	+10.00	0x0000	0xFFFF
0 - 5 Volts	0	0	76 uV	0.000	+5.000	0x0000	0xFFFF

* '1' = Jumper installed. '0' = Jumper not installed.

See Also

Register Summary (see page 43)

Example

Examples of how to write to the DAC output register in 16-bit DAC mode.

8-Bit Writes in C/C++:

```
unsigned int dac_value;
...
outp( base_address+6, dac_value & 0xFF );
outp( base_address+7, dac_value >> 8 );
...

or

union { unsigned int word; unsigned char byte[2]; } dac_value;
...
outp( base_address+6, dac_value.byte[0] );
outp( base_address+7, dac_value.byte[1] );
...
```

16-Bit Write in C/C++:

```
unsigned int dac_value;
...
outpw( base_address+6, dac_value );
...
```

11.15 Clear Interrupts (Offset=8)

Clear Interrupt Register

Register Layout

Offset=0x8, Byte 0.

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	X	X	X	X

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
X	-	-	Don't Care

Description

Writing to this register will clear any pending interrupts.

See Also

Register Summary (see page 43)

Example

11.16 ADC Status (Offset=8)

ADC Status Register.

Register Layout

Offset=0x8, Byte 0. ISSBE='1' (refer to the Interrupt Configuration Register (see page 90)).

D7	D6	D5	D4	D3	D2	D1	D0
CNV	UB	SD	INT	TAS	FF_INT	ISSB	ISSA

Offset=0x8, Byte 0. ISSBE='0' (default).

D7	D6	D5	D4	D3	D2	D1	D0
CNV	UB	SD	INT	CH3	CH2	CH1	CH0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
CNV	r	0	ADC Conversion (and/or ADC-Burst in DAS1602 mode) in progress. Writing to the Channel Register may cause the CNV bit to become active indicating that a STX104 internal reset is in progress (typically less than 1uS, and less than 10uS when moving average filter is enabled). 1 = ADC conversion, scan or acquisition reset in progress. 0 = ADC Idle (default)
UB	r	-	Unipolar / Bipolar ADC input mode setting (J9): 0 = bipolar. Measure both negative and positive input voltages (J9 not stuffed) 1 = unipolar. Measure only positive input values
SD	r	-	Single-ended / Differential (see page 19) ADC input mode setting (J8): 0 = Single-ended 1 = Differential (see page 19) (J8 not stuffed)

INT	r	0	Interrupt request status bit. 0 = No interrupt pending (default) 1 = Interrupt is pending; ADC trigger or ADC-Burst conversion has completed Note: ADC conversions continue to occur on schedule (via selected trigger source) regardless of whether this bit is cleared. If a new conversion occurs before this bit is cleared, an over-run condition may have occurred. Therefore, the programmer must ensure that the interrupt rate is not faster than the capacity of the CPU and software to respond. If FIFO Superset is enabled (jumper M1 installed), then the only over-run that will occur is if the FIFO is full (FIFO Full flag is true or FF='1'). Thus, interrupt latency requirements are greatly relaxed. If ADC interrupts are not enabled, this bit can still be used to determine when an ADC conversion has occurred when polling this bit.
CH[3:0]	r	0000	Current ADC channel. This is the channel currently selected on the board and is the channel that will be used for the next ADC conversion provided CNV=0 (unless a new value is written to the channel register). The CH[3:0] will change shortly after an ADC trigger.
TAS	r	0	Trigger Activity State. This is the same status bit as the TRG bit found in the ADC Configuration (see page 72) Register. The trigger status is provided here so that interrupt and board status can be read as one value, thus keeping an interrupt service routine short as possible. 0 = trigger inactive (default) 1 = trigger active
FF_INT	r	0	FIFO Interrupt Status (Please refer to FIFO Status MSB (see page 69) Register for further details): 0 = Not Active 1 = Interrupt Active
ISSB	r	0	Interrupt Status Source B: 0 = Not Active 1 = Interrupt Active
ISSA	r	0	Interrupt Status Source A: 0 = Not Active 1 = Interrupt Active
X	-	-	Don't Care

Description

See Also

Register Summary (see page 43)

Example

11.17 ADC Control (Offset=9)

ADC Control Register

Register Layout

Offset=0x9, Byte 0. EIS='1', Please refer to the Interrupt Configuration Register (see page 90).

D7	D6	D5	D4	D3	D2	D1	D0
IES3	IES2	IES1	IES0	FIE (a)	DMAE	ALSS1 (b)	ALSS0 (b)

Offset=0x9, Byte 0. EIS='0' (default), Please refer to the Interrupt Configuration Register (see page 90).

D7	D6	D5	D4	D3	D2	D1	D0
AIE	INTS2	INTS1	INTS0	FIE (a)	DMAE	ALSS1 (b)	ALSS0 (b)

(a) FIFO Superset Enabled (jumper M1 installed), otherwise the bit is a don't care.

(b) Was previously named TSx. The term "ADC-sample" or "ADC-burst" is used to replace the term "trigger", since triggering takes on a more generalized definition.

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
IES[3:0]	rw	0000	Interrupt enhanced select (available only in Enhanced Register Set Mode): 0000 = Not valid, interrupts disabled (default) 0001 = Not valid, interrupts disabled 0010 = IRQ9/2 (IRQ2 for 8-bit or XT, IRQ9 for 16-bit or AT) 0011 = IRQ3 0100 = IRQ4 0101 = IRQ5 0110 = IRQ6 0111 = IRQ7 1000 = Not valid, interrupts disabled 1001 = IRQ9/2 (IRQ2 for 8-bit or XT, IRQ9 for 16-bit or AT) 1010 = IRQ10 1011 = IRQ11 1100 = IRQ12 1101 = Not valid, interrupts disabled 1110 = IRQ14 1111 = IRQ15

AIE	rw	0	ADC Interrupt Enable. 0 = Disable interrupt (default) 1 = Enable interrupt
INTS[2:0]	rw	000	Interrupt select: 000 = Not valid, interrupts disabled (default) 001 = Not valid, interrupts disabled 010 = IRQ2 011 = IRQ3 100 = IRQ4 101 = IRQ5 110 = IRQ6 111 = IRQ7
FIE	rw	0	FIFO Interrupt Enable (only when jumper M1 is installed): 0 = Disable FIFO interrupt (default) 1 = Enable FIFO interrupt
DMAE	rw	0	Direct Memory Access (DMA) enable: 0 = Disable DMA transactions (default) 1 = Enable DMA transactions The DMA request lines are tri-stated until the DMA bit is enabled, thus allowing multiple DMA devices to share the selected DMA channel provided that they are not enabled at the same time.
ALSS[1:0]	rw	00	ADC-Sample or ADC-Burst Legacy Sampling Select: 0X = Software (default) 10 = DIN0 rising/falling edge 11 = 8254 Counter 1 & 2 Pacer CT_OUT2 rising edge
X	-	-	Don’t Care

Description

The ADC Control Register is used to configure interrupts, select DMA mode and select ADC trigger source.

See Interrupt Summary (see page 119) for more information on interrupts.

DMA NOTE

We strongly encourage you to use the REP INSW instruction (or Insw() function in Linux) as it performs better than DMA and is substantially simpler to set up and use. Note that DMA performs I/O to memory transfers a byte at a time (DMA 1 and DMA 3 are byte wide transfers) while REP INSW performs I/O to memory transfers a word at a time. If you are designing a real-time system, you will have better timing control over your system by using REP INSW instruction over DMA. Since processor generated I/O cycles are faster than DMA generated cycles (typically 350ns versus 800ns), data transfer can take place faster than DMA. Note that 8-bit bus transactions (including DMA) are typically at least twice as long as 16-bit bus transactions; this alone is reason enough to avoid DMA.

When DMA is enabled, the 1 mega-sample FIFO is used internally.

In DMA mode, receiving a terminal count from the CPU will generate an interrupt, if the interrupt is enabled. If you are in DAS1602 mode (jumper M0 is installed), receiving a terminal count will set the Conversion Disable bit to false, thus disabling any

additional ADC triggers (or sampling). Writing 0x00 to the Conversion Disable Register will allow ADC sampling to continue.

In order to use DMA, you must set up the computer's DMA controller and page registers before enabling DMA on the STX104 board.

The speed of any ISA bus I/O transaction is dependent on the CPU ISA bus speed which is usually set in the BIOS setup; use this setting with caution as it may affect performance of other cards and/or can cause data loss. This is usually available on many CPU cards.

Note: DMA is not available on Revision 161023 or later.

See Also

Register Summary (see page 43)

Example

11.18 Pacer Clock Control (Offset=10)

Pacer Clock Control

Register Layout

Offset=0xA, Byte 0. DAS16jr/16 Compatibility Mode (M0 jumper *not* installed).

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	X	X	CT_SRC0	GCTRL

Offset=0xA, Byte 0. DAS1602 Compatibility Mode (M0 jumper installed).

D7	D6	D5	D4	D3	D2	D1	D0
ABL3	ABL2	ABL1	ABL0	X	X	CT_SRC0	GCTRL

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
ABL[3:0]	w	-	ADC-Burst Length. Determines the number of conversions per trigger when in burst mode. One to sixteen samples (single-ended) or up to eight channels (differential) in a burst. When not in Burst mode, then these bits have no function. The burst length is only used in the DAS1602 compatibility mode. If ABL[3:0]=0x0, then 1-channel burst is performed. If ABL[3:0]=0xF, then 16-channel ADC-burst is performed.

CT_SRC0	w	1	Counter 0 Clock Source: 1 = Counter 0 Clock Source is a 100KHz on-board reference frequency. CT_SRC0 (J7.4) gates this signal. When this bit is high (default), the 100KHz signal runs, otherwise the 100KHz clock is stopped. 0 = Counter 0 Clock Source to Counter 0 is an inverted polarity copy of CT_CLK0 input. CT_CLK0 is connected to a 10K ohm pull-up resistor.
GCTRL	w	0	Counters 1 and 2 gate control: 0 = Counters 1 and 2 run freely with no gating. 1 = Counters 1 and 2 are gated by DIN0 (J7.12). DIN0 is connected to a 10K ohm pull-up resistor.
X	-	-	Don't Care

Description

See Also

Register Summary (see page 43)

Example

11.19 FIFO Status MSB (Offset=10)

FIFO Status MSB Register.

Register Layout

Offset=0xA, Byte 0. FIFO Status MSB Register.

D7	D6	D5	D4	D3	D2	D1	D0
FIFO_INT	DFR (a)	FF	FE	FBR11	FBR10	FBR9	FBR8

Offset=0xF, Byte 0. FIFO Status LSB Register.

D7	D6	D5	D4	D3	D2	D1	D0
FBR7	FBR6	FBR5	FBR4	FBR3	FBR2	FBR1	FBR0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
FIFO_INT	r	0	FIFO Data Interrupt. An interrupt will occur for 512 or more samples queued in the FIFO in legacy mode. If FIBLK[1:0] not zero (refer to Interrupt Source Select (see page 89)), then an interrupt will be generated with larger block counts. This bit can be polled and/or use to generate an interrupt to the CPU (setting FIE bit true in the ADC Control Register). 0 = No FIFO interrupt (default at reset or power-up) 1 = FIFO interrupt active. To Clear Interrupt Register, write any value at Clear Interrupt Register.
DFR	r	0	Data Fragment Ready Flag: 0 = Data Fragment not ready (default at reset or power-up) 1 = Data Fragment ready.
FF	r	0	FIFO Full Flag. 0 = FIFO not full (default at reset or power-up) 1 = FIFO full/overflow has occurred.
FE	r	1	FIFO Empty Flag. 1 = Empty (default at reset or power-up) 0 = FIFO not empty, data is present
FBR[11:0]	r	0x000	FIFO Data Blocks Remaining. This provides a mechanism for measuring amount of data remaining within the FIFO. Each block is 256 samples. For example, if FBR=0x05 then at least 1,280 samples remain to be read out of the STX104 FIFO. If FBR=0x00 and FE='0' then there are less than 256 samples remaining to be read out of the FIFO. When FBR>0, use REP INSW to read in 256 samples at a time (1 block). When FBR=0, use a software loop that monitors the FE bit while reading in the samples until the FIFO is empty.

Description

The FIFO Status Registers are only available when FIFO Superset is Enabled (M1 jumper installed).

The number of data blocks used in order to generate a FIFO interrupt (FIFO_INT), can be adjusted via the FIBLK[1:0] bits in the Interrupt Source Select (see page 89) or the Interrupt Threshold (see page 92) registers. In many cases utilizing the FIBLK[1:0] bits is sufficient and keeps the Interrupt Threshold Register available for other functions.

IOCHRDY and FIFO DATA FRAGMENT

In order to avoid using IOCHRDY or I/O bus wait states the STX104 incorporates an intermediate data buffer to support CPU-burst reads which avoid changing the state of the IOCHRDY line.

When DFR is true, then the data fragment is ready to be read out, typically using the Insw() function which incorporates the REP INSW instruction.

There is theoretically no limit, other than the maximum size of the entire STX104 FIFO memory, on the number of consecutive CPU reads that can occur. However, by limiting the number of samples readout by the CPU using the Insw() to approximately the size of the Data Fragment buffer, one can avoid I/O bus wait states and further improve bus bandwidth.

It should be further stated that the CPU can still readout data at any rate along with any ADC sampling mode. In fact, the data fragment buffer will be used in nearly all cases, thus bus wait states become a thing of the past. The only case where I/O bus wait states will exist are cases where the number of samples read out by the CPU in a CPU-bursting readout (i.e. Insw() function) exceeds the data fragment buffer size.

FIFO CLEAR or RESET

Note that writing to the Channel Register will reset the FIFO.

See Also

Register Summary (see page 43)

Example

Below is an example of properly reading the FIFO status. This applies to firmware revisions specifically before 080214H (although it will still work fine for any revision 080214H or higher).

```
static int stx104_fifo_status_blocks[STX104_BOARDS_COUNT_MAX];
static unsigned char stx104_fifo_status_full[STX104_BOARDS_COUNT_MAX];
static unsigned char stx104_fifo_status_empty[STX104_BOARDS_COUNT_MAX];


/*************************************************************
/                                                  FIFO STATUS
/ Revision History:
/ 15JAN07 - read FIFO status twice to remove the possibility
/           of reading an incorrect value due to FIFO status
/           changing during a 25nSec interval.
*/
#define STX104_FIFO_STATUS_READ_COUNT_MAX   1
void STX104_FIFO_Status( int board )
{
    union { unsigned int value; unsigned char octet[2]; }
ff_stat[STX104_FIFO_STATUS_READ_COUNT_MAX+1];
    unsigned int fbr_blocks;
    unsigned int fbr_blocks_minimum;
    unsigned char i;

    fbr_blocks_minimum = 0x0FFF;

    for ( i=0; i<=STX104_FIFO_STATUS_READ_COUNT_MAX; i++ )
    {
        ff_stat[i].octet[0] = inp( stx104_base_address[board] + STX104_FIFO_DATA_STATUS );
        ff_stat[i].octet[1] = inp( stx104_base_address[board] + STX104_FIFO_FLAGS );

        fbr_blocks = ff_stat[i].value & 0x0FFF;
        if ( fbr_blocks < fbr_blocks_minimum )
        {
            fbr_blocks_minimum = fbr_blocks;
        }
    }
    stx104_fifo_status_blocks[board] = (int) fbr_blocks_minimum;

    if ( (ff_stat[STX104_FIFO_STATUS_READ_COUNT_MAX].value & 0x1000) != 0x0000)
stx104_fifo_status_empty[board] = true;
    else stx104_fifo_status_empty[board] = false;
    if ( (ff_stat[STX104_FIFO_STATUS_READ_COUNT_MAX].value & 0x2000) != 0x0000)
```

```
stx104_fifo_status_full[board] = true;
    else stx104_fifo_status_full[board] = false;
}
```

For firmware revisions 080214H or higher:

```
static int stx104_fifo_status_blocks[STX104_BOARDS_COUNT_MAX];
static unsigned char stx104_fifo_status_full[STX104_BOARDS_COUNT_MAX];
static unsigned char stx104_fifo_status_empty[STX104_BOARDS_COUNT_MAX];


/*************************************************************
/                                                  FIFO STATUS
*/
void STX104_FIFO_Status( int board )
{
    union { unsigned int value; unsigned char octet[2]; } ff_stat;

    ff_stat.octet[0] = inp( stx104_base_address[board] + STX104_FIFO_DATA_STATUS );
    ff_stat.octet[1] = inp( stx104_base_address[board] + STX104_FIFO_FLAGS );

    stx104_fifo_status_blocks[board] = ff_stat.value & 0x0FFF;

    if ( (ff_stat.value & 0x1000) != 0x0000) stx104_fifo_status_empty[board] = true;
    else stx104_fifo_status_empty[board] = false;
    if ( (ff_stat.value & 0x2000) != 0x0000) stx104_fifo_status_full[board] = true;
    else stx104_fifo_status_full[board] = false;
}
```

11.20 ADC Configuration (Offset=11)

ADC Configuration Register

Register Layout

Offset=0xB, Byte 0. Write Only Pattern.

D7	D6	D5	D4	D3	D2	D1	D0
RBK3	RBK2	RBK1	RBK0	X	X	GAIN1	GAIN0

Offset=0xB, Byte 0. Read Only Pattern.

D7	D6	D5	D4	D3	D2	D1	D0
RB	X	X	X	TRG	ADBU	GAIN1	GAIN0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
RB	r	0	Register Bank Status (only available in Enhanced Register Mode): 0 = 8254 Counter/Timer Bank Active (default) 1 = Indexed Register Array Bank Active
RBK[3:0]	w	8254 Bank	Register Bank Key Select (only available in Enhanced Register Mode). Writing a special sequence will bank select either the 8254 counter/timer or an Indexed Register Array Set. The default state is always with the 8254 bank selected (i.e. original register set configuration). 1111 --> 0101 --> 1101 F 5 D 0011 --> 1011 --> 1010 3 B A Writing the sequence 0xFX, 0x5X, 0xDX will enable the Indexed Register Array Bank. Writing the sequence 0x3X, 0xBX, 0xAX will disable the Indexed Register Array Bank. The sequences preserve upward compatibility and ensure that the bank select remains set even under existing software activity.
TRG	r	0	Trigger Activity State. This same status bit is available at the ADC Status (see page 64) Register TAS bit. This bit is available here as well as in the event that classic interrupts are required which would make the TAS bit unavailable. 0 = trigger inactive (default) 1 = trigger active
ADBU	r	depends on J9	ADC bipolar/unipolar: 0 = bipolar setting (jumper J9 not installed) 1 = unipolar setting
GAIN[1:0]	rw	00	ADC gain setting: 00 = gain of x1 (default) 01 = gain of x2 10 = gain of x4 11 = gain of x8
X	-	-	Don’t care

Description

REGISTER BANK SELECT

Register Bank Select is a mechanism for providing additional configuration options for the STX104, while preserving the existing register set foot-print within the I/O space. Writing the sequence 0xFX, 0x5X, 0xDX will enable the Indexed Register Array Bank. Writing the sequence 0x3X, 0xBX, 0xAX will disable the Indexed Register Array Bank. Writing any other sequence will not change the state of the register bank. The sequences preserve upward compatibility and ensure that the bank select remains set even under existing software activity. Bank select can be verified by reading the RB bit shown above.

GAIN SUMMARY

Writing to this register sets the analog input gain for all 8/16 analog inputs. The ADBU bit is set by hardware. The current input gain is determined by reading this register.

INPUT RANGE	RESOLUTION	ADBU	G1	G0
+/- 10 V	305 uV	0	0	0
+/- 5 V	153 uV	0	0	1
+/- 2.5 V	76 uV	0	1	0
+/- 1.25 V	38 uV	0	1	1
0 - 10 V	153 uV	1	0	0
0 - 5 V	76 uV	1	0	1
0 - 2.5 V	38 uV	1	1	0
0 - 1.25 V	19 uV	1	1	1

The ADC data range is -32768_{10} to 32767_{10} for each of the input ranges listed below.

See the software example in the ADC Data Register section earlier in this chapter.

The gain setting is the ratio between the ADC full-scale range and the effective input signal range. For example, if the ADC full-scale range is 0-10V, a gain setting of 2 creates an input signal range of 0-5V, and gain setting of 4 creates an input range of 0-2.5V.

See Also

Register Summary (see page 43)

BANKING BETWEEN 8254 AND THE INDEXED REGISTER ARRAY

```
/*************************************************************
/                                    SET BANK 8254/INDEXED-ARRAY
/
/ Call this function when changing from one bank to another.  Recommend only
/ changing the bank when one read/writes to the 8254, since that is likely to
/ be read/written to the least amount compared to the indexed register array.
/
/ Alternatively, one could check the RB bit each time an indexed register
/ is read or written; it will then take longer on average to read/write registers.
*/
void STX104_Set_Bank( int board, char bank )
{
    unsigned char scratch;
    unsigned char value;
    unsigned int address;

    address = stx104_base_address[board] + STX104_ADC_CONFIGURATION;
    scratch = inp( address );
    if ( bank == 0 ) /* request banking to the 8254 */
    { /* test for RB bit in ADC configuration register */
        if ( (scratch & 0x80) == 0x80 )
        {
            scratch = scratch & 0x0F;
            value = scratch | 0x30;
            outp( address, value );
            value = scratch | 0x30;
            outp( address, value );
            value = scratch | 0xB0;
            outp( address, value );
            value = scratch | 0xA0;
            outp( address, value );
        }
```

```
    }
    else
    {   /* request banking to the indexed register array */
        if ( (scratch & 0x80) == 0 )
        {
            scratch = scratch & 0x0F;
            value = scratch | 0xF0;
            outp( address, value );
            value = scratch | 0xF0;
            outp( address, value );
            value = scratch | 0x50;
            outp( address, value );
            value = scratch | 0xD0;
            outp( address, value );
        }
    }
}
```

11.21 ADC Configuration PC104-DAS16Jr/12

ADC Configuration Register for PC104-DAS16Jr/12 mode

Register Layout

Offset=0xB, Byte 0. Write Only Pattern.

D7	D6	D5	D4	D3	D2	D1	D0
RBK3	RBK2	RBK1	RBK0	Range	ADBU	GAIN1	GAIN0

Offset=0xB, Byte 0. Read Only Pattern.

D7	D6	D5	D4	D3	D2	D1	D0
RB	X	X	X	Range	ADBU	GAIN1	GAIN0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
RB	r	0	Register Bank Status (only available in Enhanced Register Mode): 0 = 8254 Counter/Timer Bank Active (default) 1 = Indexed Register Array Bank Active

RBK[3:0]	w	8254 Bank	Register Bank Key Select (only available in Enhanced Register Mode). Writing a special sequence will bank select either the 8254 counter/timer or an Indexed Register Array Set. The default state is always with the 8254 bank selected (i.e. original register set configuration). 1111 --> 0101 --> 1101 F 5 D 0011 --> 1011 --> 1010 3 B A Writing the sequence 0xFX, 0x5X, 0xDX will enable the Indexed Register Array Bank. Writing the sequence 0x3X, 0xBX, 0xAX will disable the Indexed Register Array Bank. The sequences preserve upward compatibility and ensure that the bank select remains set even under existing software activity.
Range	r	0	See gain summary table
ADBU	rw	0	See gain summary table
GAIN[1:0]	rw	00	See gain summary table
X	-	-	Don’t care

Description

GAIN SUMMARY

Writing to this register sets the analog input gain for all 8/16 analog inputs. J9 must not be stuffed. The current input gain is determined by reading this register.

INPUT RANGE	RESOLUTION	Range	ADBU	G1	G0
+/- 10 V	4.88 mV	1	0	0	0
+/- 5 V	2.44 mV	0	0	0	0
+/- 2.5 V	1.22 mV	0	0	0	1
+/- 1.25 V	610 uV	0	0	1	0
+/- 0.625 V	305 uV	0	0	1	1
0 - 10 V	2.44 mV	0	1	0	0
0 - 5 V	1.22 mV	0	1	0	1
0 - 2.5 V	610 uV	0	1	1	0
0 - 1.25 V	305 uV	0	1	1	1

The ADC data range is 0_{16} to 4095_{16} for each of the input ranges listed below.

See Also

Register Summary (see page 43)

11.22 8254 CT0 Data (Offset=12, RB='0'. Index=68, RB='1')

8254 CTO Data Register. Please refer to the 8254 Configuration (see page 77) Register for further details.

See Also

Register Summary (see page 43)

11.23 8254 CT1 Data (Offset=13, RB='0'. Index=69, RB='1')

8254 CT1 Data Register. Please refer to the 8254 Configuration (see page 77) Register for further details.

See Also

Register Summary (see page 43)

11.24 8254 CT2 Data (Offset=14, RB='0'. Index=70, RB='1')

8254 CT2 Data Register. Please refer to the 8254 Configuration (see page 77) Register for further details.

See Also

Register Summary (see page 43)

11.25 8254 Configuration (Offset=15, RB='0'. Index=71, RB='1')

8254 Configuration Register

Register Layout

Offset=0xC, RB='0'. 8254 Counter/Timer Zero Data Register.This register is available when Register Bank Status is '0' (see ADC Configuration (see page 72) Register bit RB).

D7	D6	D5	D4	D3	D2	D1	D0
CT0D7	CT0D6	CT0D5	CT0D4	CT0D3	CT0D2	CT0D1	CT0D0

Offset=0xD, RB='0'. 8254 Counter/Timer One Data Register.

D7	D6	D5	D4	D3	D2	D1	D0
CT1D7	CT1D6	CT1D5	CT1D4	CT1D3	CT1D2	CT1D1	CT1D0

Offset=0xE, RB='0'. 8254 Counter/Timer Two Data Register.

D7	D6	D5	D4	D3	D2	D1	D0
CT2D7	CT2D6	CT2D5	CT2D4	CT2D3	CT2D2	CT2D1	CT2D0

Offset=0xF, RB='0'. 8254 Counter/Timer Configuration Register.

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
CT0D[7:0]	rw	-	8254 Counter/Timer Zero Data Register
CT1D[7:0]	rw	-	8254 Counter/Timer One Data Register
CT2D[7:0]	rw	-	8254 Counter/Timer Two Data Register
SC[1:0]	w	-	Select Counter: 00 = Select Counter 0 01 = Select Counter 1 10 = Select Counter 2 11 = Read-Back Command (see read operations)
RW[1:0]	w	-	Read/Write: 00 = Counter latch command 01 = Read/Write least significant byte only 10 = Read/Write most significant byte only 11 = Read/Write least significant byte first, then most significant byte.

M[2:0]	w	-	Counter/Timer Mode Select: 000 = Mode 0, Interrupt on terminal count 001 = Mode 1, Hardware retriggerable one-shot X10 = Mode 2, Rate generator X11 = Mode 3, Square wave generator 100 = Mode 4, Software triggered strobe 101 = Mode 5, Hardware triggered strobe (retriggerable)
BCD	w	-	Binary Coded Decimal (BCD) Counter (4 decades) if set to one, otherwise 16-bit binary counter.

Description

This register is available when Register Bank Status is '0' (see ADC Configuration (see page 72) Register bit RB).

See Also

Register Summary (see page 43)

ADC Configuration Register (see page 72)

8254 Data Sheet

Example

```
/*****************************************************************
/                          ANALOG INPUT 8254 COUNTER 1 & 2 TIMING
*/
static void STX104_AI_Timing_8254_Set( int board, long time_interval_ns )
{
    long high_count;
    long low_count;
    unsigned int octet;

    STX104_Set_Bank( board, 0 );

    /* assumes 10MHz clock (i.e. no 1MHz jumper) */
    low_count  = 10L; /* 1 microsecond intervals */

    high_count = time_interval_ns / 1000;
    while ( high_count > 65536L )
    {
        high_count = high_count >> 1;
        low_count  = low_count << 1;
    }
    while ( high_count < 2L )
    {
        high_count = high_count << 1;
        low_count  = low_count >> 1;
    }
#ifdef STX104_DEBUG_MODE
     fprintf( stdout, "low_count=%ld, high_count=%ld\n",low_count,high_count);
     fprintf( stdout, "Actual Interval (uSec) = %ld\n",time_interval_ns);
```

```
#endif
    /* set counter/timer 2  */
    outportb( stx104_base_address[board] + STX104_CT_CONFIGURATION, 0xB4 );
    octet = ((unsigned int) high_count) & 0x00FF;
    outp( stx104_base_address[board] + STX104_CT2_DATA, octet );
#ifdef STX104_DEBUG_MODE
    fprintf( stdout, "CT2 LB=%u\n",octet );
#endif
    octet = ((unsigned int) high_count) >> 8;
    outp( stx104_base_address[board] + STX104_CT2_DATA, octet );
#ifdef STX104_DEBUG_MODE
    fprintf( stdout, "CT2 HB=%u\n",octet );
#endif
    /* set counter/timer 1  */
    outportb( stx104_base_address[board] + STX104_CT_CONFIGURATION, 0x74 );
    octet = ((unsigned int) low_count) & 0x00FF;
    outp( stx104_base_address[board] + STX104_CT1_DATA, octet );
#ifdef STX104_DEBUG_MODE
    fprintf( stdout, "CT1 LB=%u\n",octet );
#endif
    octet = ((unsigned int) low_count) >> 8;
    outp( stx104_base_address[board] + STX104_CT1_DATA, octet );
#ifdef STX104_DEBUG_MODE
    fprintf( stdout, "CT1 HB=%u\n",octet );
#endif
}
```

11.26 FIFO Status LSB (Offset=15)

FIFO Status LSB Register. Please refer to FIFO Status MSB (see page 69) Register for details.

See Also

Register Summary (see page 43)

11.27 Index Data LSB (Offset=12, RB='1')

Indexed Data I/O least significant byte (or lower 8-bits). Please refer to Index Data (see page 81) Register for further details.

See Also

Register Summary (see page 43)

11.28 Index Data MSB (Offset=13, RB='1')

Indexed Data I/O most significant byte (or upper 8-bits). Please refer to Index Data (see page 81) Register for further details.

See Also

Register Summary (see page 43)

11.29 Index Data (Offset=12, RB='1')

Index Data Register

Register Layout

Offset=12, Byte 0. Offset=12, Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

Offset=13, Byte 1. Offset=12, Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
ID[15:0]	rw	-	Indexed Data

Description

See Also

Register Summary (see page 43)

Example

11.30 Index Pointer (Offset=14, RB='1')

Index Pointer Register. The purpose of utilizing an indexed array of registers is to fit a large number of registers into a small

region of I/O address space. The indexed array of registers are banked onto the 8254 I/O address space. At power up or reset, the entire STX104 register set will appear and function exactly as the previous firmware version of the STX104 card. By writing a special pattern to the ADC Configuration (see page 72) Register one can bank between the 8254 and indexed array registers as well as configure other STX104 registers for enhanced modes of operation.

Register Layout

Offset=14, Byte 0, RB='1'.

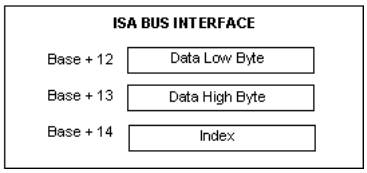
D7	D6	D5	D4	D3	D2	D1	D0
IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
IP[7:0]	rw	0x00	Index Pointer Register.

Description

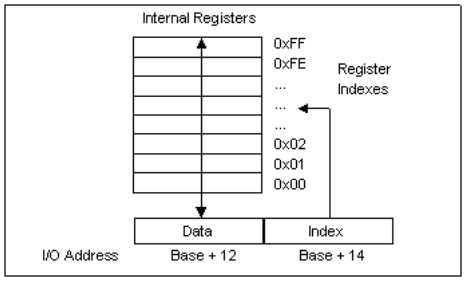
The drawing below illustrates the ISA bus interface to the indexed register array.



Index Register Description

The Index Register is an address pointer that points into the register set array. The Index Register can address any byte or word. Read or writing any data within the indexed array will cause the Index Register to automatically increment (if AIID='0', refer to General Configuration (see page 87) Register). If the data read or written is 16-bits, then the Index Register is automatic incremented by two. If the data read or written is 8-bits (one byte), then the Index Register is incremented by one. The auto-increment feature is implemented to reduce overall software overhead. The auto-increment feature can be disabled by setting the AIID bit in the General Configuration Register. One can modify the value of the Indexed Register at any time.

A logical representation of the indexed register array is illustrated below. The internal registers can consist of bytes (*signed* or *unsigned char*), words (*signed* or *unsigned ints*) or double words (*signed* or *unsigned long*). For advanced configuration it is possible to create a structure that can be simply uploaded to the indexed register array.



See Also

Register Summary (see page 43)

Example

Below are examples of functions used to read/write the indexed array. It is assumed that the indexed array has been properly banked (see page 72) into position. Since it is likely that the 8254 registers will not be regularly accessed in most situations, one could arrange to check/set the banking bit RB when one needs to read/write the 8254 register.

```
/*****************************************************************
/                                   INDEXED ARRAY DATA BYTE READ
*/
static unsigned char STX104_Read_Indexed_Data_Byte( int board, unsigned char index )
{
    unsigned char value;

    outp( stx104_base_address[board]  + STX104_INDEX_POINTER, index );
    value = (unsigned char) inp( stx104_base_address[board] + STX104_INDEX_DATA );
    return( value );
}
/*****************************************************************
/                                   INDEXED ARRAY DATA WORD READ
*/
static unsigned int STX104_Read_Indexed_Data_Word( int board, unsigned char index )
{
    unsigned int value;

    outp( stx104_base_address[board]  + STX104_INDEX_POINTER, index );
    value = inpw( stx104_base_address[board] + STX104_INDEX_DATA );
    return( value );
}
/*****************************************************************
/                                   INDEXED ARRAY DATA DWORD READ
*/
static unsigned long STX104_Read_Indexed_Data_Dword( int board, unsigned char index )
{
    union{ unsigned long dword; unsigned int word[2]; } dvalue;

    outp( stx104_base_address[board]  + STX104_INDEX_POINTER, index );
    dvalue.word[0] = inpw( stx104_base_address[board] + STX104_INDEX_DATA );
    dvalue.word[1] = inpw( stx104_base_address[board] + STX104_INDEX_DATA );
    return( dvalue.dword );
}
/*****************************************************************
/                                   INDEXED ARRAY DATA BYTE WRITE
*/
static void STX104_Write_Indexed_Data_Byte( int board, unsigned char index, unsigned char
value )
{
    outp( stx104_base_address[board] + STX104_INDEX_POINTER, index );
```

```
    outp( stx104_base_address[board] + STX104_INDEX_DATA,    (unsigned int) value );
}
/*****************************************************************
/                                   INDEXED ARRAY DATA WORD WRITE
*/
static void STX104_Write_Indexed_Data_Word( int board, unsigned char index, unsigned int value
)
{
    outp( stx104_base_address[board]  + STX104_INDEX_POINTER, index );
    outpw( stx104_base_address[board] + STX104_INDEX_DATA,    value );
}
/*****************************************************************
/                                   INDEXED ARRAY DATA DWORD WRITE
*/
static void STX104_Write_Indexed_Data_Dword( int board, unsigned char index, unsigned long
value )
{
    union{ unsigned long dword; unsigned int word[2]; } dvalue;
    dvalue.dword = value;
    outp( stx104_base_address[board]  + STX104_INDEX_POINTER, index );
    outpw( stx104_base_address[board] + STX104_INDEX_DATA,    dvalue.word[0] );
    outpw( stx104_base_address[board] + STX104_INDEX_DATA,    dvalue.word[1] );
}
```

11.31 Conversion Disable (Offset=1028; Index=64, RB='1')

Conversion Disable Register. DAS1602 Compatible Configuration Register. In 10-bit address decode mode, the DAS1602 compatible registers are also accessible through the indexed register set.

Register Layout

Offset=0x404, RB=X. Also located at: Index=0x40, Byte 0, RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
CD[7:0]	w	0x00	Conversion Disable Register. On power-up or reset the conversion triggers are enabled. This register is only available if FE bit is true (DAS1602 Functions are enabled). Writing a 0x00 to this register enables ADC triggering. Writing 0x40 (64_{10}) to this register disables ADC triggering. If the FIFO Superset jumper M1 is not installed, then when conversions are disabled, the FIFO is reset. If the jumper M1 is installed, the FIFO is only reset by writing to the channel register.

Description

Conversion Disable Register. On power-up or reset the conversion triggers are enabled. This register is only available if FE bit is true (DAS1602 Functions are enabled). Writing a 0x00 to this register enables ADC triggering. Writing 0x40 (64_{10}) to this register disables ADC triggering. If the FIFO Superset jumper M1 is not installed, then when conversions are disabled, the FIFO is reset.

If the jumper M1 is installed, the FIFO is only reset by writing to the channel register.

See Also

Register Summary (see page 43)

Example

11.32 Burst Mode Enable (Offset=1029; Index=65, RB='1')

ADC Burst Mode Enable Register. DAS1602 Compatible Configuration Register. In 10-bit address decode mode, the DAS1602 compatible registers are also accessible through the indexed register set.

Register Layout

Offset=0x405, RB=X. Also located at: Index=0x41, Byte 0, RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
ABME7	ABME6	ABME5	ABME4	ABME3	ABME2	ABME1	ABME0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
ABME[7:0]	w	0x00	ADC-Burst Mode Enable Register. On power-up or reset the ADC-burst mode is disabled. This register is only available if FE bit is true (DAS1602 Functions are enabled). Writing a 0x00 to this register disables ADC-burst mode. Writing 0x40 (6410) to this register enables ADC-burst mode.

Description

ADC Burst Mode Enable Register. On power-up or reset the ADC-burst mode is disabled. This register is only available if FE bit is true (DAS1602 Functions are enabled). Writing a 0x00 to this register disables ADC-burst mode. Writing 0x40 (6410) to this register enables ADC-burst mode.

See Also

Register Summary (see page 43)

Example

11.33 Burst Function Enable (Offset=1030; Index=66, RB='1')

ADC Function Enable Register. DAS1602 Compatible Configuration Register. In 10-bit address decode mode, the DAS1602 compatible registers are also accessible through the indexed register set.

Register Layout

Offset=0x406, RB=X. Also located at: Index=0x42, Byte 0, RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
FE7	FE6	FE5	FE4	FE3	FE2	FE1	FE0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
FE[7:0]	w	0x00 (function disabled)	DAS1602 Function Enable Register. On power-up or reset the DAS1602 functions are disabled. Writing 0x40 (64_{10}) to this register enables the DAS1602 functions. Writing a 0x00 to this register disables DAS1602 functions.

Description

DAS1602 Function Enable Register. On power-up or reset the DAS1602 functions are disabled. Writing 0x40 (64_{10}) to this register enables the DAS1602 functions. Writing a 0x00 to this register disables DAS1602 functions.

See Also

Register Summary (see page 43)

Example

11.34 Extended Status (Offset=1031; Index=67, RB='1')

ADC Extended Status Register. DAS1602 Compatible Configuration Register. In 10-bit address decode mode, the DAS1602 compatible registers are also accessible through the indexed register set.

Register Layout

Offset=0x407, RB=X. Also located at: Index=0x43, Byte 0, RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
0	ABME	FE	CD	0	0	WS	CLK

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
ABME	r	0	ADC-Burst Mode Enabled: 0 = disabled (default) 1 = enabled
FE	r	0	DAS1602 Function Enabled: 0 = disabled, DAS16jr/16 functionality only (default) 1 = enabled
CD	r	1	Conversions Allowed: 0 = ADC Conversions Disabled 1 = ADC Conversions Allowed (default)
WS	r	0	Wait States. This bit is always zero for no wait states.
CLK	r	J6.1MHz Jumper	Counter/Timer clock source: 0 = 1MHz selected (based on jumper 1MHz) 1 = 10MHz selected

Description

See Also

Register Summary (see page 43)

Example

11.35 General Configuration (Index=0, RB='1')

General Configuration Register

Register Layout

Index=0x00, Byte 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
AIID	CLK	DMAJ/MJ5	M4J	M3J	M2J	M1J	M0J

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
AIID	rw	0	Auto Index Increment Disable. Index register is automatically incremented by the size (byte/word) of the data that is read or written. 0 = Auto index increment enabled (default) 1 = Auto index increment disabled
CLKJ	r	J6.1MHZ jumper	Mode jumper 1MHZ state at connector J6. 0 = Jumper is not installed 1 = Jumper is installed
DMAJ/MJ5	r	J6.DMA jumper/J6.M5 jumper	Mode jumper DMA state at connector J6 or Mode jumper M4 state at connector J6(Revision 160923 or newer) 0 = Jumper is not installed 1 = Jumper is installed
M4J	r	J6.M4 jumper	Mode jumper M4 state at connector J6. 0 = Jumper is not installed 1 = Jumper is installed
M3J	r	J6.M3 jumper	Mode jumper M3 state at connector J6. 0 = Jumper is not installed 1 = Jumper is installed
M2J	r	J6.M2 jumper	Mode jumper M2 state at connector J6. 0 = Jumper is not installed 1 = Jumper is installed
M1J	r	J6.M1 jumper	Mode jumper M1 state at connector J6. 0 = Jumper is not installed 1 = Jumper is installed
M0J	r	J6.M0 jumper	Mode jumper M0 state at connector J6. 0 = Jumper is not installed 1 = Jumper is installed

Description

See Also

Register Summary (see page 43)

Example

11.36 Interrupt Source Select (Index=2, RB='1')

Interrupt Source Select Register

Register Layout

Index=0x02, Byte 0, Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
ISB3	ISB2	ISB1	ISB0	ISA3	ISA2	ISA1	ISA0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
ISB[3:0]	rw	0000	Interrupt Source B. Sets the ISB bit in the ADC Status Register when ISSBE='1'. 0000 = none (default) 0001 = Reserved 0010 = Interrupt Threshold (see page 92) Counter 0011 = Analog Input Frame Maximum (see page 108) 0100 = Trigger Start and transition to active trigger state 0101 = Trigger Stop and transition to inactive trigger state 0110 = none 0111 = none 1000 = DIN0 rising edge 1001 = DIN0 falling edge 1010 = DIN1 rising edge 1011 = DIN1 falling edge 1100 = DIN2 rising edge 1101 = DIN2 falling edge 1110 = DIN3 rising edge 1111 = DIN3 falling edge

ISA[3:0]	rw	0000	Interrupt Source A. Sets the ISA bit in the ADC Status Register when ISSBE='1'. 0000 = none (default) 0001 = Reserved 0010 = Interrupt Threshold (see page 92) Counter 0011 = Analog Input Frame Maximum (see page 108) 0100 = Trigger Start and transition to active trigger state 0101 = Trigger Stop and transition to inactive trigger state 0110 = none 0111 = none 1000 = DIN0 rising edge 1001 = DIN0 falling edge 1010 = DIN1 rising edge 1011 = DIN1 falling edge 1100 = DIN2 rising edge 1101 = DIN2 falling edge 1110 = DIN3 rising edge 1111 = DIN3 falling edge

Description

See Also

Register Summary (see page 43)

Example

11

11.37 Interrupt Configuration (Index=4, RB='1')

Interrupt Source Select Register

Register Layout

Index=0x04, Byte 0, Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	FIBLK3	FIBLK2	FIBLK1	FIBLK0

Index=0x05, Byte 1, Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
EIS	ISSBE	nINT_FF	SITT	ITS3	ITS2	ITS1	ITS0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
EIS	rw	0	Enhanced Interrupt Selection 0 = Normal interrupt selection in ADC control register (see page 66) (default) 1 = Enhanced interrupt selection in ADC control register
ISSBE	rw	0	Interrupt Source Status Bank Enable 0 = ADC channel mapped to ADC Status Register (see page 64) (default) 1 = Interrupt Source Status mapped to lower nibble of ADC Status Register
nINT_FF	rw	0	Interrupt Flip-Flop Disable. Classic DMA and FIFO interrupt generation is independent of this setting. 0 = Allows classic ADC-sample and ADC-bursts to generate interrupts (default) 1 = Disallows or blocks classic ADC-sample and ADC-bursts from generating interrupts. Setting the DMA bit, however over-rides this function. Use alternate interrupt generation to achieve desired interrupts.
ITS[3:0]	rw	0000	Interrupt Threshold Counter Source Select. Increment by: 0000 = none (default) 0001 = Sample or Frame (see page 15) (i.e. ADC burst) 0010 = Block (i.e. 256 samples) 0011 = Analog Input Maximum Frame (see page 15) Count 0100 = Trigger Start and transition to active trigger state 0101 = Trigger Stop and transition to inactive trigger state 0110 = none 0111 = none 1000 = DIN0 rising edge 1001 = DIN0 falling edge 1010 = DIN1 rising edge 1011 = DIN1 falling edge 1100 = DIN2 rising edge 1101 = DIN2 falling edge 1110 = DIN3 rising edge 1111 = DIN3 falling edge
SITT	rw	0	Sync Interrupt Threshold Counter to the beginning of active trigger. This can be used to synchronize interrupts to trigger start. 0 = disabled (default) 1 = Clear Interrupt Threshold Counter to the beginning of the active trigger.

FIBLK[3:0]	rw	0000	Number of Blocks to generate a FIFO Interrupt. This is the number of samples written to the FIFO in order to generate an interrupt; if the sample timing is constant, then this interrupt will be at a constant rate as well. 0000 = 2 Blocks or 512 Samples (default) 0001 = 4 Blocks or 1,024 Samples 0010 = 8 Blocks or 2,048 Samples 0011 = 16 Blocks or 4,096 Samples 0100 = 32 Blocks or 8,192 Samples 0101 = 64 Blocks or 16,384 Samples 0110 = 128 Blocks or 32,768 Samples 0111 = 256 Blocks or 65,536 Samples 1000 = 512 Blocks or 131,072 Samples 1001 = 1024 Blocks or 262,144 Samples 1010 = 2048 Blocks or 524,288 Samples 1011 = 0.0625 Block or 16 Samples 1100 = 0.125 Block or 32 Samples 1101 = 0.25 Block or 64 Samples 1110 = 0.5 Block or 128 Samples 1111 = 1 Block or 256 Samples
X	-	-	Don't Care

Description

See Also

Register Summary (see page 43)

Example

11.38 Interrupt Threshold (Index=8, RB='1')

Interrupt Threshold Register

Register Layout

Index=0x08, Byte 0. Index=0x08, Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
ITH7	ITH6	ITH5	ITH4	ITH3	ITH2	ITH1	ITH0

Index=0x09, Byte 1. Index=0x08, Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
ITH15	ITH14	ITH13	ITH12	ITH11	ITH10	ITH9	ITH8

Index=0x0A, Byte 2. Index=0x0A, Word 1. RB='1'.

D23	D22	D21	D20	D19	D18	D17	D16
ITH23	ITH22	ITH21	ITH20	ITH19	ITH18	ITH17	ITH16

Index=0x0B, Byte 3. Index=0x0A, Word 1. RB='1'.

D31	D30	D29	D28	D27	D26	D25	D24
ITH31	ITH30	ITH29	ITH28	ITH27	ITH26	ITH25	ITH24

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
ITH[31:0]	rw	0x00000000	Interrupt Threshold Register. See Description section for further information.

Description

When the number of events defined by the Interrupt Threshold Source Counter Select (refer to the Interrupt Source Select (see page 89) Register) is reached, an output is generated which can subsequently be used to generate an interrupt or be used to cause an ADC-sample or ADC-burst, depending on configurations.

It is possible for this counter to be used as a form of an event pre-scalar for ADC-sampling or ADC-bursting or interrupt generation.

One time: Interrupt_time_interval = ITH * interrupt_count_source_time_interval.

Periodic: Interrupt_time_interval = (ITH + 1) * interrupt_count_source_time_interval.

See Also

Register Summary (see page 43)

Example

11.39 Digital Output Configuration (Index=12, RB='1')

Digital Output Configuration Register

Register Layout

Index=0x0C, Byte 0. Index=0x0C, Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
SSHP	X	CT0OP	CT2OP	DOP3	DOP2	DOP1	DOP0

Index=0x0D, Byte 1. Index=0x0C, Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
DO3_MAP1	DO3_MAP0	DO2_MAP1	DO2_MAP0	DO1_MAP1	DO1_MAP0	DO0_MAP1	DO0_MAP0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
DO3_MAP[1:0]	rw	00	Digital Output DOUT3 (J7.5) Mapping. NOTE: This selection occurs before polarity control. 00 = Digital Output Register.DOUT3. This is the default Digital Output Register (see page 56). 01 = unused 10 = unused 11 = unused
DO2_MAP[1:0]	rw	00	Digital Output DOUT2 (J7.6) Mapping. NOTE: This selection occurs before polarity control. 00 = Digital Output Register.DOUT2. This is the default Digital Output Register (see page 56). 01 = SPI Module Chip Select Output (High Voltage or High Gain Buffer Board) 10 = unused 11 = unused
DO1_MAP[1:0]	rw	00	Digital Output DOUT1 (J7.7) Mapping. NOTE: This selection occurs before polarity control. 00 = Digital Output Register.DOUT1. This is the default Digital Output Register (see page 56). 01 = SPI Module Clock Output (High Voltage or High Gain Buffer Board) 10 = unused 11 = unused

DO0_MAP[1:0]	rw	00	Digital Output DOUT0 (J7.8) Mapping. NOTE: This selection occurs before polarity control. 00 = Digital Output Register.DOUT0. This is the default Digital Output Register (see page 56). 01 = SPI Module Data Output (High Voltage or High Gain Buffer Board) 10 = unused 11 = unused
SSHP	rw	0	SSH (J7.14) Polarity 0 = Non-inverting (default) 1 = Inverted
CT0OP	rw	0	CT_OUT0 (J7.3) Polarity 0 = Non-inverting (default) 1 = Inverted
CT2OP	rw	0	CT_OUT2 (J7.2) Polarity 0 = Non-inverting (default) 1 = Inverted
DOP3	rw	0	Digital Output DOUT3 (J7.5) Polarity 0 = Non-inverting (default) 1 = Inverted
DOP2	rw	0	Digital Output DOUT2 (J7.6) Polarity 0 = Non-inverting (default) 1 = Inverted
DOP1	rw	0	Digital Output DOUT1 (J7.7) Polarity 0 = Non-inverting (default) 1 = Inverted
DOP0	rw	0	Digital Output DOUT0 (J7.8) Polarity 0 = Non-inverting (default) 1 = Inverted
X	-	-	Don't Care

Description

The Digital Output Configuration Register sets the polarity on the digital outputs and the SSH pin.

See Also

Register Summary (see page 43)

Example

11.40 Digital Input Configuration (Index=14, RB='1')

Digital Input Register

Register Layout

Index=0x0E, Byte 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
SDGF	SDI31	X	X	DIP3	DIP2	DIP1	DIP0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
SDGF	rw	0	Short Deglitch Filter 0 = 200 nSec deglitch (default) 1 = 100 nSec deglitch
X	-	-	Don't Care
SDI31	rw	0	Swap DIN1 and DIN3 0 = Classic mode. DIN1 wired to J7.11 and DIN3 to J7.9 to support existing customers. 1 = Normal mode. DIN1 wired to J7.9 and DIN3 to J7.11.
DIP3	rw	0	DIN3 (J7.9) Polarity 0 = Non-inverting (default) 1 = Inverted
DIP2	rw	0	DIN2 (J7.10) Polarity 0 = Non-inverting (default) 1 = Inverted
DIP1	rw	0	DIN1 (J7.11) Polarity 0 = Non-inverting (default) 1 = Inverted
DIP0	rw	0	DIN0 (J7.12) Polarity 0 = Non-inverting (default) 1 = Inverted

Description

Sets the polarity and deglitch filters for the digital inputs.

See Also

Register Summary (see page 43)

Example

11.41 Trigger Configuration (Index=16, RB='1')

Trigger Configuration Register

Register Layout

Index=0x10 Byte 0. Index=0x10 Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
ETS3	ETS2	ETS1	ETS0	STS3	STS2	STS1	STS0

Index=0x11 Byte 1. Index=0x10 Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
TEN	X	STM1	STM0	TSS3	TSS2	TSS1	TSS0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
X	-	-	Don't Care
TEN	rw	0	Trigger Enable 0 = disabled (default) 1 = enabled
STM[1:0]	rw	00	Start Trigger Mode. 00 = trigger source (default) 01 = trigger followed by trigger delay 10 = trigger followed by sync followed by trigger delay 11 = Reserved

TSS[3:0]	rw	0000	Start Trigger Synchronization Source. 0000 = none (default) 0001 = writing 0x5A to the software strobe register 0010 = none 0011 = Analog Input Sample/Frame (see page 15) Timer 0010 = none 0011 = none 0100 = 8254 Counter 0 Output (CT_OUT0) rising-edge 0101 = 8254 Counter 0 Output (CT_OUT0) falling-edge 0110 = 8254 Counter 2 Output (CT_OUT2) rising-edge 0111 = 8254 Counter 2 Output (CT_OUT2) falling-edge 1000 = DIN0 rising edge 1001 = DIN0 falling edge 1010 = DIN1 rising edge 1011 = DIN1 falling edge 1100 = DIN2 rising edge 1101 = DIN2 falling edge 1110 = DIN3 rising edge 1111 = DIN3 falling edge
ETS[3:0]	rw	0000	End/Stop Trigger Source 0000 = none (default) 0001 = writing 0xAA to the software strobe register 0010 = Analog Input Frame (see page 15) Maximum 0011 = Analog Input Sample/Frame (see page 15) Timer 0100 = 8254 Counter 0 Output (CT_OUT0) rising-edge 0101 = 8254 Counter 0 Output (CT_OUT0) falling-edge 0110 = 8254 Counter 2 Output (CT_OUT2) rising-edge 0111 = 8254 Counter 2 Output (CT_OUT2) falling-edge 1000 = DIN0 rising edge 1001 = DIN0 falling edge 1010 = DIN1 rising edge 1011 = DIN1 falling edge 1100 = DIN2 rising edge 1101 = DIN2 falling edge 1110 = DIN3 rising edge 1111 = DIN3 falling edge

STS[3:0]	rw	0000	Start Trigger Source 0000 = none (default) 0001 = writing 0x55 to the software strobe register 0010 = none 0011 = Analog Input Sample/Frame (see page 15) Timer 0100 = 8254 Counter 0 Output (CT_OUT0) rising-edge 0101 = 8254 Counter 0 Output (CT_OUT0) falling-edge 0110 = 8254 Counter 2 Output (CT_OUT2) rising-edge 0111 = 8254 Counter 2 Output (CT_OUT2) falling-edge 1000 = DIN0 rising edge 1001 = DIN0 falling edge 1010 = DIN1 rising edge 1011 = DIN1 falling edge 1100 = DIN2 rising edge 1101 = DIN2 falling edge 1110 = DIN3 rising edge 1111 = DIN3 falling edge

Description

Configures the trigger section for a variety of ADC data collection start and stop situations.

See Also

Register Summary (see page 43)

Example

11.42 Trigger Start Delay (Index=20, RB='1')

Trigger Start Delay Register. Range is 0 to 107.37 Seconds in steps of 25 nanoseconds.

Register Layout

Index=0x14 Byte 0. Index=0x14 Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
TSD7	TSD6	TSD5	TSD4	TSD3	TSD2	TSD1	TSD0

Index=0x15 Byte 1. Index=0x14 Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
TSD15	TSD14	TSD13	TSD12	TSD11	TSD10	TSD9	TSD8

Index=0x16 Byte 2. Index=0x16 Word 1. RB='1'.

D23	D22	D21	D20	D19	D18	D17	D16
TSD23	TSD22	TSD21	TSD20	TSD19	TSD18	TSD17	TSD16

Index=0x17 Byte 3. Index=0x16 Word 1. RB='1'.

D31	D30	D29	D28	D27	D26	D25	D24
TSD31	TSD30	TSD29	TSD28	TSD27	TSD26	TSD25	TSD24

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
TSD[31:0]	rw	0x00000000	

Description

The actual delay interval is calculated as:

Delay_nanoseconds = 25 * TSD. Where 0 <= TSD <= 4,294.967.295. UINT32_MAX.

Thus, the trigger start delay is adjustable from 0 nSec to 107.37 Seconds.

Range: 0 to 107.37 Seconds.

Resolution: 25 nanoseconds.

See Also

Register Summary (see page 43)

Example

11.43 Analog Input General Configuration (Index=32, RB='1')

Analog Input General Configuration Register

Register Layout

Index=0x20, Byte 0. Index=0x20, Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
MAVG_INV	SAIFTTS	SAIFCTS	nSGATE	AISS3	AISS2	AISS1	AISS0

Index=0x21, Byte 1. Index=0x20, Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
X	X	X	X	X	AI_ENCODE	AI_ENCODE_BEHAVIOR1	AI_ENCODE_BEHAVIOR0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
X	-	-	Don't Care
AI_ENCODE_BEHAVIOR	rx	00	Analog Input data encoding behavior. If AI_ENCODE='0' then straight binary and the following behavior: X0 = Normal (legacy default). Polarity bit ignored. Most negative (0x0000) to most positive (0xFFFF). X1 = Multiply by -1, but only if polarity bit for channel is set. polarity = '1': Most negative (0xFFFF) to most positive (0x0000). polarity = '0'. Most negative (0x0000) to most positive (0xFFFF). If AI_ENCODE='1' then two's complement encoding with the following behavior: 00 = Normal. Polarity bit ignored. Most negative (0x8000) to most postive (0x7FFF). 01 = Multiply by -1, but only if polarity bit for channel is set. (NOTE: 0x8000 value is suppressed.) ai_polarity[channel] = '1': Most negative (0x7FFF) to most positive (0x8001). ai_polarity[channel] = '0'. Most negative (0x8001) to most positive (0x7FFF). 1X = Absolute value. ai_polarity[channel] = '1': Range: most negative (0x8001) to most positive (0x0000). ai_polarity[channel] = '0'. Range: most negative (0x0000) to most positive (0x7FFF).
AI_ENCODE	rw	0	Analog Input data encoding or formatting. Straight binary or twos complement for all analog input values. 0 = Straight Binary (default). Most negative (0x0000) to most positive (0xFFFF). 1 = Twos Complement. Most negative (0x8000) to most postive (0x7FFF).

MAVG_INV	rw	0	Invert the state of the moving average filter jumper (M4). This allows one to enable/disable moving average filter through software. 0 = non-inverted or normal operation (default) 1 = invert state of jumper to enable/disable moving average filter function. Observe the state at General Configuration (see page 87) Register. If the M4J bit is set in the General Configuration Register, then the moving average filter is enabled.
SAIFTTS	rw	0	Sync Analog Input Frame (see page 15) Timer to Trigger Start. If set, then also moves the first sample to just after the trigger start rather than sampling at any time (i.e. 0 to 1 sample interval) after the trigger start. 0 = disabled (default). Timing between samples remains fixed regardless of point of trigger. 1 = Clear Analog Input Frame (see page 15) Timer upon Trigger Start. Sampling begins just after the trigger-start, thus synchronizing the sampling to the trigger start.
SAIFCTS	rw	0	Sync Analog Input Frame (see page 15) Counter to Trigger Start. This can be used to synchronize sample or frame counting to a trigger start. 0 = disabled (default). 1 = Clear Analog Input Frame (see page 15) Counter upon Trigger Start
nSGATE	rw	0	Sampling Gate 0 = Ignore triggering subsystem (default) 1 = Use triggering subsystem for ADC-sampling
AISS[3:0]	rw	0000	Analog Input Sampling Sources. This select the signal that is used to generate ADC-samples or ADC-bursts. 0000 = use ALSS[1:0] configuration (default, legacy ADC-sampling/-burst source) 0001 = Analog Input Sample/Frame (see page 15) Timer. 0010 = Interrupt Threshold Counter 0011 = none 0100 = none 0101 = none 0110 = none 0111 = none 1000 = DIN0 rising edge 1001 = DIN0 falling edge 1010 = DIN1 rising edge 1011 = DIN1 falling edge 1100 = DIN2 rising edge 1101 = DIN2 falling edge 1110 = DIN3 rising edge 1111 = DIN3 falling edge

Description

The Analog Input General Configuration Register configures the source of ADC-sampling or ADC-burst signalling (in legacy terms it is the old ADC-triggering sources). Upon reset or power up it defaults to legacy mode of operation.

See Also

Register Summary (see page 43)

Example

11.44 Analog Input Polarity (Index=33, RB='1')

Analog Input Polarity Register

Register Layout

Index=0x22, Byte 0. Index=0x22, Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
AIPOL7	AIPOL6	AIPOL5	AIPOL4	AIPOL3	AIPOL2	AIPOL1	AIPOL0

Index=0x23, Byte 1. Index=0x22, Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
AIPOL15	AIPOL14	AIPOL13	AIPOL12	AIPOL11	AIPOL10	AIPOL9	AIPOL8

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
AIPOL[15:0]	rw	0x0000	Polarity setting for each analog input channel. This allows for the bit patterns to be inverted or multiplied by -1. For each bit: 0 = normal 1 = polarity inverted (straight binary encoding) or multiply by -1 (twos complement). Please refer to the description section for further details.

Description

This register is used to control the analog input polarity for each of the given analog input channels. The "aicfg.AI_ENCODE" is the Analog Input General Configuration (see page 100) Register AI_ENCODE bit. The "aicfg.AI_ENCODE_BEHAVIOR[1:0]" is the Analog Input General Configuration (see page 100) Register AI_ENCODE_BEHAVIOR[1:0] bits. The "aipol(x)" is the polarity bit for a given analog input channel as defined within the bit pattern for the Analog Input Polarity Register.

aicfg.AI_ENCODE	aicfg.AI_ENCODE_BEHAVIOR[1:0]	aipol(x)	Resulting Analog Input Bit Encoding at Channel x (shown most negative to most positive word encoding)
0	X0	X	Straight Binary: 0x0000 - 0xFFFF (default legacy mode)

0	X1	0	Straight Binary: 0x0000 - 0xFFFF
0	X1	1	Bitwise inverted or one's complement Straight Binary: 0xFFFF - 0x0000
1	00	X	Two's Complement: 0x8000 to 0xFFFF, then 0x0000 to 0x7FFFF
1	01	0	Twos Complement: 0x8001 to 0xFFFF, then 0x0000 to 0x7FFFF (0x8000 re-assigned to 0x8001)
1	01	1	-1 * Twos Complement: 0X7FFF to 0x0000, then 0xFFFF to 0x8001 (0x8000 re-assigned to 0x8001)
1	1X	0	Absolute value(Two's Complement): 0x0000 to 0x7FFFF (0x8000 re-assigned to 0x8001, and 0x8001 to 0xFFFF multiplied by -1)
1	1X	1	-1 * Absolute value(Two's Complement): 0x8001 to 0xFFFFF (0x8000 re-assigned to 0x8001, and 0x0001 to 0x7FFF multiplied by -1)

See Also

Register Summary (see page 43)

Example

11.45 Analog Input Frame Timer (Index=36, RB='1')

Analog Input Frame (see page 15) Timer Register. Range is 5 uSec to 53.68 Seconds in steps of 25 nanoseconds.

Register Layout

Index=0x24 Byte 0. Index=0x24 Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
AIFT7	AIFT6	AIFT5	AIFT4	AIFT3	AIFT2	AIFT1	AIFT0

Index=0x25 Byte 1. Index=0x24 Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
AIFT15	AIFT14	AIFT13	AIFT12	AIFT11	AIFT10	AIFT9	AIFT8

Index=0x26 Byte 2. Index=0x26 Word 1. RB='1'.

D23	D22	D21	D20	D19	D18	D17	D16
AIFT23	AIFT22	AIFT21	AIFT20	AIFT19	AIFT18	AIFT17	AIFT16

Index=0x27 Byte 3. Index=0x26 Word 1. RB='1'.

D31	D30	D29	D28	D27	D26	D25	D24
AIFT31	AIFT30	AIFT29	AIFT28	AIFT27	AIFT26	AIFT25	AIFT24

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
AIFT[31:0]	rw	0x00000000	

Description

The actual time interval is calculated as:

ai_time_frame_ns = (AIFT + 1) * (25 nSec).

Where 0 <= AIFT <= 2147483648.

Anything below 5 uSec will be limited by the intra-sample timing which is limited to 5 uSec at a minimum.

Thus, the ADC frame time is adjustable from 5 microseconds to 53.68 seconds in 25 nanosecond steps.

Range: 5 uSec to 53.68 Seconds.

Resolution: 25 nanoseconds.

See Also

Register Summary (see page 43)

Example

11.46 Analog Input Burst Timer (Index=40, RB='1')

Analog Input Burst Timer. Adjusts timing between samples during ADC-burst mode.

Register Layout

Index=0x28, Byte 0. Index=0x28, Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
AIBT7	AIBT6	AIBT5	AIBT4	AIBT3	AIBT2	AIBT1	AIBT0

Index=0x29, Byte 1. Index=0x28, Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
AIBT15	AIBT14	AIBT13	AIBT12	AIBT11	AIBT10	AIBT9	AIBT8

Index=0x2A, Byte 2. Index=0x2A, Word 1. RB='1'.

D23	D22	D21	D20	D19	D18	D17	D16
AIBT23	AIBT22	AIBT21	AIBT20	AIBT19	AIBT18	AIBT17	AIBT16

Index=0x2B, Byte 3. Index=0x2A, Word 1. RB='1'.

D31	D30	D29	D28	D27	D26	D25	D24
AIBT31	AIBT30	AIBT29	AIBT28	AIBT27	AIBT26	AIBT25	AIBT24

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
AIBT[31:0]	rw	0x00000000	

Description

Adjusts the time between ADC-samples in a ADC-Burst mode (i.e. the intra-sample time). In legacy mode this is 5 microseconds.

The actual time interval is calculated as:

ai_time_intra_sample_ns = AIBT * (25 nSec) + (5000 nSec).

Where 0 <= AIBT <= 2147483648.

Thus, the ADC burst time (or intra-sample time) is adjustable from 5 microseconds to 53.68 seconds in 25 nanosecond steps.

Range: 5 uSec to 53.68 Seconds.

Resolution: 25 nanoseconds.

How to determine optimal ADC-Burst timing value? Since the ADC input is multiplexed, the speed limitation is the overall RC time constant or settling time to achieve 16-bit resolution. The equation below describes how to determine minimum intra-burst timing.

Settling Time in ADC-Burst Mode:

Some background. A burst, once started will sample from *first_channel* to *last_channel* in 5uS intervals in legacy mode. So, eight channels will take 40uS to complete. The trigger for the next burst can actually come early (i.e. before the last sample completes) and it should start the next burst immediately as if one is continuously sampling at 5uS/sample.

As with all multiplexed front-ends, it is desirable to keep the source impedance of the signals driving each of the inputs as low as possible to minimize cross-talk (i.e. more specifically settling time). Differential (see page 19) inputs have 47pF input capacitance and single ended inputs have 27pF input capacitance (we tend to lean on the high side or worse case). So, to keep

signal settle times to less than 1 LSB requires:

Rmax <= (settle_time) / [Cin * bits_resolution * ln(2)]

<= 4uS / [47pF * 16 * ln(2)]

<= 7674 ohms.

You will want to keep the devices driving the STX104 inputs with source impedance of less than 8K ohms when operating in Burst mode at its maximum speed and assuming no other source of input capacitance (short cables, for example).

This equation above comes from:

V(t) = Vf + (Vi - Vf) exp(-t / RC)

in which you go from maximum to minimum input voltages

(i.e. +10V to -10V and settle to within 1 LSB).

Note that in the above calculation we set the settle time to within 1 LSB over 4uS time frame; we could go as far as 5uS, but 4uS gives us a little extra margin.

Increasing Settle Time in ADC-Burst Mode:

To increase settle time, you have to set the Analog Input Burst Timer to a non-zero value. Here is an example with a sensor with a source impedance of 10K ohms.

Settle_time = Rin * [Cin * bits_resolution * ln(2)]

= 10K * [300pF * 16 * ln(2)] = 11.09 * Rin * Cin

= 34uS

Thus, the Analog Input Burst Timer Register should be set to:

AIBT = [(34 uSec) - (5 uSec)] / 25 nSec = 1160 or 0x00000488.

See Also

Register Summary (see page 43)

Example

11.47 Analog Input Frame Maximum (Index=44, RB='1')

Analog Input Frame (see page 15) Maximum Register. Maximum Frame (see page 15) range is 0 to 2,147,483,648 Frames.

Register Layout

Index=0x2C, Byte 0. Index=0x2C, Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
AIFM7	AIFM6	AIFM5	AIFM4	AIFM3	AIFM2	AIFM1	AIFM0

Index=0x2D, Byte 1. Index=0x2C, Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
AIFM15	AIFM14	AIFM13	AIFM12	AIFM11	AIFM10	AIFM9	AIFM8

Index=0x2E, Byte 2. Index=0x2E, Word 1. RB='1'.

D23	D22	D21	D20	D19	D18	D17	D16
AIFM23	AIFM22	AIFM21	AIFM20	AIFM19	AIFM18	AIFM17	AIFM16

Index=0x2F, Byte 3. Index=0x2E, Word 1. RB='1'.

D31	D30	D29	D28	D27	D26	D25	D24
AIFM31	AIFM30	AIFM29	AIFM28	AIFM27	AIFM26	AIFM25	AIFM24

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
AIFM[31:0]	rw	0x00000000	Analog Input Frame (see page 15) Maximum

Description

When the value of the Analog Input Frame (see page 15) Counter reaches the value in the Analog Input Frame (see page 15) Maximum Register, and output is generated which can be used to generate such things as interrupts, trigger-stops, etc.

Frame (see page 15) Count Maximum Range: 1 <= AIFM <= 2147483648.

Range: 0 to 2147483648 Frames.

Resolution: 1 Frame (see page 15).

Counting every ADC-sample at 5 microsecond sample rate, the maximum can be set to just under 3 hours.

Counting every ADC-Burst at 50uSec/ADC-Burst (20KHz), the maximum can be set to just under 29 hours.

See Also

Register Summary (see page 43)

Example

11.48 Analog Input Frame Counter (Index=48, RB='1')

Analog Input Frame (see page 15) Count Register.

Register Layout

Index=0x30, Byte 0. Index=0x30, Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
AIFC7	AIFC6	AIFC5	AIFC4	AIFC3	AIFC2	AIFC1	AIFC0

Index=0x31, Byte 1. Index=0x30, Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
AIFC15	AIFC14	AIFC13	AIFC12	AIFC11	AIFC10	AIFC9	AIFC8

Index=0x32, Byte 2. Index=0x32, Word 1. RB='1'.

D23	D22	D21	D20	D19	D18	D17	D16
AIFC23	AIFC22	AIFC21	AIFC20	AIFC19	AIFC18	AIFC17	AIFC16

Index=0x33, Byte 3. Index=0x32, Word 1. RB='1'.

D31	D30	D29	D28	D27	D26	D25	D24
AIFC31	AIFC30	AIFC29	AIFC28	AIFC27	AIFC26	AIFC25	AIFC24

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
AIFC	r	0x00000000	Analog Input Frame (see page 15) Count

Description

The Analog Input Frame (see page 15) Count Register indicates the actual number of samples (when ADC set for non-ADC-bursting mode of operation) or the number of frames (when ADC set for ADC-burst mode of operation).

See Also

Register Summary (see page 43)

Example

11

11.49 Miscellaneous Output Configuration Register (Index=208, RB='1')

Miscellaneous Output Configuration Register

Register Layout

Index=0xD0, Byte 0. Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
CTOUT2PE	CTOUT2SEL2	CTOUT2SEL1	CTOUT2SEL0	CTOUT0PE	CTOUT0SEL2	CTOUT0SEL1	CTOUT0SEL0

Index=0xD1, Byte 1. Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
X	X	X	X	X	X	X	X

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
CTOUT2PE	rw	0	CT_OUT2 (J7.2) pulse extender 0 = disabled (default) 1 = enabled, extend selected signal by 100nSec approximately.
CTOUT2SEL[2:0]	rw	000	CT_OUT2 (J7.2) source select 000 = 8254 CT2 output (default) 001 = Digital Output, DOUT5. Reference Digital Outputs (see page 56) Register. 010 = Analog Input ADC-sample / ADC-Burst sample pulse 011 = Active ADC Sampling 100 = Analog Input Frame (see page 15) Timer 101 = Interrupt Counter Increment 110 = zero 111 = IOCHRDY
CTOUT0PE	rw	0	CT_OUT0 (J7.3) pulse extender 0 = disabled (default) 1 = enabled, extend selected signal by 100nSec approximately.
CTOUT0SEL[2:0]	rw	000	CT_OUT0 (J7.3) source select 000 = 8254 CT0 output (default) 001 = Digital Output, DOUT4. Reference Digital Outputs (see page 56) Register. 010 = Trigger delay end pulse 011 = Trigger delay start pulse 100 = Interrupt Threshold Counter maximum reached 101 = Active ADC Sampling 110 = ADC Data MSB address decode 111 = Interrupt

Description

This register is used to configure the CT_OUT2 (J7.2) and the CT_OUT0 (J7.3) outputs. In other words, these outputs are now generalized and various signals can be routed to these outputs.

See Also

Register Summary (see page 43)

Example

11.50 FIFO Data Available (Index=224, RB='1')

FIFO Data Available Register

Register Layout

Index=0xE0, Byte 0. Index=0xE0, Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
FDA7	FDA6	FDA5	FDA4	FDA3	FDA2	FDA1	FDA0

Index=0xE1, Byte 1. Index=0xE0, Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
FDA15	FDA14	FDA13	FDA12	FDA11	FDA10	FDA9	FDA8

Index=0xE2, Byte 2. Index=0xE2, Word 1. RB='1'.

D23	D22	D21	D20	D19	D18	D17	D16
FDA23	FDA22	FDA21	FDA20	FDA19	FDA18	FDA17	FDA16

Index=0xE3, Byte 3. Index=0xE2, Word 1. RB='1'.

D31	D30	D29	D28	D27	D26	D25	D24
FDA31	FDA30	FDA29	FDA28	FDA27	FDA26	FDA25	FDA24

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
FDA	r	0x00000000	FIFO Data Available. Number of analog input samples that remain in the FIFO.

Description

The number of 16-bit words in the FIFO or buffer.

See Also

Register Summary (see page 43)

Example

11.51 FIFO Configuration (Index=228, RB='1')

FIFO Configuration Register

Register Layout

Index=0xE4, Byte 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
HSFIFOEN	X	X	X	X	X	X	X

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
X	-	-	Don't Care
HSFIFOEN	rw	0	High Speed pre-queue FIFO Buffer Enable 0 = Disabled, utilize main memory only (default) 1 = Enabled, use high speed CPU buffer

Description

Enabling the High Speed CPU FIFO Buffer can reduce bus wait states generated due to waiting for STX104 main memory data availability. In many cases, bus wait states (due to IOCHRDY) are eliminated. We found that overall throughput through the ISA bus was improved by approximately 15%.

By default the CPU FIFO Buffer is disabled in order to maintain classic timing characteristics which might be critical to a customer completed application.

See Also

Register Summary (see page 43)

Example

11.52 Scratch Pad (Index=248, RB='1')

Scratch Pad Register

Register Layout

Index=0xF8, Byte 0. Index=0xF8 Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
SCR7	SCR6	SCR5	SCR4	SCR3	SCR2	SCR1	SCR0

Index=0xF9, Byte 1. Index=0xF8 Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
SCR15	SCR14	SCR13	SCR12	SCR11	SCR10	SCR9	SCR8

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
SCR[15:0]	rw	0x0000	Scratch Pad Register

Description

Can be used for temporarily storing a previous state. Also used to verify ISA bus interface.

See Also

Register Summary (see page 43)

Example

11.53 Board ID (Index=250, RB='1')

Board Identification Register

Register Layout

Index=0xFA, Byte 0. Index=0xFA Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
BID7	BID6	BID5	BID4	BID3	BID2	BID1	BID0

Index=0xFB, Byte 1. Index=0xFA Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
BID15	BID14	BID13	BID12	BID11	BID10	BID9	BID8

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
BID[15:0]	r	-	Board ID = 0x1008. Revision 080214 (14FEB08) and Revision 080407H (07APR08) Board ID = 0x1009. Revision 090115 (15JAN09) Board ID = 0x100A. Revision 111021 (30OCT11) Board ID = 0x100B. Revision either Date code 160923 or 161023. Rev I hardware.

Description

See Also

Register Summary (see page 43)

Example

11.54 General Information (Index=252, RB='1')

General information (no longer board serial number)

Register Layout

Index=0xFC, Byte 0. Index=0xFC Word 0. RB='1'.

D7	D6	D5	D4	D3	D2	D1	D0
FPGA_TARGET3	FPGA_TARGET2	FPGA_TARGET1	FPGA_TARGET0	CPLD_TARGET3	CPLD_TARGET2	CPLD_TARGET1	CPLD_TARGET0

Index=0xFD, Byte 1. Index=0xFC Word 0. RB='1'.

D15	D14	D13	D12	D11	D10	D9	D8
0	0	0	0	0	0	0	0

Index=0xFE, Byte 2. Index=0xFE Word 1. RB='1'.

D23	D22	D21	D20	D19	D18	D17	D16
0	0	0	0	0	0	0	0

Index=0xFF, Byte 3. Index=0xFE Word 1. RB='1'.

D31	D30	D29	D28	D27	D26	D25	D24
0	0	0	0	0	0	0	0

Bit Definitions

NAME	DIRECTION	DEFAULT	DESCRIPTION
FPGA_TARGET[3:0]	r	-	FPGA Target. Both FPGA_TARGET and CPLD_TARGET nibbles must match.
CPLD_TARGET[3:0]	r	-	CPLD Target. Both FPGA_TARGET and CPLD_TARGET nibbles must match.

Description

Board CPLD and FPGA Target information.

See Also

Register Summary (see page 43)

Example

12 Power Supply

PC/104 Connector

Minimum input Power Requirements at the PC/104 Connector: +5V at 250mA typical (300mA maximum).

Power Available at J7

J7.1: +5V at 200mA (+5V directly from the PC/104 connector).

J7.15: -5V at 25mA

The +5V at J7 pin 1 is tied directly to the PC/104 +5V supply line.

The -5V at J7 pin 15 is derived from the on-board DC-DC converter and on-board regulator.

Recommend that the -5V be returned via one of the AGND pins at J7.

Recommend that the +5V be returned via the DGND for larger currents. If +5V is going to drive a small analog load (<50mA) then it is OK to return via AGND pins at J7.

13 Interrupt Summary

Description

CLASSIC INTERRUPT SOURCE SELECTION (When Interrupt Configuration Register Bit nINT_FF='0')

Note: DMA is not available on Revision 161023.

AIE or EIS	FIE	DMA	M1	Interrupt Function
0	X	1	0	No interrupt generated. Poll INT to determine when a DMA terminal count is received from the DMA controller to indicate completion of the DMA transfer.
1	X	0	0	Interrupt generated when an ADC conversion has completed. Write to the Clear Interrupt Register to clear the interrupt. In DAS1602 compatibility mode (jumper M0 installed), an interrupt is generated for each sample when in ADC-burst mode.
1	X	1	0	Interrupt generated when a DMA terminal count is received from the DMA controller to indicate completion of the DMA transfer.
0	0	0	1	No interrupt generated. Poll INT to determine when an ADC conversion has completed. Poll INT_FF to determine when 512 additional samples have been queued to the FIFO. The number of blocks is now configurable, refer to the Interrupt Configuration (see page 90) Register.
0	X	1	1	No interrupt generated. Poll INT to determine when a DMA terminal count is received from the DMA controller to indicate completion of the DMA transfer. Poll INT_FF to determine when 512 additional samples have been queued to the FIFO. The number of blocks is now configurable, refer to the Interrupt Configuration (see page 90) Register.
1	X	1	1	Interrupt generated when a DMA terminal count is received from the DMA controller to indicate completion of the DMA transfer.
0	1	0	1	Interrupt generated when 512 additional samples are deposited in the FIFO. Write to the Clear Interrupt Register to clear the interrupt. The number of blocks required to generate an interrupt is now configurable, refer to the Interrupt Configuration (see page 90) Register. Poll INT to determine when deglitched DIN0 rising-edge has occurred. By connecting CT_OUT0 or CT_OUT2 to DIN0 you can create a polled timing function. Any external input can produce a polled rising-edge. If GCTRL bit is set then this can be used to detect the beginning of pacer clock gating (i.e. start of one or more samples). Writing to the Clear Interrupt Register will also clear the INT bit.
1	0	0	1	Interrupt generated when an ADC conversion has completed. Write to the Clear Interrupt Register to clear the interrupt. In DAS1602 compatibility mode (jumper M0 installed), an interrupt is generated for each ADC-burst completion when in ADC-burst mode. Poll INT_FF to determine when 512 additional samples have been queued to the FIFO. The number of blocks is now configurable, refer to the Interrupt Configuration (see page 90) Register.

1	1	0	1	Interrupt generated when 512 samples deposited in the FIFO. Write to the Clear Interrupt Register to clear the interrupt. The number of blocks required to generate an interrupt is now configurable, refer to the Interrupt Configuration (see page 90) Register. Interrupt generated when deglitched DIN0 rising-edge has occurred. By connecting CT_OUT0 or CT_OUT2 to DIN0 you can create an interrupt timing function. Any external input can produce an interrupt. If GCTRL bit is set, this can be used to detect the beginning of pacer clock gating (i.e. start of a group of samples). Writing to the Clear Interrupt Register will also clear the INT bit.

ADDITIONAL INTERRUPT CONFIGURATIONS AND SOURCES

Please reference the following registers listed for detailed information. In summary, additional IRQ selection, IRQ sources along with programmable interrupt threshold count is possible.

Interrupt Source Select Register (see page 89)

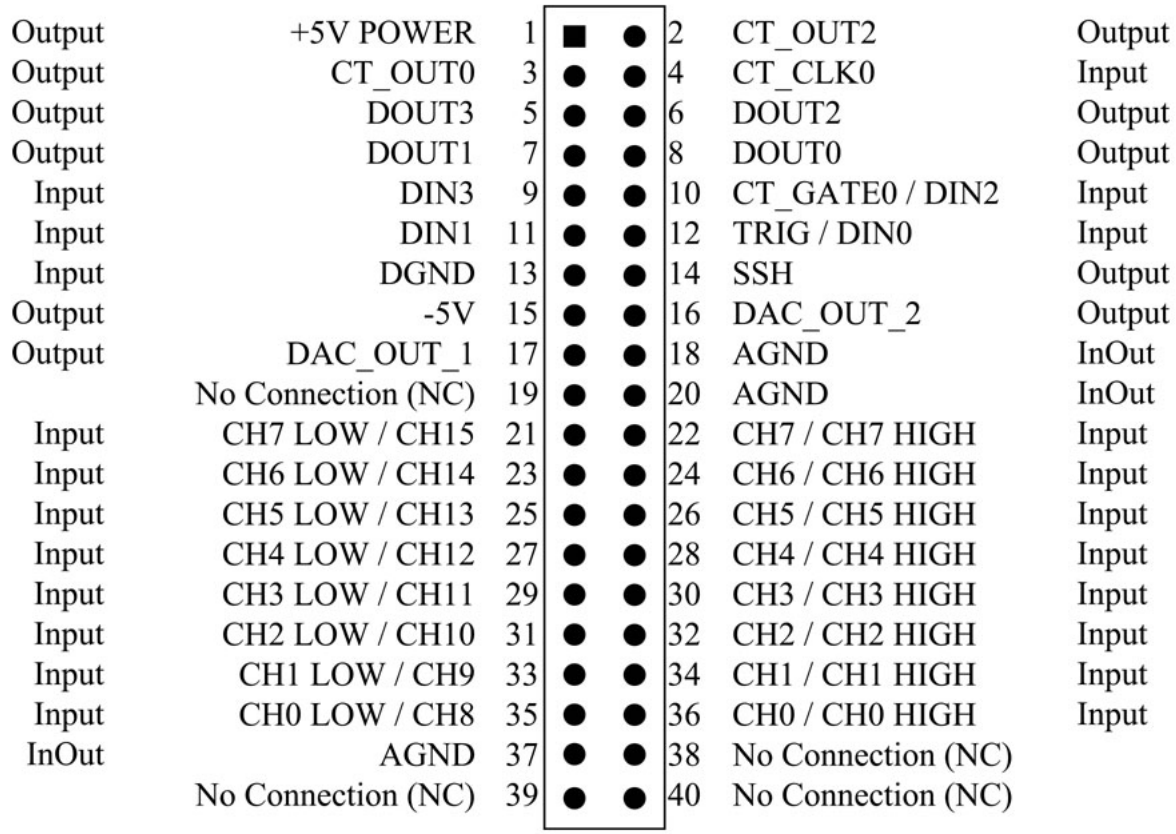
Interrupt Configuration Register (see page 90)

Interrupt Threshold Register (see page 92)

14 Connector Summary

Description

STX104 I/O Connector J7



Name	Notes
CT_OUT0	
CT_CLK0	
CT_OUT2	
TRIG/DIN0	
DIN1	
CT_GATE0 / DIN2	

DIN3	
DOUT0	
DOUT1	
DOUT2	
DOUT3	
DAC_OUT_1	
DAC_OUT_2	
CHn LOW / CHn	
CHn HIGH / CHn	
+5V POWER	200mA maximum
-5V	25mA maximum
CHn	
DGND	
AGND	
SSH	See ADC Channel (see page 54)Register for details

UNIPOLAR OR BIPOLAR ANALOG INPUT (J9)

Jumper	Function
0*	Bipolar Analog Inputs (+/- values accepted)
1	Unipolar Analog Inputs (+ values only)

* Factory Default

Note: 1 = Jumper installed, 0 = Jumper not installed.

14

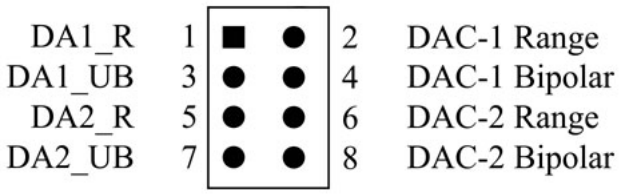
DIFFERENTIAL OR SINGLE-ENDED ANALOG INPUT (J8)

Jumper	Function
0*	Differential (see page 19) Inputs, 8-channels
1	Single-Ended Inputs, 16-channels

* Factory Default

Note: 1 = Jumper installed, 0 = Jumper not installed.

DAC RANGE SETTINGS (J5)



DA1_UB	DA1_R	DAC-1 RANGE
0	0	0 to +5 Volts *
0	1	0 to +10 Volts
1	0	-5 to +5 Volts
1	1	-10 to 10 Volts

* Factory Default

Note: 1 = Jumper installed, 0 = Jumper not installed.

DA2_UB	DA2_R	DAC-2 RANGE
0	0	0 to +5 Volts *
0	1	0 to +10 Volts
1	0	-5 to +5 Volts
1	1	-10 to 10 Volts

* Factory Default

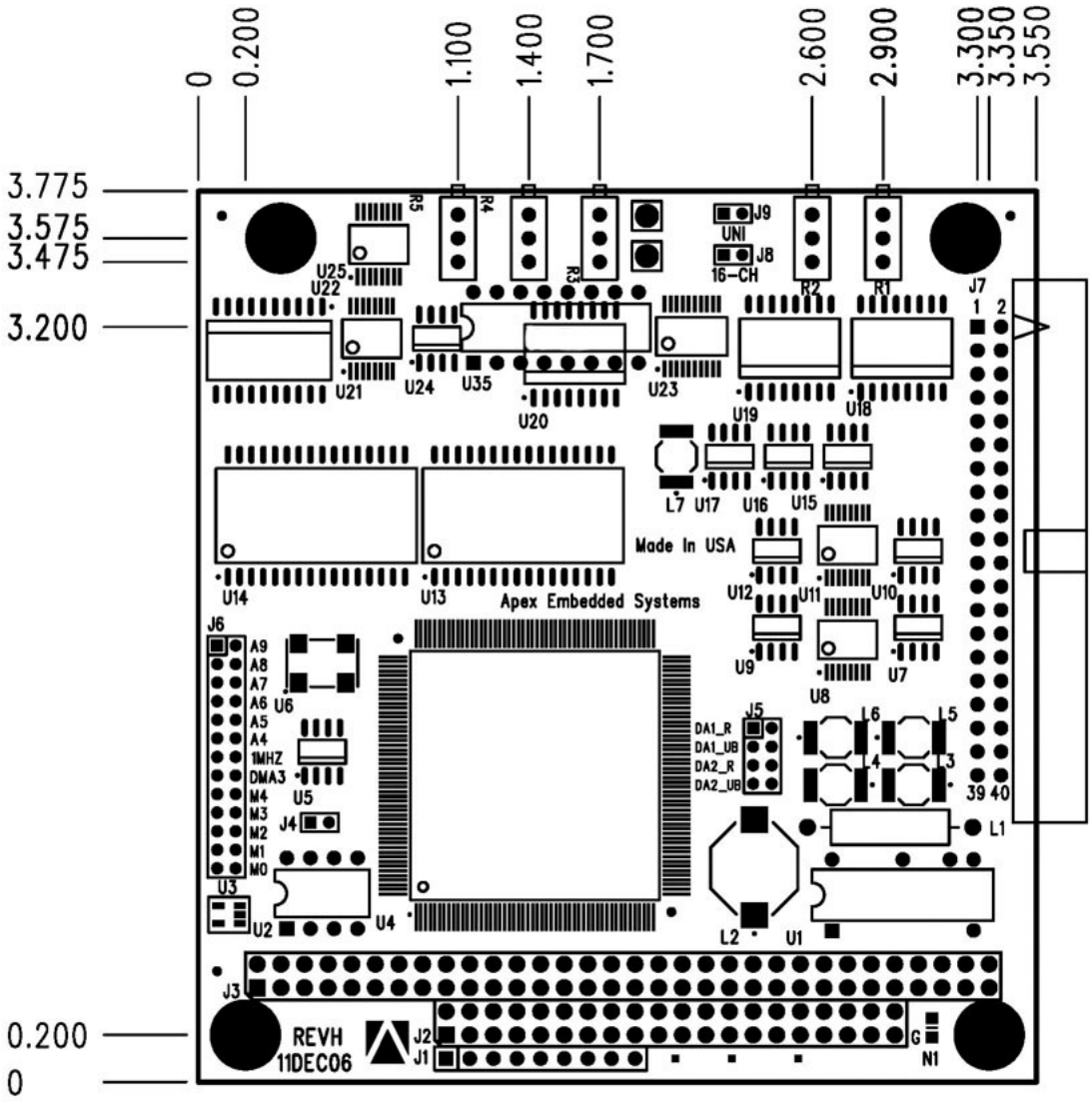
Note: 1 = Jumper installed, 0 = Jumper not installed.

15 Mechanicals

Description

The table below lists the locations of key mechanicals of the STX104 board.

LOCATION	X (milli-inches or mils)	Y (milli-inches or mils)
Lower Left Mounting Hole	200	200
Lower Right Mounting Hole	3350	200
Upper Left Mounting Hole	350	3575
Upper Right Mounting Hole	3250	3575
J7 pin 1	3300	3200



16 Support Policy

16.1 General Support Policy

We support all hardware products for a period of 3 months from time of delivery assisting in the integration process. See limited warranty terms.

16.2 Recommended Sequence in Obtaining Customer Support

Review user manuals for additional information not found in demo software.

Go to our website at www.apexembeddedsystems.com.

Contact us via email at customer.service@apexembeddedsystems.com. Technical support related inquiry answered typically within a 24-hour period.

16.3 Need Custom Modifications?

Contact us for customization or modifications to our standard product. If you need large quantities we can generally save you money through optimizing card designs to meet your exact needs.

17 Specifications

Common specifications across all STX104 part numbers

General	Operating temperature range:	-40 to +85C
		-55 to +125C
	Storage temperature range:	0 to 95% non-condensing
	Humidity:	5.0 VDC ± 10%, less than 350mA typical
	Power supply:	PC/104 8/16-bit bus. Supports either 8-bit or 16-bit data connections.
	Interface:	PC/104 form factor
	Mechanical dimensions:	

Analog I/O		
	Maximum aggregate sample rate:	200,000 Samples per second
	Conversion time:	5 uS minimum, adjustable to 25 nS using the Analog Input Frame Timer (see page 104) (AIFT)
	Sampling sources:	32-bit sample/frame timer, interrupt threshold counter, rising/falling edges of any digital input, and all legacy modes including 8254 counter, external source (DI0/TRIG) or software polled. Note: interrupt threshold counter can be used (independent of any interrupts) to pre-scale sampling source.
	Data transfer:	8- or 16-bit transfers. Single (one at a time) or Burst readout.
	Resolution:	16-bits (1/65536 of full scale), no missing codes guaranteed
	Number of channels:	8 differential or 16 single-ended Inputs.
	Input type:	Bipolar or Unipolar
	Input ranges:	Bipolar: ±10V, ±5V, ±2.5V, ±1.25V Unipolar: 0 to 10V, 0 to 5V, 0 to 2.5V, 0 to 1.25V
	Input sensitivity:	19 microvolts
	Input bias current:	50nA maximum
	Input impedance:	Differential (see page 19): 20MO min. resistance in parallel with 47pF. Single-Ended: 20MO min. resistance in parallel with 27pF.
	Common mode voltage range:	±10V
	Absolute maximum input voltage:	±35V
	DC drift or zero drift:	±2ppm/°C, ±1.5ppm/°C typical
	Gain drift:	±7ppm/°C
	Accuracy:	0.003% of reading, ±1 LSB
	Integral linearity error:	±1.5 LSB (±3 LSB on 1.25V range)

Differential (see page 19) linearity:	±1 LSB
Noise characteristics:	Gaussian behavior with maximum peak-to-peak internal noise of less than 1.5-LSB RMS over all input ranges and operating temperatures (1.2-LSB RMS typical). Jumper selectable 16-point moving average filter drops noise to less than 1-LSB RMS over all input ranges and operating temperatures (0.6-LSB RMS typical)
Common mode rejection ratio:	70dB at 60Hz

Analog Input Triggering

Start event sources:	none, software, analog input sample/frame timer, 8254 counter outputs, any digital input falling or rising edge.
Stop event sources:	none, software, analog input sample/frame maximum count, analog input sample/frame timer, 8254 counter outputs, any digital input falling or rising edge.
Synchronization event sources:	none, software, analog input sample/frame timer, 8254 counter outputs, any digital input falling or rising edge. Example: 50/60Hz line synchronization.
Start delay:	0 nanoseconds to 54 seconds, 25 nanosecond resolution
Trigger sequencing:	(1) Start ? Sample, (2) Start ? Delay ? Sample, (3) Start ? Sync ? Delay ? Sample.

Counter / Timers

Analog input pacer timer:	32-bit Analog Input Frame Timer (see page 104) (AIFT) or legacy 8254 (see page 77) 32-bit down counter (82C54 counters cascaded)
Clock source:	8254: legacy 1 MHz or 10 MHz, jumper selectable AIFT: 40 MHz (25 nS)
General purpose:	8254: 16-bit down counter

Digital I/O

Digital inputs:	4
Compatibility:	TTL and LVTTL
Logic input low maximum (Vil):	0.8 volts
Logic input high minimum (Vih):	2.0 volts
Range:	0.0 to 5.0 volts (5 volt tolerant)
Input current:	Vil: 500uA max (due to 10K pull-up to +5V) Vih: 300uA max (due to 10K pull-up to +5V)
Digital outputs:	4

Compatibility:	TTL and LVTTL
Logic output low maximum (Vol):	0.5 volts @ 12 mA
Logic output high minimum (Voh):	2.4 volts @ -12 mA
Range:	0 to 3.3 volts (directly from FPGA, not 5V tolerant)

NOTES

1. Least Significant Bit (LSB)
2. DMA is not available from revision 160923 forward. We have been deprecating DMA since year 2003 as it is 8-bit transfers and 1/4 the overall throughput compared to 16-bit transfers.

17.1 STX104-1MFIFO-NODACS

MTBF Information

MIL-HDBK-217F (Ground benign) at 25^{o}C	Failure Rate = 1.4829 MTBF = 674,355.56 hours
MIL-HDBK-217F (Ground benign) at 40^{o}C	Failure Rate = 2.376 MTBF = 420,814.34 hours
Telcordia SR-332 (Ground, Fixed, Controlled) at 25^{o}C	FR(90) = 1535.5061 MTBF = 651,251.06 hours
Telcordia SR-332 (Ground, Fixed, Controlled) at 40^{o}C	FR(90) = 2018.73 MTBF = 495,360.56 hours

17.2 STX104-1MFIFO-DAQ

Analog Outputs

Resolution:	16-bits (1/65536 of full scale) Monotonicity Guaranteed
Number of channels:	2

Output ranges:	±10V, ±5V, 0-5V, 0-10V Each channel independently jumper configurable
Output current:	±5 milliamps maximum per channel
Offset error:	Less than 16 LSB
Gain error:	Adjustable to 0 LSB by potentiometer
Differential (see page 19) non-linearity:	±1 LSB max
Settle time:	10 microseconds
Integral non-linearity:	±1 LSB max

MTBF Information

MIL-HDBK-217F (Ground benign) at 25^{o}C	Failure Rate = 1.504 MTBF = 664,780.25 hours
MIL-HDBK-217F (Ground benign) at 40^{o}C	Failure Rate = 2.396 MTBF = 417,347.25 hours
Telcordia SR-332 (Ground, Fixed, Controlled) at 25^{o}C	FR(90) = 1042.3965 MTBF = 625,114.4 hours
Telcordia SR-332 (Ground, Fixed, Controlled) at 40^{o}C	FR(90) = 2133.5 MTBF = 468,710.88 hours

17.3 Valid Part Numbers

Valid Part Number	Description
STX104-1MFIFO-NODACS	STX104 16-bit Analog I/O Module with 1M sample FIFO and dual 16-bit DACs.
STX104-1MFIFO-DAQ	STX104-ND 16-bit PC/104 Analog I/O Module with 1M sample FIFO **without** dual 16-bit DAC

17.4 Revision Information

By scanning and manipulating the STX104 registers one can determine the firmware revision that is installed.

Description

Revision Releases

Date	Revision ROM Label	Board ID Register Value
July 16, 2004	071604	Does not exist

February 14, 2008	080214H	0x1008
April 4, 2008	080407H	0x1008
January 15, 2009	090115H	0x1009
October 30, 2011	111021H	0x100A
October 23, 2016	See new revision information below	0x100B

Note:

1. Revision 080214H has been replaced by 080407H including boards in the field effectively eliminating the 080214H revision entirely. Revision 080407H corrects for several minor issues that were discovered in revision 080214H which does not warrant a change to the board ID.

2. Revision 090115H added mapping the 8254 registers through the index register set to improve accessibility by driver software.

3. For all revisions, October 23, 2016 and beyond, the register remains at 0x100B for legacy software compatibility. We have added additional registers to retrieve FPGA date code.

NEW REVISION INFORMATION

Date	PCB Date Code	CPLD Date Code	FPGA Date Code	Board ID Register Value	Description
November 2, 2016	160923	161101	161102	0x100B	Original release of firmware
December 21, 2016	160923	161221	161221	0x100B	Recompiled revision
March 7, 2017	161023	161219	170307	0x100B	FPGA has counter timer read back fixed
March 7, 2017	161023	161219	170307	0x100B	FPGA has counter timer read back fixed
November 15, 2017	161023	161219	171115	0x100B	FPGA has index pointer read back issue fixed and also has valid LED blink fixed

Notes:

Board ID information can be found here: Board ID (Index=250, RB='1') (see page 114).

Example

```
/* STX104 Revision Information */
#define STX104_REVISION_071604                0
#define STX104_REVISION_080214             0x1008
#define STX104_REVISION_080407             0x1008
#define STX104_REVISION_090115             0x1009
#define STX104_REVISION_111021             0x100A
#define STX104_REVISION_161023             0x100B
/*************************************************************
/                                          REVISION DETECTION
*/
static unsigned int STX104_Revision_Detected( int board )
```

```
{
    unsigned int value;

    value = 0x55AA;
    STX104_Set_Bank( board, 1 );
    STX104_Write_Indexed_Data_Word( board, STX104_SCRATCH_PAD, value );
    value = STX104_Read_Indexed_Data_Word( board, STX104_SCRATCH_PAD );

    if ( value == 0x55AA ) value = STX104_Read_Indexed_Data_Word( board, STX104_BOARD_ID );
    else  value = 0;

    return( value );
}
```

18 LImited Warranty

Unless altered by written agreement, Apex Embedded Systems LLC (APEX) warrants to the original purchaser for a period of one year from the date of original purchase, that the products shall be free from defects in material and workmanship. APEX's obligation under this warranty is limited to replacing or repairing, at its option and its designated site, any products (except consumables) within the warranty period that are returned to APEX in the original shipping container(s) with an APEX RMA number referenced on the shipping documents.

This warranty will not apply to products that have been misused, abused, or altered. This warranty will not apply to prototypes of any kind, engineering services, software or products under pre-release status. Any returns must be

supported by a Return Material Authorization (RMA) number issued by APEX. APEX reserves the right to refuse delivery of any shipment containing any shipping carton which does not have an RMA number displayed on the outside.

Purchaser shall prepay transportation to APEX's location. If returned parts or products are repaired under the terms of this warranty, APEX will pay return transportation charges. Allow six (6) to eight (8) weeks for warranty repairs.

Index

F

G

H

I

L

M

N

P

R

S

T

V

W