Stanford Linear Accelerator Center

Accelerator Controls Department

Electronic Systems Engineering

Hytec IP-ADC-8413
Data Averager Modification

Register Description

Version 3.0

Author: John Dusatko

July 31, 2007

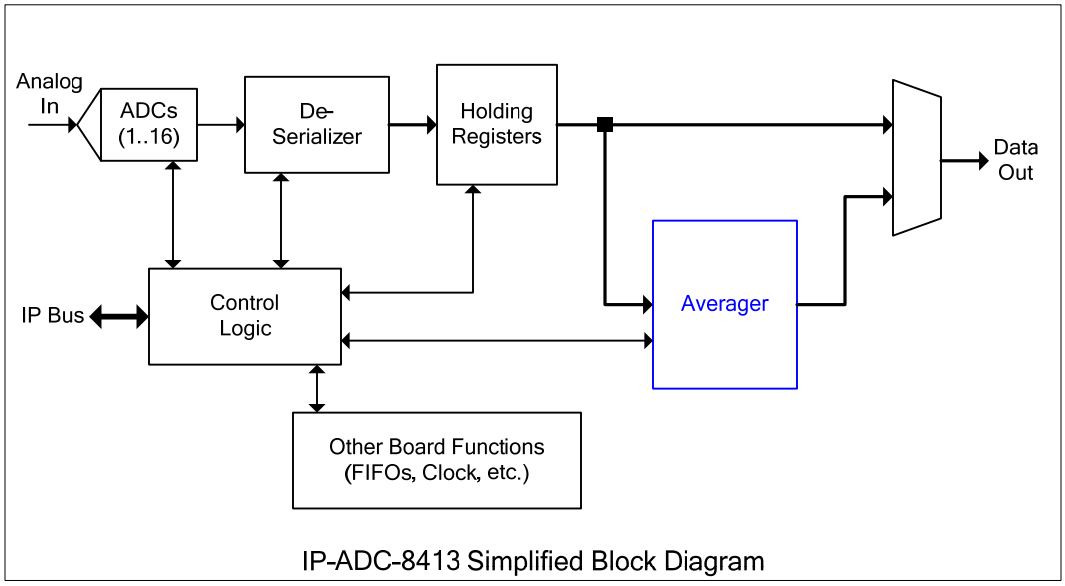
Hytec IP-ADC-8413
Data Averager Register Description

Introduction:

This document describes the registers and sequence of operations required to support the added Data Averager feature of the Hytec model IP-ADC-8413 16-bit, 16-channel Industry Pack Analog to Digital Converter (ADC) module. It gives a brief overview of the averager function and provides descriptions of the modified and/or added board registers that control this feature and read out its results. Finally the sequence of register operations is given to guide the programmer in developing code. Note that in this document all additions and changes having to do with the Data Averager are highlighted in BLUE.

Description of the averager:

In order to provide a rudimentary level of signal processing with the purpose of improving signal-to-noise performance while reducing random noise, a data averaging function was added to the IP-ADC-8413. This is a gateware modification to the board's FPGA. A simplified block diagram of the board is shown below.



IP-ADC-8413 Simplified Block Diagram

This block diagram shows the overall board and where the data averager fits into it. The averager, as it is currently designed, performs the following operation:

$$V(average) = \frac{1}{64}\sum_{n=1}^{64} V(sample)$$

Where V(average) is the averaged ADC voltage value and V(sample) is the raw ADC sample value. In the above relation, 64 successive ADC values are summed. This sum is then divided by 64 to give the average value. The number of averaged samples, 64, gives a coherent averaging gain of 18dB. The averager, outlined in blue in the block diagram, implements the above equation. It operates on all sixteen ADC channels, with the sum being accumulated with each ADC sample. Note that the accumulation rate is equal to the ADC sampling rate, which is set by either an external clock or the internal clock. The internal clock is controlled by register 0x06, the Internal Clock Rate Register. In order to provide 60Hz rejection, this clock must be set to acquire 64 samples over a period of 1/60th

of a second. This corresponds to a 3.8KHz sample rate. The board's hardware and FPGA gateware has been modified to support this rate.

Averager Operation:

The averager functions as follows: In looking at the block diagram, the sixteen ADCs output a serial datastream that is input to the FPGA. The FPGA contains a deserializer that converts the sixteen 16-bit serial data streams into sixteen two's compliment format data words. These values are stored in holding registers. Normally, the data in the holding registers can be read out directly (by reading the ADC Data Registers) and/or be loaded into the external and internal FIFOs. The data averager unit is connected to these holding registers.

After the ADCs have acquired a sample, the data is read into the de-serializer and then stored into the FIFOs. When this is finished, the averager block then reads out the holding registers again and adds each channel's data value with an accumulated sum that has been previously stored in the averager's internal memory. The averager has internal counters to keep track of how many samples have been stored and which channel is which. The averager's internal memory can be thought of as having sixteen separate slots where the accumulated sums for each channel are stored. When all 64 samples have been accumulated for each channel, the averager's internal control logic performs a divide operation for each channel to arrive at the averaged value.

The averager itself is divided into three functional sections: the master controller, the accumulator and the divider. Each subsection is controlled by a state machine. The master controller interfaces to the rest of the IP-ADC-8413 logic. It is controlled by bits set in the auxiliary control register. The whole design is contained in two VHDL files which are dropped into the main FPGA design.

Note that the averager operates sequentially on all sixteen channels. Therefore a necessary time delay is incurred. The averager logic runs at the IP clock rate (8MHz or 32MHz) so ample margin is available to complete all the necessary operations.

There are two modes that can be used to read out the averaged data:

Polling Mode:

When all 16 channels have been completed, the averager signals the computer that new data values are ready by setting a done bit in the auxiliary control register. It is the responsibility of the control software to poll this bit to know when the averaged data is ready to be read out. In addition, a handshake mechanism exists that holds off the collection and averaging of new ADC samples until the current averages are read out.

The 8413 is placed into acquisition mode and the averager is enabled. A status bit in the ACR is polled to tell when averaged data is available. When it is, averaging is halted and the results are ready for readout. After the data has been read out, and ACR bit it toggled to resume averaging and the status bit is again polled to indicated when new data is ready for readout. This sequence of operations repeats for each set of averaged data.

SAM Readout Mode:

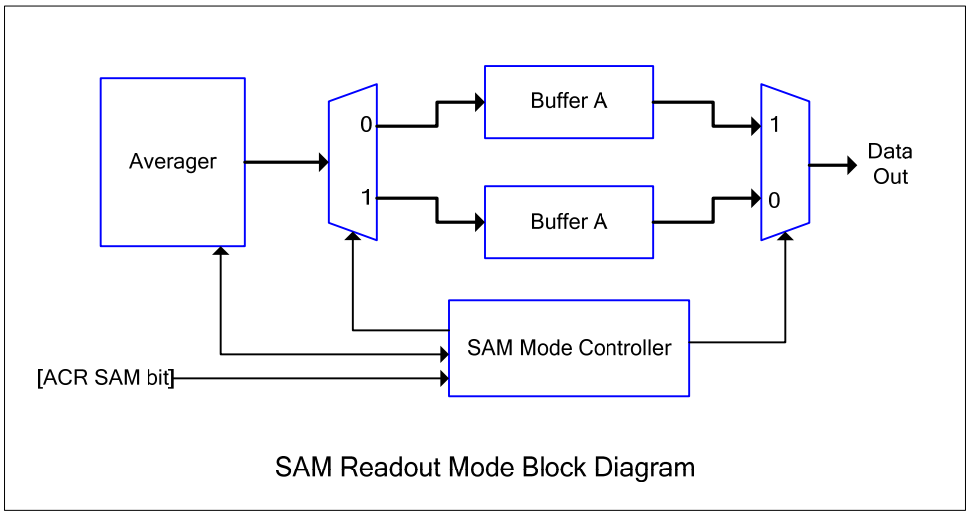
This mode emulates the data readout mode of the SLAC VME Smart Analog Monitor (VSAM) module. In this mode, the averaged data results are automatically written and made available for external readout without the need for handshake/polling bits. This is done using a simple double-buffer configuration with one buffer containing the latest averaged values for readout while the other buffer accumulates the next averaged values. When accumulation of the new averages is complete, the buffers are automatically swapped so that the new results can be read out, while the previous values get over written with the latest accumulated samples to be averaged. This sort of ping-pong action occurs automatically ensuring that the most recent data is available to the CPU when it reads out the ADC module. The eliminates the need to have polling bits or an interrupt mechanism (not implemented here) to indicate when data is ready. This is done with the slight risk of missing data either too early (stale data) or too late (missed group of averages) depending on how fast the CPU can read out the ADC module. For most applications, it

is assumed that either the CPU is fast enough and/or missing a set of averages will not have a negative effect. When SAM mode is disabled, only one buffer is used for readout; which is transparent to the user.

SAM readout mode is enabled by setting the Auxillary Control Register (ACR) SAM bit = 1. Note that SAM Readout Mode is only available when the data averager feature is enabled and does not apply to normal (non-averaged) data collection.

The SAM readout controller incorporates an arbitration mechanism that holds off the buffer swap if a VME (IP bus) readout is in progress. Once the readout is complete, the swap occurs. The holdoff method is simply a time delay that prevents the swap from happening for a fixed duration. As soon as an external access is detected, a swap inhibit is asserted and a delay counter is started. When the delay counter is exhausted, the swap inhibit is de-asserted, and the swap occurs. The length of the delay is equal to the calculated time it takes for a series of 18 consecutive IP bus read cycles to occur plus a safety margin. This equates to 256 counts of the IP bus clock which is equal to a 25us delay for an 8MHz clock and 6us for a 32MHz clock.

Note that when in SAM mode, writes to the ADN bit, which is required for polling mode are not needed and have no effect.



SAM Readout Mode Block Diagram

Detailed Description:

The SAM readout controller is simply a state machine and support logic that implements an arbiter and control mechanism for selecting which region of the Averager's dual-port RAM memory to read out, and which to have the averager accumulate its results into. The VME readout has precedence. This module interfaces to the averager's master state machine and dual-port ram. The module functions as follows: when enabled via the SAM Mode Enable (SAM) bit and the averager enable (AEN) bit, the averager selects buffer A for accumulation and buffer B for readout. The controller waits for the averager to signal that new avg'd data is ready. The SAM contoller then checks to see if a VME access is in progress to buffer B. If there is, it waits until that access is done before proceeding to swap the buffers and signal the Averager that it can proceed with a new averaging cycle. The SAM controller state machine waits for the Averager's state machine to signal that it has started a new avg cycle (by de-asserting the av_drdy signal) before it de-asserts the drd_done_s signal, thus completing the handshake. This ping-pong action continues as long as the averager and the SAM mode controller are enabled. Note that there is a selection mux in the Averager that selects the source of the Averager's Master State Machine drd_done signal from

either the Aux Ctrl Reg ACSR bit or the Sam Mode Controller (which is driven by the AND of the sam_mode and avgr mode enable signals). In addition, when SAM mode is disabled, the data buffers select only buffer A for both sides of the averager so that it can operate in polling mode. The VME active signal is generated by the logical OR of the ADC channel read bus strobe bits. This OR'd signal feeds a counter that generates a hold off gate signal for a fixed period of time. This period of time has been chosen to equal the total amount of time it would take to readout out all 18 channels using standard VME read cycles, which are estimated to take approximately 256 IP bus clock counts.

Registers Changed in SLAC version of IP-ADC-8413:

Three main sets of registers that have been changed. These are the eighteen ADC data registers, the Auxillary Control Register (ACR) and the Internal ADC Clock Rate register. These changes are illustrated in the following sections.

ADC Data Registers:

In normal operating mode, these registers contain the most recently converted ADC value for all sixteen input channels. When the aux control register's *ADC Readout Select* (ARS) bit is set to a logic one, these registers contain the averaged data values. These values are valid when the aux control register's AVG_DONE bit is set high. Note that in both cases, the data format can be selected to be either straight binary or two's compliment depending upon the value of the 2C bit in the Auxiliary Control Register.

Normal Mode:

Read Address: [0x10 – 0x32]

The first sixteen ADC buffer registers store the last sample conversions and may be read at any time. The seventeenth is used to monitor the 0V reference and the eighteenth is used to monitor the 2.5V reference.

Register Address (Hex)	Description	Register Address (Hex)	Description
10	ADC 0	22	ADC 9
12	ADC 1	24	ADC 10
14	ADC 2	26	ADC 11
16	ADC 3	28	ADC 12
18	ADC 4	2A	ADC13
1A	ADC 5	2C	ADC 14
1C	ADC 6	2E	ADC 15
1E	ADC 7	30	0V Reference
20	ADC 8	32	2.5V Reference

Average Mode:

Read Address: [0x10 – 0x32]

The first sixteen ADC buffer registers contain the respective ADC channel's averaged data value. The seventeenth and eighteenth channels are not defined in this mode.

Register Address (Hex)	Description	Register Address (Hex)	Description
10	ADC 0 Average	22	ADC 9 Average
12	ADC 1 Average	24	ADC 10 Average
14	ADC 2 Average	26	ADC 11 Average
16	ADC 3 Average	28	ADC 12 Average
18	ADC 4 Average	2A	ADC13 Average
1A	ADC 5 Average	2C	ADC 14 Average
1C	ADC 6 Average	2E	ADC 15 Average
1E	ADC 7 Average	30	xxxx
20	ADC 8 Average	32	xxxx

Auxiliary Control Register (ACR):

This register is reset to all zeros at power-up and must be set for correct operation

Control
Read/write Address: 0x34

D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
X	X	X	X	**SAM**	**AINI**	**ARS**	**A/B**	**AEN**	**PG2**	**PG1**	**PG0**	**2C**	**N/S**	**X**	**RGE**

The following bits are described in the Hytec IP-ADC-8413 User manual:

Bit 0 **RGE** Sets the range of the ADCs. 0 - +/- 10V and 1 - +/-5V.
Bit 2 **N/S** 0 – Standby Mode, 1 – Normal Operation.
Bit 3 **2C** 0 – ADC Values in 2's Complement (default), 1 – ADC Values 0000 (Neg FS)-FFFF (Pos FS).
Bit 4 **PG0** Bit 0 of ID PROM Paging.*
Bit 5 **PG1** Bit 1 of ID PROM Paging.*
Bit 6 **PG2** Bit 2 of ID PROM Paging.*

The following bits have been added for the Data Averager function:

AEN Bit(s): [7]
Read/Write

Averager ENable: Enables the data averager subsystem. When enabled, 64 successive samples will be summed and divided by 64. The results are then stored in the ADC Data Registers. Note that when enabled, the averaging started as soon as the next ADC sample clock occurs as long as the ADC has been set to normal operation, is ARMed and receives the appropriate trigger.

0 = not enabled
1 = enabled

ADN/BUF (A/B) Bit(s): [8]
Note: This bit is a dual-mode type, set by the state of the SAM mode bit. The descriptions follow:

Polling Mode (SAM = 0)

ADN Bit(s): [8]
Read/Write – Special!

Averager DoNe: This bit indicates when the average operation has been completed and valid averaged data is present in the ADC Readout Registers (Regs 0x10 – 0x30). Note that this is a dual-use bit. In read mode, it indicates done status, in write mode writing a one tells the averager subsystem that its last results have been read out and that it can now resume averaging the next set of 64 samples. When SAM Readout Mode is enabled, writing to this bit has no effect.

Read:

0 = averaging not done
1 = averaging done

Write:

0 = no action
1 = Readout complete, start averaging the next set of samples / this bit must be cleared to zero

SAM Mode (SAM = 1)

BUF Bit(s): [12]

Read/Write

Readout BUFfer Status: In SAM Readout Mode, this bit indicates which of the two ping-pong buffers is ready for readout. As a convenience, changes in the state of this bit can be monitored by software to indicate when new data is available.

Read:

0 = Buffer A is available for external readout
1 = Buffer B is available for external readout

Write:

<no action>

ARS Bit(s): [9]

Read/Write

ADC Readout Select: Selects whether the ADC Readout Registers (regs 0x10 – 0x30) contain the latest digitized values or the averaged data values. This bit can be changed at any time.

0 = ADC Data Samples read out at addr 0x10 – 0x30
1= Averaged Data read out at addr 0x10 – 0x30

AINI Bit(s): [10]

Read/Write

Averager INItialize: Initializes the average subsystem: resets all state machine, zeroes out all counters, sets all averager logic to idle state.

0 = Normal operation
1 = Initialize

SAM Bit(s): [11]

Read/Write

SAM mode enable: Enables SAM readout mode: the averaged data is automatically written into an output buffer while another buffer accumulated sampled data. When a new set of averages is available, the buffers are automatically swapped.

0 = SAM mode DISabled
1 = SAM mode ENAbled

Internal ADC Sample Clock Rate:

Address offset: 0x06

This register sets the ADC sample frequency. The averager mode also includes a hardware change to the on-board crystal oscillator from 16.000MHz to 15.36MHz. This alters the divided clock rates. In addition, the clock system was modified to provide additional division ratios (3.84KHz, 7.68KHz and 15.36KHz) to provide sample rates harmonically related to the US 60Hz AC line frequency. For the 64-sample averager, select the 3.84KHz rate. The other two rates have been provided to support larger numbers of averaged samples (128 and 256, respectively). Note that these three frequencies replace the 4KHz, 8KHz and 16KHz register value selections, respectively. These are indicted in **BOLD** below. In addition, all of the other frequencies change value do to the different base clock frequency.

Original Version Clock Rate Register Frequency Table

Register Value	Clock Rate Frequency	Register Value	Clock Rate Frequency
0	1Hz	9	1KHz
1	2Hz	10	**2KHz**
2	5Hz	11	**5KHz**
3	10Hz	12	10KHz
4	20Hz	13	**20KHz**
5	50Hz	14	50KHz
6	100Hz	15	100KHz
7	200Hz	16	160KHz
8	500Hz	--	--

Averager Modification Clock Rate Register Frequency Table

Register Value	Clock Rate Frequency	Register Value	Clock Rate Frequency
0	0.960Hz	9	960.0KHz
1	1.92Hz	10	**3.84KHz**
2	4.80Hz	11	**7.68KHz**
3	9.60Hz	12	9.6KHz
4	19.20Hz	13	**15.36KHz**
5	48.0Hz	14	48.0KHz
6	96.0Hz	15	96.0KHz
7	192.0Hz	16	15.36KHz
8	480.0Hz	--	--

Clock System Modification:

This addition modifies the clock system to include clock rates the allow the ADC to run at specific harmonics of the 60Hz AC line frequency. This is required to provide rejection at 60Hz and its harmonic multiples. To take 64 samples at $1/60^{th}$ of a second, this corresponds to a 3.84KHz sampling rate. The board's clock system only offers 2KHz and 5KHz selections. A 3.84Khz selection must be added, this is accomplished by an additional FPGA gateware modification. Even with the modification, the 8413's on-board 16MHz crystal oscillator will not support this rate due to the non-integer relationship between the two numbers. Therefore, the crystal oscillator must be changed to one with a 15.36MHz clock rate. This frequency give the correct harmonic relationship to the 60Hz line frequency. Therefore, when the FPGA EEPROM is updated, the crystal oscillator will also be replaced. Two additional harmonically-related clock frequencies will also be supplied, 7.68KHz and 15.36KHz. These frequencies are meant support future different averaged sample values.

Sequence of Operations:

This section describes how the board should be set up in order to perform data averaging. Note that these operations are described in a sequential fashion and are meant to be implemented in the same way.

Format: Reg<address>_[<bit(s)>]	**=**	**<value>**	**(<comment>)**

Setup Board:

Perform normal board setup and initialize operations:

<among the other normal setup/init operations, do the following>:

Set clock source = internal

Set internal clock rate = <as required>

DO NOT set ARM = 1 yet

Averager Operation:

Polling Readout Mode:

Initialize Averager:

Set and clear ACR AINI bit:

Write:	Reg0x34_[10]	=	0x1	(initialize averager unit)
Write:	Reg0x34_[10]	=	0x0	(release reset)

Enable Averager:

Set AEN bit:

Write:	Reg0x34_[7]	=	0x1	(enable averager)

Set board to begin sampling:

Set board into Normal Operating mode

Write:	Reg0x34_[2]	=	0x1	(Set ACR N/S bit)

Set board to ARMED state

Write:	Reg0x00_[15]	=	0x1	(Set CSR ARM bit)

Wait for Averaging to complete:

Read:	Reg0x34_[8]	=	0x<X>	(Poll the ADN bit)

(When Reg0x34_[8] = 0x1, the Averaged data is ready to be read out)

Read Out the Avg Data:

Read:	Reg0x10 – Reg0x30	=	<Avg Data>

When reading is done, tell the averager:

Write:	Reg0x34_[8]	=	0x1	(Set the ADN bit)

Clear the ADN bit:

Write:	Reg0x34_[8]	=	0x0	(Clr the ADN bit)

Verify that it has re-inited:

Read:	Reg0x34_[8]	=	0x0	(ADN bit should = zero)

At this point if ARM=1 and AEN=1, then the averager will automatically start averaging the next set of samples. To stop averaging at any point, set AEN=0, always set AINI =1 then =0 to re-init the unit before getting more averaged data.

SAM Readout Mode:

Initialize Averager:

Set and clear ACR AINI bit:

Write:	Reg0x34_[10]	=	0x1	(initialize averager unit)
Write:	Reg0x34_[10]	=	0x0	(release reset)

Enable Averager:

Set AEN bit:

Write:	Reg0x34_[7]	=	0x1	(enable averager)

Enable SAM Mode:

Set SAM bit:

Write:	Reg0x34_[11]	=	0x1	(enable SAM readout mode)

Set board to begin sampling:

Set board into Normal Operating mode

Write:	Reg0x34_[2]	=	0x1	(Set ACR N/S bit)

Set board to ARMED state

Write:	Reg0x00_[15]	=	0x1	(Set CSR ARM bit)

(optional) ***Test the state of the A/B bit to see if new data is available****:*

Write:	*Reg0x34_[8]*	=	*0x1*	*(read this bit, look for state change)*

Read Out the Avg Data:

Read:	Reg0x10 – Reg0x30	=	<Avg Data>

<wait a specific amount of time>

<Repeat Read Out operation *or* Goto A/B bit state test and wait for new data indication before reading out>

At this point the module runs automatically and new data is ready for readout every 100ms.

Additional Features:

This section lists additional features that are intended to be added to later version(s) of the FPGA gateware:

Number of Averaged Samples:
The current averager is designed to average over 64 samples. This is should be adequate for our requirements. If it is not, then a higher number of samples can be used: 128, 256, 1024, etc. The tradeoff will be between the accumulation time and the system data rate requirements. It may be useful to provide a settable number of averaged samples. Further evaluation will drive this feature's implementation importance.

FIFO Usage:
It is sometimes convenient to acquire a record of averaged data into a memory instead of polling for each separate sample set. In this case, the external FIFO could be used to store the averaged data. This feature would require extensive modification of the averager and original 8413 control logic.

Change history:

Hytec IP-ADC-8413 Data Averager Modification *Register Description* Document Change History		
Version	**Date**	**Changes**
1.0	07-31-2007	-original version
2.0	09-xx-2007	- added clock system mod description
3.0	10-30-2007	- Added SAM readout mode