IP-SOFTDAC-M

16-channel 16-bit
Digital/Analog Converter
With memory
Industry Pack Module

PROGRAMMING MANUAL

819-20-000-4000

Version 1.0

September 2006

ALPHI TECHNOLOGY CORPORATION

6202 S. Maple Avenue #120

Tempe, AZ 85283 USA

Tel: (480) 838-2428

Fax: (480) 838-4477

NOTICE

The information in this document has been carefully checked and is believed to be entirely reliable. While all reasonable efforts to ensure accuracy have been taken in the preparation of this manual, ALPHI TECHNOLOGY assumes no responsibility resulting from omissions or errors in this manual, or from the use of information contain herein.

ALPHI TECHNOLOGY reserves the right to make any changes, without notice, to this or any of ALPHI TECHNOLOGY's products to improve reliability, performance, function or design.

ALPHI TECHNOLOGY does not assume any liability arising out of the application or use of any product or circuit described herein; nor does ALPHI TECHNOLOGY convey any license under its patent rights or the rights of others.

ALPHI TECHNOLOGY CORPORATION

All Rights Reserved

This document shall not be duplicated, nor its contents used
for any purpose, unless express permission has been granted in advance.

TABLE OF CONTENTS

Copyright ALPHI Technology Corp, 2006

1 GENERAL DESCRIPTION

1.1 INTRODUCTION

The **IP-SOFTDAC-M** is a high performance DIGITAL TO ANALOG module. The **IP-SOFTDAC-M** outputs 16 channels with a 16-bit resolution at a maximum settling time of 2 μS.

The primary features of the **IP-SOFTDAC-M** are as follows:

- 2 μSecond settling time (0 to 5 V)
- Six Programmable Output Ranges per channel
- Unipolar: 0V to 5V, 0V to 10V
- Bipolar Mode: ±5V, ±10V, ±2.5V,–2.5V to 7.5V
- 1LSB Max DNL and INL Over the Industrial Temperature Range
- Glitch Impulse < 2nV-s
- 16-Lead SSOP Package
- Power-On Reset to 0V
- Local 8kx8 Flash EPROM to store local user information
- Two stage buffers
- Global output buffer w/ internal or external triggering
- 64Kbytes memory for waveform generation

1.2 FUNCTIONAL DESCRIPTION

The IP-SOFTDAC-M uses 16 Linear LTC 1592 D/A converters.

The Linear LTC1592 are serial input 16-bit multiplying current output DACs that operates from a single 5 Volt supply. These SoftSpan DACs can be software-programmed for either uni-polar or bi-polar mode through a 3-wire SPI interface. In either mode, the voltage output range can also be software-programmed. Two output ranges in uni-polar mode and four output ranges in bi-polar mode are available.

The DACs are accurate to 1LSB over the industrial temperature range in both uni-polar and bi-polar modes. True 16-bit 4-quadrant multiplication is achieved with on-chip four quadrant multiplication resistors.

These devices include an internal deglitcher circuit that reduces the glitch impulse to less than 2nV-s (typ).

 Copyright ALPHI Technology Corp, 2006

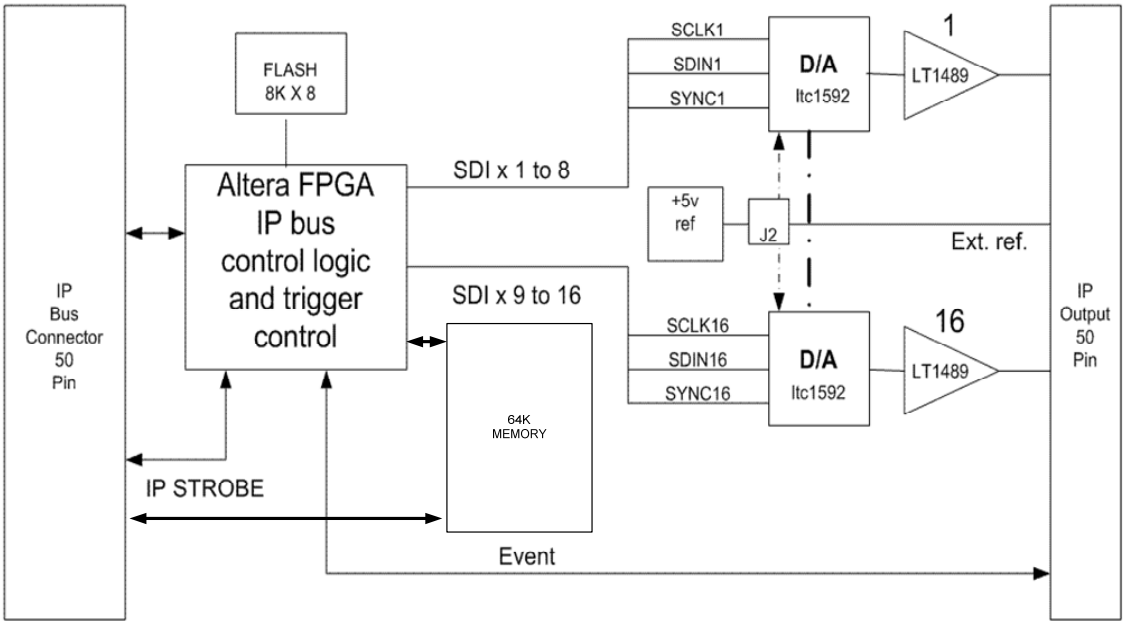

Figure 1.1: Block Diagram

 Copyright ALPHI Technology Corp, 2006

2 INTERNAL ORGANIZATION

2.1 IP INTERFACE

2.1.1 IDSPACE

The on-board logic provides information about the module to the user. The lower address contains data related to the type of module, revision, etc...

ID space address	Description	Value
0x01	ASCII “I”	0x49
0x03	ASCII “P”	0x50
0x05	ASCII “A”	0x41
0x07	ASCII “H” (32 MHz)	0x48
0x09	Manufacturer identification	0x11
0x0B	Module type	0x23
0x0D	Revision module	0x0A
0x0F	Reserved	

Table 2-1 IDSPACE content 32MHz IP

ID space address	Description	Value
0x01	ASCII “I”	0x49
0x03	ASCII “P”	0x50
0x05	ASCII “A”	0x41
0x07	ASCII “H” (8 MHz)	0x43
0x09	Manufacturer identification	0x11
0x0B	Module type	0x23
0x0D	Revision module	0x0A
0x0F	Reserved	

Table 2-2 IDSPACE content 8MHz IP

2.1.2 IOSPACE

The **IP-SOFTDAC-M** module use 16 Linear Technology LTC1592 D/A converter.

A double buffered interface is use to transfer incoming data to the output.

NAME	ADDR	DATA	R/W	COMMENTS
INT SAMP CLK	0x00	DW	R/W	Divisor for Internal Sampling Clock
SM ADDRESS	0x04	DW	RO	Current Address for the State Machine

 Copyright ALPHI Technology Corp, 2006

LAST ADDR 0	0x08	DW	R/W	Last Address with Valid Data, Bank 0
LAST ADDR 1	0x0C	DW	R/W	Last Address with Valid Data, Bank 1
BANK 0 CTRL	0x10	DWB	R/W	Controls Bank 0
BANK 1 CTRL	0x11	DWB	R/W	Controls Bank 1
CTRL/STAT 0	0x12	DWB	R/W	General control and status
CTRL/STAT 1	0x13	DWB	R/W	General control and status
RESET SAMP CLK	0x14	W	WS	Reset sampling clock counter
RESET ADDRESS	0x16	W	WS	Reset address counter
RESET DACS	0x18	W	WS	Reset all DACs
UPDATE DACS	0x1a	W	WS	Update all DACs
SWITCH BANKS	0x1c	W	WS	Switch active bank
DAC01	0x20	DW	W	Direct DAC output
DAC02	0x22	DW	W	Direct DAC output
DAC03	0x24	DW	W	Direct DAC output
DAC04	0x26	DW	W	Direct DAC output
DAC05	0x28	DW	W	Direct DAC output
DAC06	0x2a	DW	W	Direct DAC output
DAC07	0x2c	DW	W	Direct DAC output
DAC08	0x2e	DW	W	Direct DAC output
DAC09	0x30	DW	W	Direct DAC output
DAC10	0x32	DW	W	Direct DAC output
DAC11	0x34	DW	W	Direct DAC output
DAC12	0x36	DW	W	Direct DAC output
DAC13	0x38	DW	W	Direct DAC output
DAC14	0x3a	DW	W	Direct DAC output
DAC15	0x3c	DW	W	Direct DAC output
DAC16	0x3e	DW	W	Direct DAC output
Trigger	0x40			Writing to this address generates an internal trigger
Control Register	0x44			Trigger configuration. Specifies which trigger to use, and if needed, where to output an external trigger
Command Register	0x48			Command sent to the DAC in the next access

 Copyright ALPHI Technology Corp, 2006

Offset	Register Name	
0x00	Ch # 0	D/A # 0 data register
0x02	Ch # 1	D/A # 1 data register
0x04	Ch # 2	D/A # 2 data register
0x06	Ch # 3	D/A # 3 data register
0x08	Ch # 4	D/A # 4 data register
0x0A	Ch # 5	D/A # 5 data register
0x0C	Ch # 6	D/A # 6 data register
0x0E	Ch # 7	D/A # 7 data register
0x10	Ch # 8	D/A # 8 data register
0x12	Ch # 9	D/A # 9 data register
0x14	Ch # 10	D/A # 10 data register
0x16	Ch # 11	D/A # 11 data register
0x18	Ch # 12	D/A # 12 data register
0x1A	Ch # 13	D/A # 13 data register
0x1C	Ch # 14	D/A # 14 data register
0x1E	Ch # 15	D/A # 15 data register
0x40	Trigger	Writing to this address generates an internal trigger.
0x44	Control Register	Trigger configuration. Specifies which trigger to use, and if needed, where to output an external trigger
0x48	Command Register	Command sent to the DAC in the next access.

Table 2.1: SoftDAC IOSPACE Address Map

2.1.3 INT SAMP CLK (Read / Write 32 bits)

This register sets the sampling rate of the internal sampling rate generator. The internal sampling rate generator is based on a 32 MHz oscillator on the card. The sampling rate is set by the following formula where N is the contents of this register.

$$SamplingRate = \frac{32000000}{2 + N}$$

Since the maximum sampling rate supported by the DACs on the **IP-SOFTDAC-M** is 500 KHz, the smallest value for *N* should be 62.

This register can be accessed in WORD and DWORD modes.

2.1.4 SM ADDRESS (Read Only 24 bits)

This register reports the current buffer address of the state machine. It can be cleared by writing to either **RESET ADDRESS** or **SWITCH BANKS**.

 Copyright ALPHI Technology Corp, 2006

The current buffer address is the next offset into the active bank, which will be written to the DACs on the next sample clock. Although the address counter is 24 bits, only the lowest 13 bits are significant at this time.

This register can be accessed in WORD and DWORD modes.

2.1.5 LAST ADDR 0 / LAST ADDR 1 (Read / Write 24 bits)

These registers contain the last valid data point address when the card is operated in state machine mode. Each bank has its own individual last address.

Although these registers and the address counter are 24 bits, only the lowest 13 bits are significant at this time.

Note: If the last address register for the currently active bank is changed while the state machine is active, it is possible that the current address will pass the new last address. In this case, the current address register will continue to increment, and will wrap to 0 after up to 2^{24} sample clocks.

This register can be accessed in WORD and DWORD modes.

2.1.6 BANK 0 CTRL / BANK 1 CTRL (Read / Write 4 bits)

These registers can be accessed in BYTE, WORD and DWORD modes.

BIT 03	BIT 02	BIT 01	BIT 00
N/A	INT WHEN DONE	MODE 1	MODE 0

INT WHEN DONE ***Interrupt HOST when bank is done***

When this bit is set to 1 and this bank has output its final value, the appropriate **BANK *N* DONE INT** bit is set to 1 and the HOST is interrupted. When this bit is cleared to 0, no interrupt is generated and no bits are set when the bank is done. This bit is cleared to 0 by a board RESET.

MODE 1, MODE 0 ***What to do when bank is done***

The following table describes the available options when a bank is finished being output.

MODE 1	MODE 0	Description
0	0	Play this bank again from address 0 (Default at board RESET)
0	1	Switch to the other bank at address 0
1	0	Disable the state machine and end output at last value in bank
1	1	Disable the state machine, end output at last value in bank, and set the **UNDERFLOW** bit

Table 2.4: Options at Bank Completion

 Copyright ALPHI Technology Corp, 2006

2.1.7 CTRL/STAT 0 (Read / Write 8 bits)

This register can be accessed in BYTE, WORD and DWORD modes.

BIT 07	BIT 06	BIT 05	BIT 04	BIT 03	BIT 02	BIT 01	BIT 00
AUTO UPDATE DAC	ENABLE FP_RST	ENABLE STATE MACH	UNDER FLOW	ENABLE EXT SAMP CLOCK	ENABLE INT SAMP CLOCK	ENABLE CLOCK OUTPUT	ACTIVE BANK

AUTO UPDATE DAC ***Update outputs on manual DAC writes***

When this bit is set to 1, a manual update to a DAC register will also immediately update the output voltage. When this bit is cleared to 0, a manual write will only store the value in a holding register, and the output voltage can be updated by manually writing to the update address **UPDATE DACS**, or by the internal or external sampling clocks if enabled.

This bit is cleared to a 0 by a board RESET.

ENABLE FP_RST ***Allows FP_RST from front panel connector to stop output***

When this bit is set to 1, a negative edge at the **FP_RST** line at the front panel connector will clear certain bits in the control status register in order to stop DAC output. When this bit is cleared to a 0, the **FP_RST** line has no effect. The state of this bit does not affect the generation of an interrupt based on the **FP_RST** line.

This bit is cleared to a 0 by a board RESET.

ENABLE STATE MACH Enables automatic output from the RAM buffers

When this bit is set to 1, the card will automatically reload the DAC registers from the output buffers at each sampling clock based on the current address and the active bank. When this bit is cleared to 0, no automatic updates will occur.

This bit is cleared to a 0 by a board RESET, and by a negative edge on **FP_RST**, if enabled.

UNDER FLOW ***Card is reporting an underflow state***

This bit can potentially be used as a fault bit when the state machine is used for arbitrary output and the buffers are written alternately by an interrupt routine. If for some reason, the currently playing buffer reaches the end before the other buffer is updated and the mode bits updated to *switch banks* (from *stop output with underflow*), the output will stop and the underflow bit will be set.

The user can directly write this bit into either state.

This bit is cleared to a 0 by a board RESET.

ENABLE EXT SAMP CLOCK, ENABLE INT SAMP CLOCK

These bits determine the source of the sampling clock as demonstrated in the following table.

ENABLE EXT	ENABLE INT	

 Copyright ALPHI Technology Corp, 2006

SAMP CLOCK	SAMP CLOCK	SOURCE
X	1	Internal sampling clock generator
1	0	External sampling clock from **SAMPLING_CLOCK_IN**
0	0	Writes to **UPDATE DACS** generate sampling clock

Table 2.5: Sampling Clock Options

These bits are cleared to a 0 by a board RESET, and by a negative edge on **FP_RST**, if enabled.

ENABLE CLOCK OUTPUT ***Enables output of selected sampling clock***

When this bit is set to a 1, the sampling clock source as determined from the above table is output on the SAMPLING_CLOCK_OUT line of the front panel connector. When this bit is cleared to 0, no sampling clock is output.

This bit is cleared to a 0 by a board RESET.

ACTIVE BANK (Read Only) ***Reports active bank***

This read only bit reports the current active bank for the state machine. The current active bank can be changed by the state machine switching banks automatically upon reaching the end of the bank, or by the user writing to **SWITCH BANKS**.

2.1.8 CTRL/STAT 1 (Read / Write 8 bits)

This register can be accessed in BYTE, WORD and DWORD modes.

BIT 07	BIT 06	BIT 05	BIT 04	BIT 03	BIT 02	BIT 01	BIT 00
INT FP_RST	INT SAMPLE CLOCK	INT BANK 1 DONE	INT BANK 0 DONE	N/A	ENABLE SAMPLE CLK INT	ENABLE FP_RST INT	CURRENT STATE FP_RST

INT FP_RST ***Interrupt caused by FP_RST going low***

When set, this bit indicates that an interrupt was generated by FP_RST going low. FP_RST can also stop the card from outputting values, and reset the DACs to 0 if enabled by ***ENABLE FP_RST***. Otherwise, **FP_RST** can be used as an external interrupt.

This bit is cleared by writing a 1 to it. Writing a 0 will not affect the state of this bit.

INT SAMPLE CLOCK ***Interrupt caused by sampling clock***

When set, this bit indicates that an interrupt was generated by the sampling clock triggering an update of the DAC outputs. The user can use this interrupt to write the next outputs to the DAC holding registers prior to the next sample clock.

This bit is cleared by writing a 1 to it. Writing a 0 will not affect the state of this bit.

INT BANK 1 DONE ***Interrupt caused by Bank 1 completing***

When set, this bit indicates that an interrupt was generated by the state machine writing the last value from Bank 1 to the DAC holding registers. The user can use this interrupt to fill Bank 1 with the next outputs when the card is operated alternating between both banks.

This bit is cleared by writing a 1 to it. Writing a 0 will not affect the state of this bit.

 Copyright ALPHI Technology Corp, 2006

INT BANK 0 DONE ***Interrupt caused by Bank 0 completing***

When set, this bit indicates that an interrupt was generated by the state machine writing the last value from Bank 0 to the DAC holding registers. The user can use this interrupt to fill Bank 0 with the next outputs when the card is operated alternating between both banks.

This bit is cleared by writing a 1 to it. Writing a 0 will not affect the state of this bit.

ENABLE SAMPLE CLK INT ***Enable interrupt on sampling clock***

When this bit is set to 1, an interrupt is generated during each sampling clock. When this bit is cleared to 0, no interrupt is generated.

Note that interrupt latency issues restrict the use of this interrupt to fairly slow sampling clock frequencies.

ENABLE FP_RST INT ***Enable interrupt on FP_RST going low***

When this bit is set to 1, an interrupt is generated by a falling edge of FP_RST. When this bit is cleared to 0, no interrupt is generated. FP_RST can also stop the card from outputting values, and reset the DACs to 0 if enabled by ***ENABLE FP_RST***. Otherwise, **FP_RST** can be used as an external interrupt.

CURRENT STATE FP_RST (Read Only) ***Current state of the FP_RST input***

This read only bit reports the current state of the **FP_RST** line from the front panel connector.

2.1.9 RESET SAMP CLK (Write Strobe)

Writing to this location will force the internal sampling clock generator to reload the count from the **INT SAMP CLK** register.

2.1.10 RESET ADDRESS (Write Strobe)

Writing to this location will clear the **SM ADDRESS** counter.

NOTE: If this is written while the state machine is active, there will be a phase jump in the output, and additionally, if the switch occurs in the middle of an update, then some outputs may be updated from the old location, and some from the new location for one sample clock.

2.1.11 RESET DACS (Write Strobe)

Writing to this location will clear DAC holding registers, and force the DACs to output 0 volts.

2.1.12 UPDATE DACS (Write Strobe)

Writing to this location will generate a manual sample clock pulse. It can be used in a full manual mode to update all DAC outputs simultaneously.

2.1.13 SWITCH BANKS (Write Strobe)

Writing to this location will clear the **SM ADDRESS** counter and switch the active bank.

 Copyright ALPHI Technology Corp, 2006

NOTE: If this is written while the state machine is active, there will be a phase jump in the output, and additionally, if the switch occurs in the middle of an update, then some outputs may be updated from the old bank, and some from the new bank for one sample clock.

2.1.14 DAC (Write Only)

These 16 registers will directly write into the holding registers of the DACs. Writes can also force an update of the DAC if the ***AUTO UPDATE DAC*** bit is set. Otherwise, the update will occur on the next sampling clock.

2.2 RAM Buffer Region

There are 8192 output locations available for each DAC channel in each bank. The output buffers can be directly written and read by the HOST.

When the state machine is active, the currently outputting bank is not accessible, but the other bank is accessible. The current bank can be determined by a status bit in the control status register.

The RAM buffers are addressable in WORD and DWORD modes. Most HOST bridge chipsets will compact individual WORD writes into DWORD writes, which are more efficient on a CPCI bus without any additional effort by the customer.

Start	End	Bank	Channel	Start	End	Bank	Channel
0x00000	0x03FFF	0	DAC01	0x40000	0x43FFF	1	DAC01
0x04000	0x07FFF	0	DAC02	0x44000	0x47FFF	1	DAC02
0x08000	0x0BFFF	0	DAC03	0x48000	0x4BFFF	1	DAC03
0x0C000	0x0FFFF	0	DAC04	0x4C000	0x4FFFF	1	DAC04
0x10000	0x13FFF	0	DAC05	0x50000	0x53FFF	1	DAC05
0x14000	0x17FFF	0	DAC06	0x54000	0x57FFF	1	DAC06
0x18000	0x1BFFF	0	DAC07	0x58000	0x5BFFF	1	DAC07
0x1C000	0x1FFFF	0	DAC08	0x5C000	0x5FFFF	1	DAC08
0x20000	0x23FFF	0	DAC09	0x60000	0x63FFF	1	DAC09
0x24000	0x27FFF	0	DAC10	0x64000	0x67FFF	1	DAC10
0x28000	0x2BFFF	0	DAC11	0x68000	0x6BFFF	1	DAC11
0x2C000	0x2FFFF	0	DAC12	0x6C000	0x6FFFF	1	DAC12
0x30000	0x33FFF	0	DAC13	0x70000	0x73FFF	1	DAC13
0x34000	0x37FFF	0	DAC14	0x74000	0x77FFF	1	DAC14
0x38000	0x3BFFF	0	DAC15	0x78000	0x7BFFF	1	DAC15
0x3C000	0x3FFFF	0	DAC16	0x7C000	0x7FFFF	1	DAC16

Table 2.6: RAM Buffer Locations

2.3 MODES OF OPERATION

The card can be viewed as operating in one of the following modes.

- State Machine providing Automatic Update and Load on Sampling Clock
- Manual Load with Update on Sampling Clock
- Manual Load and Update

 Copyright ALPHI Technology Corp, 2006

2.3.1 State Machine providing Automatic Update and Load on Sampling Clock

The card contains a state machine capable of automatically loading the DAC holding registers from the RAM buffers on each sample clock. There are two banks of RAM buffer so that one can be updated by the HOST while the other is being output. Full interrupt support to the HOST allows for easy synchronization of the buffer updates.

Additionally, a single RAM buffer can be played out repeatedly until a change in signal is required, at which time the HOST can write the new waveform to the other buffer and set up a switch at the end of the previous one to ensure phase consistency. Separate length registers for each bank help to achieve this functionality.

On each sampling clock, the DACs are updated from the holding registers, then 16 values are read from the active buffer bank into the holding registers for the next sampling clock.

On the first sampling clock after the state machine is enabled, the DAC holding registers will contain zero if the DACs were RESET. The first data point will be output on the second sampling clock. When the state machine is disabled at the end of a bank, the actual last point is output one sampling clock later.

2.3.2 Algorithm for arbitrary output using both buffers

Starting Output

Ensure that the sampling clock is turned off, and that the state machine is turned off. Ensure that BANK 0 is active.

Hook HOST interrupt service routine to the card.

Program both bank length registers, and load the first data to be played into bank 0, and the second data into bank 1. Set bank 0 to switch banks at the end, and to generate interrupts. Set bank 1 to disable state machine and set underflow, and to generate interrupt.

Enable the state machine and the sampling clock. Continue with non-interrupt processing.

Interrupt Routine

Check underflow bit. If it is set, then interrupt latency would have caused gaps in the output. Check and clear the interrupt cause.

Write the next frame of data to the inactive bank.

Set inactive bank to disable state machine and set underflow and to generate interrupt, and set active bank to switch banks at the end and to generate interrupt. This can be done in one WORD write.

Return from interrupt.

Stopping Output

Disable the state machine and the sampling clock. Disconnect the interrupt routine.

 Copyright ALPHI Technology Corp, 2006

2.3.3 Manual Load with Update on Sampling Clock

In this mode, the card will transfer the values in the DAC holding registers to the output on each internal or external sampling clock. A HOST interrupt is used to have the HOST load the next set of data to the DAC holding registers.

Obviously, this mode will make much greater demands of the HOST as it will be interrupted at every sample clock. Sample rates above a few kHz will not be possible due to the needs of the interrupt routine.

2.3.4 Manual Load and Update

This is a purely manual mode of operation, without making use of any timing on the part of the card. The HOST can write the desired values to the DAC holding registers and then write to **UPDATE DACS** to update all 16 DACs at the same time.

Alternatively, if the ***AUTO UPDATE DAC*** bit is set, the DACs will update the output voltage at the same time that the holding register is written.

2.3.5 MEM SPACE

2.3.5.1 Waveform Sram storage

2.3.5.2 Flash

The SOFTDAC board contains an AT28C64 8K by 8 Flash EPROM.

It is available to the user to stored information related to the module offset gain error for eventual software correction

 Copyright ALPHI Technology Corp, 2006

2.4 ANALOG OUTPUT

The **IP-SOFTDAC-M** has sixteen analog outputs each with its own buffer.

The output ranges are programmable using the board control register.

2.4.1 LTC1592 Command Structure

The LTC1592 receive serially a 24-bit input word. The 4 first bits are a command. The next 4 bits are unused. The last 16-bits are the value to be digitized.

The input word is send to one of the D/A when a write access takes place to the corresponding data register. The serialization logic on the SOFTDAC module use the 4 lower bits of the Command Register (0x48) as the 4 command bit, and the 16 bits being written as the value to be digitized.

It is not necessary to write into the command register between 2 data access. Whatever value was already there is used. Conversely, writing into the command register does not generate any access to the D/As.

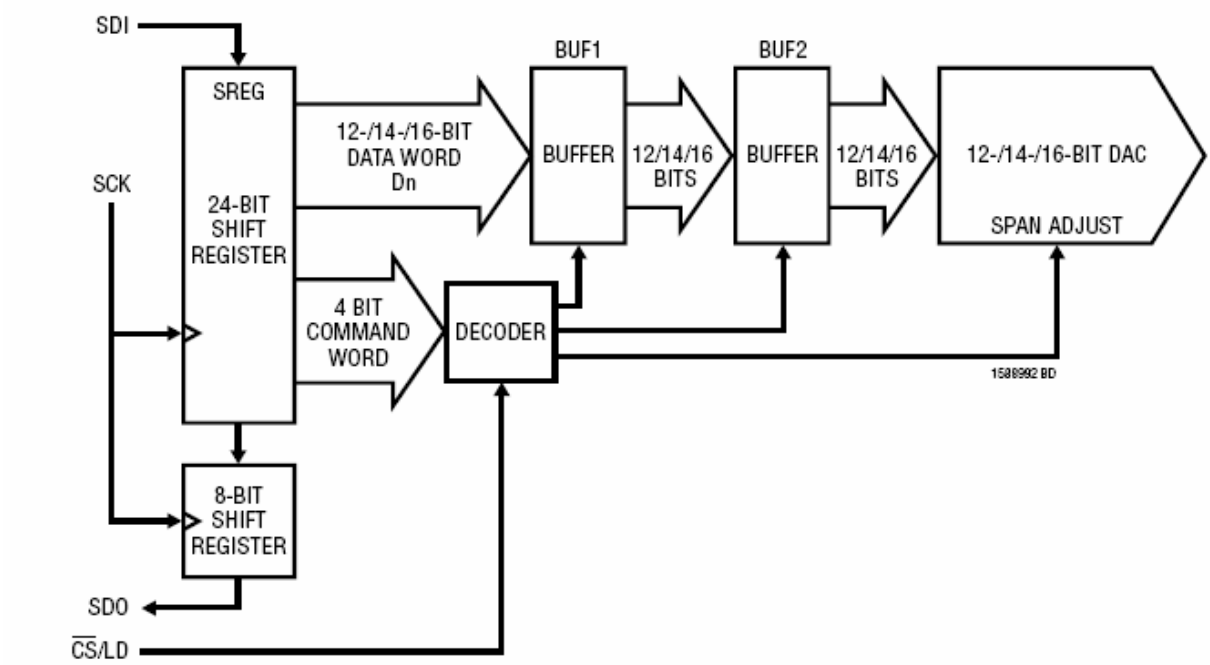

Figure 2.1: LTC1592 Block Diagram

 Copyright ALPHI Technology Corp, 2006

COMMAND				OPERATION EACH COMMAND IS EXECUTED ON THE RISING EDGE OF CS/LD	Internal Register Status			
C3	C2	C1	C0		SREG DATA WORD Dn IN INPUT SHIFT REGISTER	BUF1 INPUT BUFFER	BUF2 DAC BUFFER (DAC OUTPUT)	DAC OUTPUT RANGE
0	0	0	0	Copy Dn in SReg to Buf1. Does not change range.	Dn	Dn	No Change	No Change
0	0	0	1	Copy the Data in Buf1 to Buf2	X	Dn	Dn	No Change
0	0	1	0	Copy Dn in SReg to Buf1 and Buf2 Does not change range.	Dn	Dn	Dn	No Change
0	0	1	1	Reserved (Do Not Use)				
0	1	0	0	Reserved (Do Not Use)				
0	1	0	1	Reserved (Do Not Use)				
0	1	1	0	Reserved (Do Not Use)				
0	1	1	1	Reserved (Do Not Use)				
1	0	0	0	Set Range to 0 to 5V. Copy Dn in SReg to Buf1 and Buf2	Dn	Dn	Dn	5V
1	0	0	1	Set Range to 0 to 10V. Copy Dn in SReg to Buf1 and Buf2	Dn	Dn	Dn	10V
1	0	1	0	Set Range to ±5V. Copy Dn in SReg to Buf1 and Buf2	Dn	Dn	Dn	±5V
1	0	1	1	Set Range to ±10V. Copy Dn in SReg to Buf1 and Buf2	Dn	Dn	Dn	±10V
1	1	0	0	Set Range to ±2.5V. Copy Dn in SReg to Buf1 and Buf2	Dn	Dn	Dn	±2.5V
1	1	0	1	Set Range to -2.5V to 7V. Copy Dn in SReg to Buf1 and Buf2	Dn	Dn	Dn	-2.5V to 7.5V
1	1	1	0	Reserved (Do Not Use)				
1	1	1	1	No Operation	X	No Change	No Change	No Change

Data Word Dn (n = 0 to 15) is the last 16 bits shifted into the input shift register SReg that corresponds to the DAC code.

Table 2.2: LTC1592 Commands

2.4.2 Immediate Mode Operation

Writing in a D/A data register will update the corresponding analog output. Along the data, the content of the command register will be sent to the D/A.

Since the command register contains the output range information, to allow for different ranges in different channels, either the command register will need to be updated between the data writes, or, once all the channels have been programmed with the proper range, the value 0x02 should be used in the command register.

The Command Register value 0x02 allows sending data without range information. It should always be used, in the Command Register, after the initial range programming is completed, particularly if the range selection is not the same among the channels.

 Copyright ALPHI Technology Corp, 2006

```
uint16 *commandReg = (uint16 *)(SOFTDAC_IOSPACE + 0x48);
uint16 *DA_data = (uint16 *)(SOFTDAC_IOSPACE + 0x48);

void initOutput(uint16 range[], uint16 initialValue[])
// range is an array of 16 command word to set the range of each D/A
{
	int i;

	for (i=0; i<16; i++) {
		*commandReg = range [i];
		DA_data[i] = initialValue[i];	// send the initial
			// value and the range data to the D/A
			// the best initial value has to be determined
			// depending on the application
	}

	*commandReg = 0x2;	// value that will allow to change
			// the data without changing the range

}

void output_noTrigger(uint16 out_data[])
// out_data is an array of 16 values to be sent to the D/A
{
	int i;

	for (i=0; i<16; i++) DA_data[i] = out_data[i];
}
```

 Copyright ALPHI Technology Corp, 2006

2.4.3 Trigger-Synchronized Operation

The board has the ability to input or output an external trigger for synchronized updates.

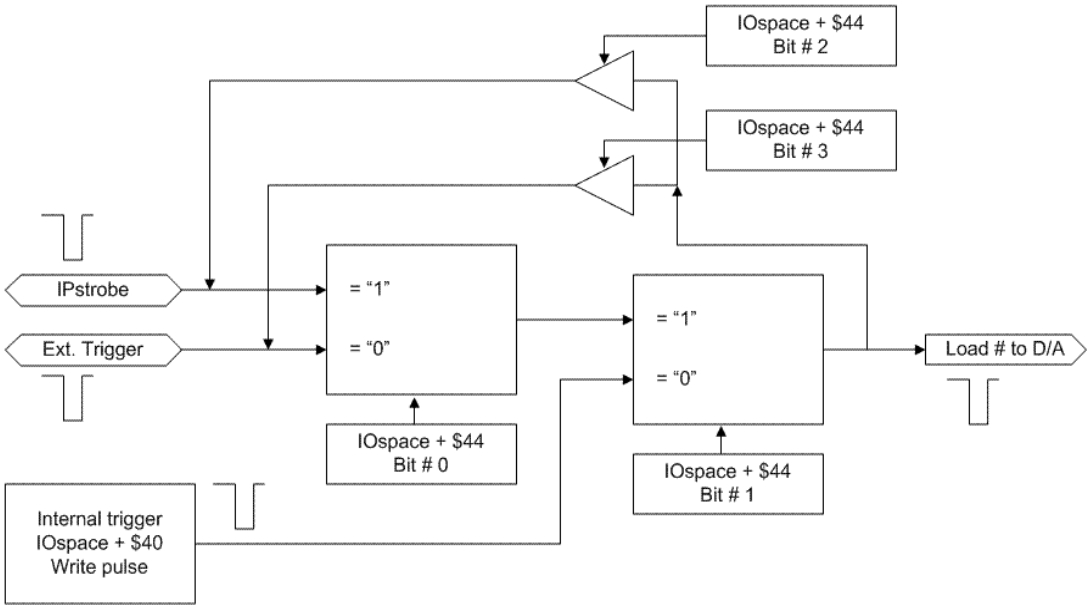

Figure 2.1: External Trigger Block Diagram

The Control Register is responsible for selecting the trigger source, and possible trigger output to synchronize other boards.

The source can come from the on-board logic. In this case, writing to the trigger register will generate the trigger signal. It can also come from the IP Strobe line or from the External Trigger pin on the IP I/O connector.

For system integrity reasons, the trigger is disabled if the Command Register does not contain 0x01

The trigger can be output on either the IP Strobe line or from the External Trigger pin on the IP I/O connector, depending on the control register programming

To use the trigger synchronized operation, follow this sequence:

- Set the Command Register to 0: this will allow programming the D/As while preventing the outputs from being updated.
- Write the updated data in each D/A data registers.
- Set the Command Register to 1: this will tell the D/A to update the outputs, however, this command will not be sent until the trigger is received.
- Generate the internal trigger by writing any value to the Trigger Register, **or** wait for the outside trigger.

 Copyright ALPHI Technology Corp, 2006

Control Register Data	Trigger Input Source	Trigger Output Destination
0x00	Internal Trigger	No output
0x02	External Trigger from the Event line	No output
0x03	External Trigger from the Strobe line	No output
0x04	Internal Trigger	Strobe line
0x06	External Trigger from the Event line	Strobe line
0x08	Internal Trigger	Event line
0x0B	External Trigger from the Strobe line	Event line
0x0C	Internal Trigger	Event line, Strobe line
others	Reserved (Do Not Use)	Reserved (Do Not Use)

Upon trigger reception, the logic sends simultaneously a write to all the D/As. As in a normal write, the 4 command bits are coming from the command register. The 16 data bits are the same value as the latest write to the D/A register.

This trigger operation is intended to be used with 0x01 in the command register. Using that command, the data field of the 24-bit input word is ignored.

It should be noted that writing in the command register does not write to the D/A. The command register value is sent to the D/As in only 2 cases:

- One particular D/A when writing the D/A data register.
- All the D/As simultaneously when receiving a trigger, provided that the command register contains 0x01.

 Copyright ALPHI Technology Corp, 2006

```
uint16 *commandReg = (uint16 *)(SOFTDAC_IOSPACE + 0x48);
uint16 *DA_data = (uint16 *)(SOFTDAC_IOSPACE + 0x48);
uint16 *DA_trigger = (uint16 *)(SOFTDAC_IOSPACE + 0x40);

void output_noTrigger(uint16 out_data[])
// out_data is an array of 16 values to be sent to the D/A
{
        int i;

        *commandReg = 0;    // prevents the D/A from
                            // updating the output

        for (i=0; i<16; i++) DA_data[i] = out_data[i];  // the data is set
                            // in the data latches but the DA will not
                            // update the outputs yet

        *commandReg = 1;    // set the command register to update
                            // the output with the data stored
                            // already stored in the D/A

        *DA_trigger = 0;        // send simultaneously a command word
                                // to all the D/A, this updates the output
                            // synchronously.  The data is the same
                            // as earlier but, anyway, not used since the
                            // command is '1'.

}
```

2.4.4 External 5 Volt Reference

The jumper area J2 allows selecting an external reference for the D/As.

When pins 1 and 2 are connected, the channels 0-7 use the internal 5-Volt reference, when pins 3 and 4 are connected; the channels 0-7 use the external reference.

When pins 5 and 6 are connected, the channels 8-15 use the internal 5-Volt reference, when pins 7 and 8 are connected; the channels 8-15 use the external reference.

 Copyright ALPHI Technology Corp, 2006

3 APPENDIX A: OUTPUT CONNECTOR

Pin	Signal	Pin	Signal
1	D/A # 0	26	AGND
2	D/A # 1	27	AGND
3	D/A # 2	28	AGND
4	D/A # 3	29	AGND
5	D/A # 4	30	AGND
6	D/A # 5	31	AGND
7	D/A # 6	32	AGND
8	D/A # 7	33	AGND
9	D/A # 8	34	AGND
10	D/A # 9	35	AGND
11	D/A # 10	36	AGND
12	D/A # 11	37	AGND
13	D/A # 12	38	AGND
14	D/A # 13	39	AGND
15	D/A # 14	40	AGND
16	D/A # 15	41	AGND
17		42	
18		43	
19		44	
20		45	
21		46	
22		47	
23		48	
24	GND	49	AGND
25	Ext. Trigger	50	Ext. 5V Reference

 Copyright ALPHI Technology Corp, 2006