IP-SOFTDAC-M

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

PROGRAMMING MANUAL

819-20-000-4000

Version 1.1

September 2013

ALPHI TECHNOLOGY CORPORATION
1898 E. Southern Ave
Tempe, AZ 85282 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.

Table of Contents

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).

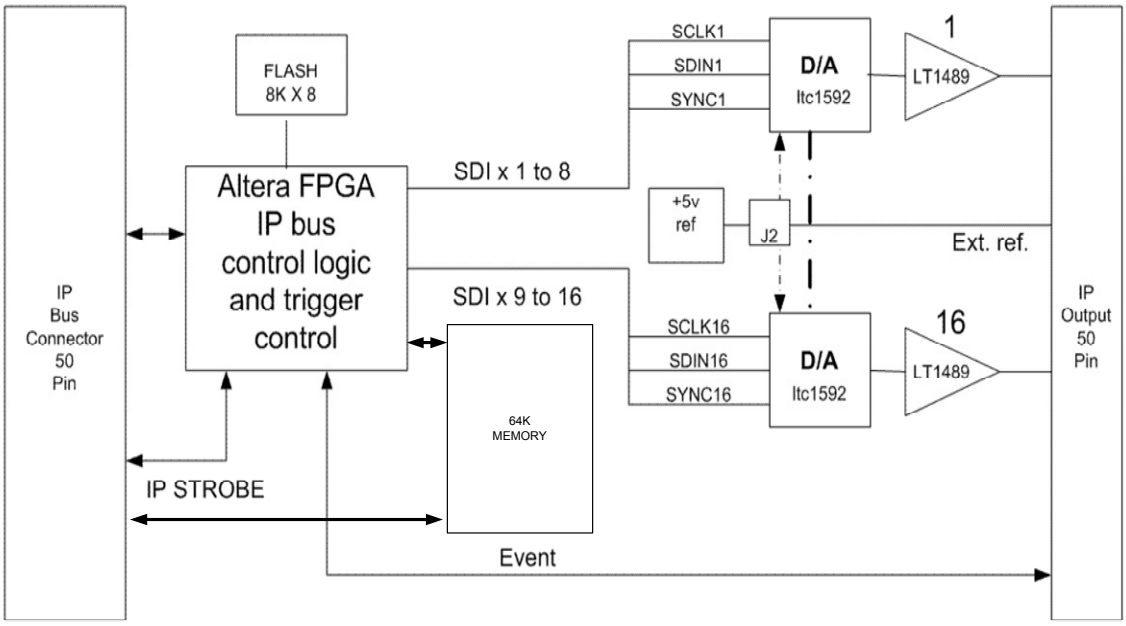

Figure 1.1: Block Diagram

1.3 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 and Update

1.3.1 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.
On each sampling clock, the DACs are updated from the holding registers. Sixteen values are then 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.

1.3.2 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.

1.4 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.

1.4.1 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.

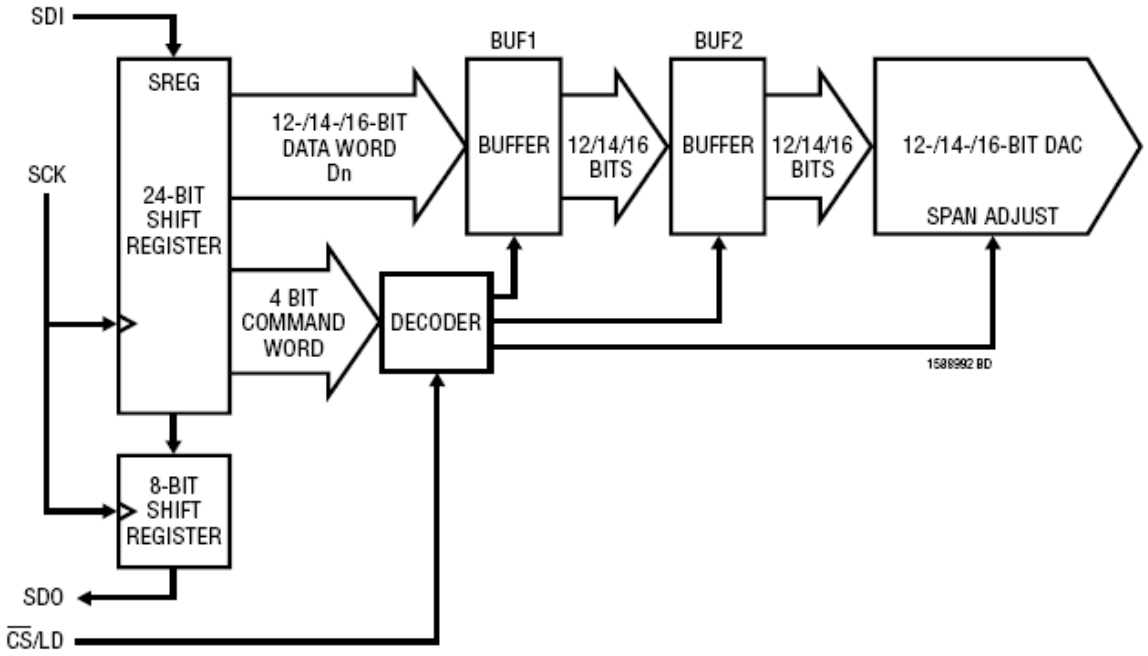

Figure 2.1: LTC1592 Block Diagram

1.4.2 Command Register (Read/Write)

ADDRESS: IOspace + 0x48
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.
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.

Each D/A converter can be programmed for different range.
First the Command register is loaded with the desired value, then a write to the selected D/A will send the information to the D/A.

1.4.3 Global Update

It is possible to update all the D/A with the same Command Register value by setting bit # 4 to a "1". Therefore a sole write to any D/A will transfer the Command Register value to all the D/A.
Example: Setting all the D/A output to +/-10v, then immediate transfer of next data to the output.

-write 0x1B to address 0x48
-write any data to D/A #1 at address 0x20

1.4.4 Immediate Mode Operation

-write 0x02 to address 0x48

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.

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.	Dn	Dn	Dn	10V

				Copy Dn in SReg to Buf1 and Buf2				
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

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.

```
uint16 *commandReg = (uint16 *)(SOFTDAC_IOSPACE + 0x48);
uint16 *DA_data = (uint16 *)(SOFTDAC_IOSPACE + 0x20);

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];
}
```

Also setting bit #4 of the CMD_WORD_REG will allow to program all the D/A with the same voltage by writing only once to any D/A converter.

1.4.5 Trigger-Synchronized Operation

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

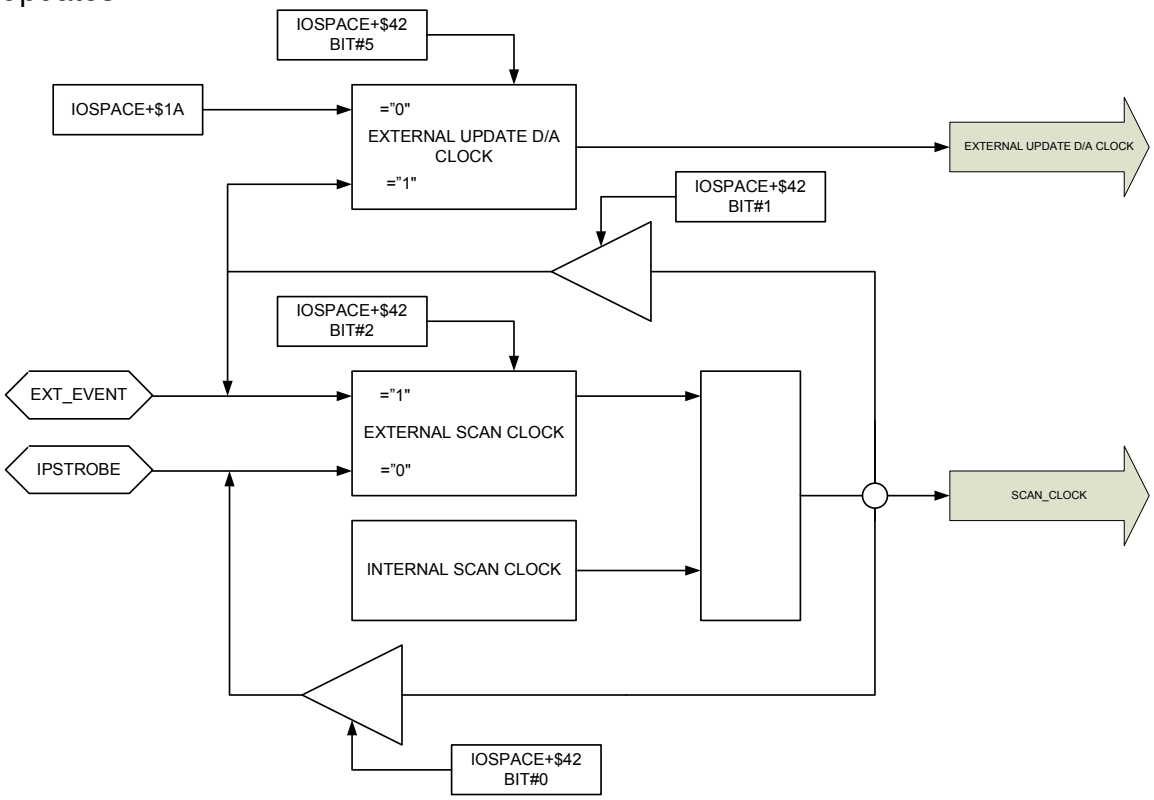

Figure 2.1: External Trigger Block Diagram

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 and/or to the External Event 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.

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.

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

```
uint16 *DA_data = (uint16 *)(SOFTDAC_IOSPACE + 0x20);
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'.

}
```

1.5 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.

2 IP INTERFACE

2.1 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 “C” (8 MHz)	0x43
0x09	Manufacturer identification	0x11
0x0B	Module type	0x23
0x0D	Revision module	0x0A
0x0F	Reserved	

Table 2-2 IDSPACE content 8MHz IP

Only the 8 lower bits of the word are valid. Depending on the system endianness, the values might be seen on the odd or even addresses. The idSpace accesses the lower 64 bytes of the FLASH. It is possible for the user to modify the idSpace content just by writing into it.

2.2 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	Size	R/W	COMMENTS
Internal Sampling Clk	0x00	DW	R/W	Divisor for Internal Sampling Clock
Current SM Address	0x04	DW	RO	Current Address for the State Machine
Last SM Address	0x08	DW	R/W	Last Address with Valid Data, Bank 0
Bank Control	0x10	DWB	R/W	Controls Bank 0
CTRL/STAT 0	0x12	DWB	R/W	General control and status

CTRL/STAT 1	0x13	DWB	R/W	General control and status
Reset Sampling Clk	0x14	W	WS	Reset sampling clock counter
Reset SM Address	0x16	W	WS	Reset address counter
Reset DACs Strobe	0x18	W	WS	Reset all DACs
Update DACS Strobe	0x1a	W	WS	Update all DACs
DAC01 data	0x20	DW	W	Direct DAC output
DAC02 data	0x22	DW	W	Direct DAC output
DAC03 data	0x24	DW	W	Direct DAC output
DAC04 data	0x26	DW	W	Direct DAC output
DAC05 data	0x28	DW	W	Direct DAC output
DAC06 data	0x2a	DW	W	Direct DAC output
DAC07 data	0x2c	DW	W	Direct DAC output
DAC08 data	0x2e	DW	W	Direct DAC output
DAC09 data	0x30	DW	W	Direct DAC output
DAC10 data	0x32	DW	W	Direct DAC output
DAC11 data	0x34	DW	W	Direct DAC output
DAC12 data	0x36	DW	W	Direct DAC output
DAC13 data	0x38	DW	W	Direct DAC output
DAC14 data	0x3a	DW	W	Direct DAC output
DAC15 data	0x3c	DW	W	Direct DAC output
DAC16 data	0x3e	DW	W	Direct DAC output
Trigger	0x40			Writing to this address generates an internal trigger
Control Register	0x42			Scan clock selection/flash/memory update mode
Command Register	0x48			Command sent to the DAC in the next access
Interrupt vector register #0	0x42		R/W	Interrupt vector register provide during an interrupt cycle.
Interrupt vector register #1	0x44		R/W	Interrupt vector register provide during INTSELA cycle.
Reset Interrupt #0	0x50		W	Reset Interrupt #0 latch
Reset Interrupt #1	0x52		W	Reset Interrupt #1 latch

Table 2.1: SoftDAC IOSPACE Address Map

2.2.1 Internal Sample Clock (Read / Write 32 bits)

ADDRESS: 0x00

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.(0x3E hex)

This register can be accessed in WORD .

2.2.2 State machine Address Register (Read Only 24 bits)

ADDRESS: 0x04

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

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 .

2.2.3 Last Address Register (Read / Write 24 bits)

ADDRESS: 0x08 LAST ADDR 0

This 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 .

2.2.4 Bank Control Register (Read / Write 8 bits)

ADDRESS: 0x10 BANK 0 CTRL

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

Bit 7-3	Bit 2	Bit 1	Bit 0
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	Not used
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

2.2.5 Control/Status Register 0 (Read / Write 8 bits)

ADDRESS: IOspace + 0x12
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
0	0	ENABLE STATE MACH	0	ENABLE EXT SAMP CLOCK	ENABLE INT SAMP CLOCK	ENABLE CLOCK OUTPUT	0

ENABLE STATE MACH: Enables automatic output from the RAM buffers
When this bit is set to 1, the state machine copying automatically the data from the channel memory to the DAC is enabled..
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 SAMP CLOCK	ENABLE INT 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
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 set to a 0 by a board RESET.

2.2.6 Control/Status Register 1 (Read / Write 8 bits)

ADDRESS: IOspace + 0x13

BIT 07	BIT 06	BIT 05	BIT 04	BIT 03	BIT 02	BIT 01	BIT 00
0	IRQ SAMPLE CLOCK	0	IRQ BANK DONE	0	ENABLE SAMPLE CLK IRQ	0	0

IRQ SAMPLE CLOCK: Interrupt caused by sampling clock
This bit indicates that an interrupt was generated by the sampling clock while updating 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.
IRQ BANK DONE: Interrupt caused by Bank completing

This bit indicates that an interrupt was generated by the state machine writing the last value from the data bank to the DAC holding registers. The user can use this interrupt to fill the data bank 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 IRQ: 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.

2.2.7 Reset Sampling Clock Strobe (Write Strobe)

ADDRESS: IOspace + 0x14
Writing to this location will force the internal sampling clock generator to reload the count from the **internal sample clock** register.

2.2.8 Reset SM Address Strobe (Write Strobe)

ADDRESS: IOspace + 0x16
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.

2.2.9 Reset DACs Strobe (Write Strobe)

ADDRESS: IOspace + 0x18
Writing to this location will clear DAC holding registers, and force the DACs to output 0 volts.

2.2.10 Update DACs Strobe (Write only)

ADDRESS: IOspace + 0x1A
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.

2.2.11 DAC data registers (Write Only)

ADDRESS: IOspace + 0x20 - IOspace + 0x3E
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.

2.2.12 Control Register

ADDRESS: IOspace + 0x42

BIT 7-6	BIT 05	BIT 04	BIT 03	BIT 02	BIT 01	BIT 00
Not used	Update slt	Ram Update	Flash enable	Ext scan selection	Event out enable	Strobe out enable

Update D/A selection
When postponing update of the D/A is selected, the update can be triggered by a Host write to Update DAC Strobe or by using the external pin signal Ext_EVENT as source pulse. (bit#5= “1”).Care must be made not to enable the EXT_EVENT buffer as output or to select the SCAN clock as EXT_EVENT.
RAM Update
When set to “1”, updating the RAM buffer by the host occurs only at the end of the current cycle while the state machine is running.
The drawback is the fact that the update will hold the IP_BUS up to 2 µS. This might slow down the performance of the IP carrier.
Flash enable
When this bit is set to 1, memSpace accesses are directed to the FLASH memory. The first 16 bytes of the memory are reserved for the idSpace access and should not be modified. The remaining 8kbytes are at the disposition of the software. When this bit is set, the state machine should not be running.
EXT_SCAN selection
Upon reset the signal strobe is the source of the external scan signal.
If the bit #2 is set to “1”, the signal ex_event will be the source for the DAC update strobe.
Event out enable
This bit #1 need to be set to “1” to have the internal Scan clock to be output on the EX_Event signal front connector.
Strobe out enable
This bit needs to be set to “1” to have the internal scan clock output to the IPSTOBE pin of the IP-BUS connector.

2.3 MEM Space

The board features a 128 Kbyte memory accessible in the board memSpace. There are 8192 output locations available for each DAC channel. The output buffers can be directly written and read by the HOST.
It is preferable not to access the memory when the state machine is active. Since the memory cannot be accessed reliably while the DAC are being updated.
The Control Register *Ram Update* bit allows 2 different mode access:

- *Ram Update* = "0".When the state machine is active, the memory is accessible, but glitches can occur and bad data value will be written. This mode can be used when the software insures that the memory is not accessed while an update of the DAC is taking place.

- *Ram Update* = "1". A request for a write into memory will be granted only at the end of the current update cycle. It prevents glitches but the request can last close to 2 µS.

PCI bus will be held during all this time or it will initiate a retry, slowing sown the bus performance.

.

Start	End	Bank	Channel
0x00000	0x03FFF	0	DAC01
0x04000	0x07FFF	0	DAC02
0x08000	0x0BFFF	0	DAC03
0x0C000	0x0FFFF	0	DAC04
0x10000	0x13FFF	0	DAC05
0x14000	0x17FFF	0	DAC06
0x18000	0x1BFFF	0	DAC07
0x1C000	0x1FFFF	0	DAC08
0x20000	0x23FFF	0	DAC09
0x24000	0x27FFF	0	DAC10
0x28000	0x2BFFF	0	DAC11
0x2C000	0x2FFFF	0	DAC12
0x30000	0x33FFF	0	DAC13
0x34000	0x37FFF	0	DAC14
0x38000	0x3BFFF	0	DAC15
0x3C000	0x3FFFF	0	DAC16

Table 2.6: RAM Buffer Locations

2.4 Flash

The SOFTDAC board contains an AT28C64 8K by 8 Flash EPROM.
The lower 64 bytes (0x40) are visible and modifiable from the idSpace.
The bit #3 of the Control Register (0x42) is used to switch the memSpace access from the data storage RAM to the FLASH access. The state machine should be idle when accessing the FLASH.
The FLASH might be used by the user to store information related to the module. For instance offset and gain error for eventual software correction, or information related to other elements of the system.
Only the low 8-bits of the word are valid. Depending on the system endianness, the FLASH might be seen on odd or even addresses.

2.5 Interrupt

2.5.1 Interrupt source

Sole source of interrupt is when the last valid address in the buffer has been reached. The interrupt is enabled by setting the appropriate bit (3) of the BANK 0 CTRL register.

2.5.2 Interrupt Vector register

ADDRESS: IOspace + 0x44 ADDRESS: IOspace + 0x46
ADDRESS: IOspace + 0x50 ADDRESS: IOspace + 0x52
Two interrupt vector registers are available at address I/O space + 0x44 and 0x46 for Interrupt #0 and interrupt #1. The pre-loaded data value is provided into the IPbus when an IPCYCLE is valid.
Reset of the interrupt source is made when:

-the Interrupt cycle is active or

-by writing at the IOSPACE + 0x50 and 0x52 respectively for interrupt #0 and #1

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. EVENT	50	Ext. 5V Reference