51 IIR Accelerator (IIR)

The processor includes an IIR filter accelerator implemented in hardware that reduces the processing load on the core, freeing it up for other tasks.

Features

The accelerator supports a maximum of 24 channels. There is support for up to 12 cascaded bi-quads per channel. This means that the accelerator locally stores all the biquad coefficients of 24 channels. Window size can be configured from 1 (sample based) to 1024. The IIR has the following additional features.

- Supports IEEE floating point format 32/40-bit
- Supports various rounding modes
- Sample based or window based processing
- Up to 12 cascaded biquads per channel
- Up to 24 filter channels available in TDM
- Allows Biquad save state storage

NOTE: The IIR accelerator module has local memory which is not accessible by the core during regular operation mode. Unlike previous SHARC processors, the IIR accelerator modules each have access to the system memory (on-chip or off-chip).

Unlike in previous SHARC processors, where only one of the IIR or FIR Accelerator can be enabled at a time, the ADSP-SC58x processor can use both the IIR and the FIR Accelerators at the same time.

Clocking

The IIR accelerator runs at the maximum speed of SCLK0.

Functional Description

The **IIR Accelerator Block Diagram** shows the various blocks of the IIR hardware accelerator. The accelerator has a coefficient memory size of 1440 × 40 bits (12 biquads × 12 channels × 5 coeffs), a data memory size of 576 × 40 bits (12 biquads × 12 channels × 2 states) and one MAC unit with an input data buffer to supply data to the MAC.

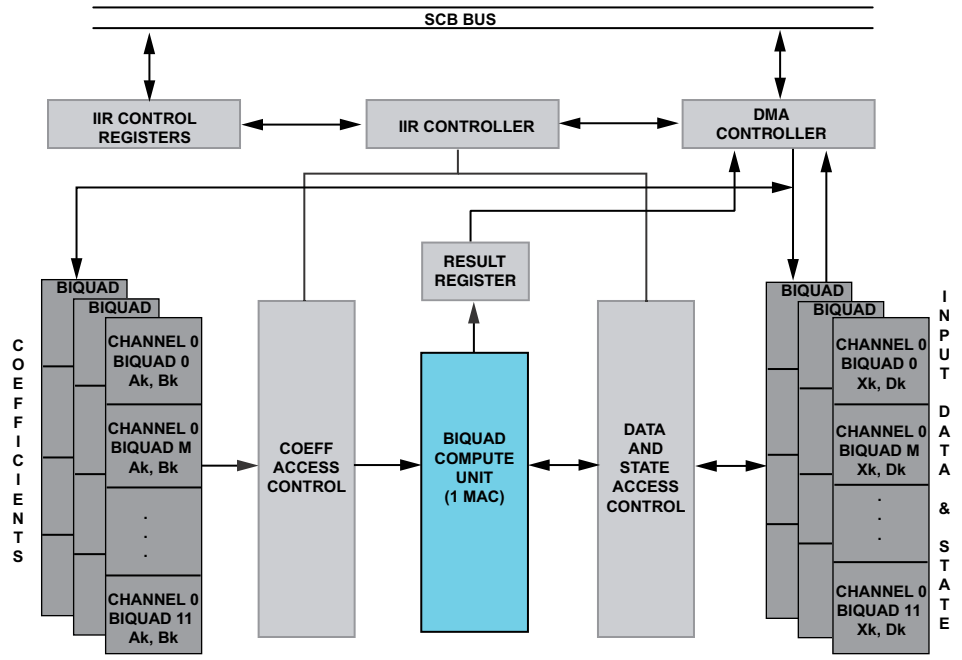

Figure 1: IIR Accelerator Block Diagram

The IIR accelerator is implemented using Transposed Direct Form II biquad which has less coefficient sensitivity. The **Transposed Direct Form II Biquad** figure shows the signal flow graph for the biquad structure.

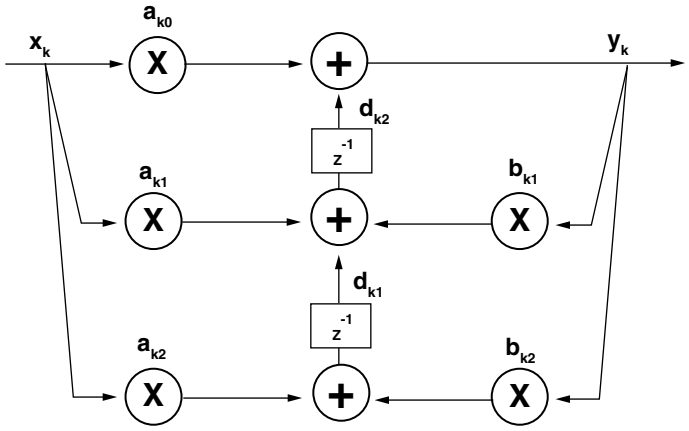

Figure 2: Transposed Direct Form II Biquad

The accelerator has the following logical sub blocks.

- A data path unit with the following elements:
  - 32/40-bit coefficient memory (Ak, Bk) for storing biquad coefficients
  - 32/40-bit input data (Xk) and state (Dk)

  - One 40/32-bit floating-point multiplier and adder (MAC) unit
  - An input data buffer to efficiently supply data to MAC
  - One 40-bit result register to hold result of biquad
- Configuration registers for controlling various parameters such as the number of biquads, the number of channels, interrupt control, and DMA control
- A core access interface for writing the DMA/filter configuration registers and for reading the status registers
- A DMA bus interface for transferring data to and from the accelerator. This interface is also used to preload the coefficients (Ak, Bk) and state (Dk) at start up.
- DMA configuration registers for the transfer of input data, output data and coefficients

ADSP-SC58x IIR Register List

The IIR module contains the following registers.

Table 1: ADSP-SC58x IIR Register List

Name	Description
IIR_CHNPTR	Chain Pointer Register
IIR_COEFIDX	Coefficient Buffer Index Register
IIR_COEFLEN	Coefficient Buffer Length Register
IIR_COEFMOD	Coefficient Index Modifier Register
IIR_CTL1	Global Control Register
IIR_CTL2	Channel Control Register
IIR_DBG_ADDR	IIR Debug Address
IIR_DBG_CTL	IIR Debug Control
IIR_DBG_RDDAT_HI	IIR Debug Read Data High
IIR_DBG_RDDAT_LO	IIR Debug Read Data Low
IIR_DBG_WRDAT_HI	IIR Debug Write Data High
IIR_DBG_WRDAT_LO	IIR Debug Write Data Low
IIR_DMASTAT	DMA Status

Table 1: ADSP-SC58x IIR Register List (Continued)

Name	Description
IIR_INBASE	Input Buffer Base Register
IIR_INIDX	Input Data Index Register
IIR_INLEN	Input Data Buffer Length Register
IIR_INMOD	Input Data Index Modifier Register
IIR_MACSTAT	MAC Status Register
IIR_OUTBASE	Output Buffer Base Register
IIR_OUTIDX	Output Data Buffer Index Register
IIR_OUTLEN	IIR Output Data Buffer Length
IIR_OUTMOD	IIR Output Data Index Modifier

ADSP-SC58x IIR Trigger List

Table 2: ADSP-SC58x IIR Trigger List Masters

Trigger ID	Name	Description	Sensitivity
61	IIR0_DMA	IIR0DMA	Edge

Table 3: ADSP-SC58x IIR Trigger List Slaves

Trigger ID	Name	Description	Sensitivity
None			

Multiply and Accumulate (MAC) Unit

The **IIR MAC Unit** figure shows a pipelined multiplier and accumulator unit that operates on the data and coefficient fetched from the data and coefficient memory. The MAC can perform either 32-bit floating-point or 40-bit floating-point MAC operations. 32-bit floating-point operations generate 32-bit results and 40-bit floating-point operations generate 40-bit results.

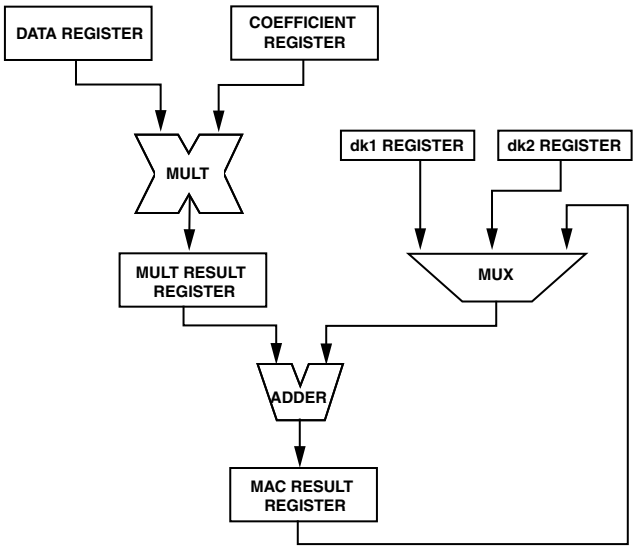

Figure 3: IIR MAC Unit

Input Data and Biquad State

The size of data memory is 576 × 40 bits and is used to hold the dk1 and dk2 state of all the biquads locally. The DMA controller fetches the sample data from internal memory and calculates the output as well as the dk1 and dk2 values for each biquad and stores them in local data memory.

Coefficient Memory

The size of coefficient memory is 1440 × 40 bits and is used to store all the coefficients of all the biquads. At start-up, DMA loads the coefficients from system memory into local coefficient memory.

Internal Memory Storage

This section describes the required storage model for the IIR accelerator.

Coefficient Memory Storage

Coefficients and Dk values for a particular biquad BQD[k] should be stored in internal memory in the order Ak0, Ak1, Bk1, Ak2, Bk2, Dk2, Dk1.

NOTE: The naming convention for the filter coefficients used here is different from the one used in MATLAB. The following conversion should be used when using MATLAB generated coefficients:

(Akx = bx and Bkx = -ax).

In other words, the coefficients for each biquad should be stored in the order:

```
b0, b1, -a1, b2, -a2, dk2, dk1
```

For N biquad stages, the order of coefficients should be as follows:

```
b01, b11, -a11, b21, -a21, dk21, dk11,
b02, b12, -a12, b22, -a22, dk22, dk12,
,.....
b0N, b1N, -a1N, b2N, -a2N, dk2N, dk1N.
```

where bxN and axN are the coefficients ([b, a]) for the Nth biquad stage.

Operating Modes

The accelerator can be operated in the following modes.

Window Processing

Sample based processing mode is selected by configuring window size to 1. In this mode, one sample from a particular channel is processed through all the biquads of that channel and the final output sample is calculated.

In window-based mode, multiple output samples (up to 1024) equal to the window size of that channel are calculated. After these calculations are complete, the accelerator begins processing the next channel. A configurable window size parameter is provided to specify the length of the window.

40-Bit Floating-Point Mode

In 40-bit floating-point mode, the input data/coefficient is treated as a 40-bit floating-point number. 40-bit floating-point MAC operations generate 40-bit results. This mode can be selected by setting the `IIR_CTL1.FORTYBIT` bit.

Since the DMA bus width is 32 bits, in 40-bit mode the IIR accelerator performs two packed 32-bit accesses to the memory to fetch one 40-bit input or coefficient data, or to store one 40-bit output word. The first 32-bit word provides the lower 32 bits and the 8 LSBs of the second 32-bit word provides rest of the upper 8 bits of the a complete 40-bit word. The **32-Bit To 40-Bit Packing** figure shows the 32-40 bit packing used by accelerator.

NOTE: Overheads might be required to pack the input 40-bit data into the format acceptable by the IIR accelerator and for unpacking the output of accelerator to the format acceptable by the rest of the application.

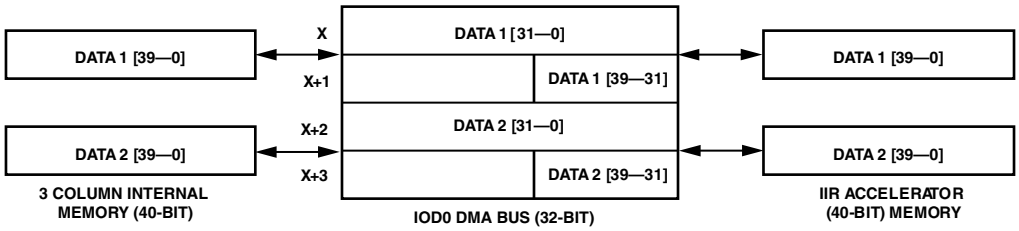

Figure 4: 32-Bit To 40-Bit Packing

Save Biquad State Mode

The IIR_CTL1.SS bit completely stores the current biquad states in local memory (writes all the DK1 and DK2 states back into the system memory states). This is useful in applications that require fast switching to another high priority accelerator task?a required IIR to FIR processing transition for example. After resuming these states can be reloaded and IIR processing can be continued. Note that the DMA status is automatically stored after each iteration.

NOTE: The save state operation cannot be stopped after it starts, even by clearing the IIR_CTL1.EN or IIR_CTL1.DMAEN bits. Although the bits would clear on the core side, settings take effect only after the save state operation completes. Therefore, before trying to disable the IIR accelerator, you must poll the corresponding status bits in the IIR_DMASTAT register to ensure the save state operation completed successfully. The following expressions provide the latency due to the save state operation, assuming no higher priority DMA is ON:

- For 32-bit mode: $14 \times N + ((8 \times M)+2) \times N)$
- For 40-bit mode: $14 \times N + ((15 \times M)+2) \times N)$

where N is the number of channels and M is the number of biquads per channel.

NOTE: Write access to any of the IIR accelerator registers loaded by chaining is not allowed while the save state operation is in progress. Attempted writes to these registers might result in the blocking of IOP core reads until the save state operation completes.

Data Transfers

The IIR filter works exclusively through DMA.

IIR Accelerator TCB

The location of the DMA parameters for the next sequence comes from the chain pointer register. This register points to the next set of DMA parameters stored in the system memory of the processor known as TCB. In chained DMA operations, the processor automatically initializes and then begins another DMA

transfer when the current DMA transfer is complete. Each new set of parameters is stored in a user-initialized memory buffer or TCB for a chosen peripheral.

Chain Assignment

The structure of a TCB is conceptually the same as the structure of a traditional linked-list. Each TCB has several data values and a pointer to the next TCB. Further, the chain pointer of a TCB can point to itself to rerun the same DMA continuously. The FIR accelerator reads each word of the TCB and loads it into the corresponding register. A TCB with a chain pointer register value of zero indicates the end of the chain (no further TCBs are loaded). The IIR accelerator supports circular buffer chained DMA. The **IIR TCBs for Chained DMA** table shows the required TCBs for chained DMA. In the table, TCB refers to the start address of the TCB array.

NOTE: In the IIR accelerator DMA, two different TCB loading sequences are available: one TCB loads five parameters for the coefficients (IIR_CTL2, IIR_COEFIDX, IIR_COEFMOD, IIR_COEFLEN, and IIR_CHNPTR). The second loads 10 parameters for the data (IIR_CTL2, IIR_INIDX, IIR_INMOD, IIR_INLEN, IIR_INBASE, IIR_OUTIDX, IIR_OUTMOD, IIR_OUTLEN, IIR_OUTBASE, and IIR_CHNPTR).

Initialize IIR_CHNPTR to TCB+12.

Table 4: IIR TCBs for Chained DMA

Address	Register		
TCB	IIR_CHNPTR		
TCB + 0x1	IIR_COEFLEN		
TCB + 0x2	IIR_COEFMOD		
TCB + 0x3	IIR_COEFIDX		
TCB + 0x4	IIR_OUTBASE		
TCB + 0x5	IIR_OUTLEN		
TCB + 0x6	IIR_OUTMOD		
TCB + 0x7	IIR_OUTIDX		
TCB + 0x8	IIR_INBASE		
TCB + 0x9	IIR_INLEN		
TCB + 0xA	IIR_INMOD		
TCB + 0xB	IIR_INIDX		
TCB + 0xC	IIR_CTL2		

DMA Access

The IIR accelerator has two DMA channels (accelerator input and output) to connect to the system memory. The DMA controller fetches the data and coefficients from memory and stores the result.

Chain Pointer DMA

The DMA controller supports circular buffer chain pointer DMA. One transfer control block (TCB) needs to be configured for each channel. The TCB contains:

- A control register value to configure the filter parameters (such as number of biquads, window size) for each channel
- DMA parameter register values for the input data
- DMA parameter register values for coefficient load
- DMA parameter register values for output data

NOTE: The chain pointer (IIR_CHNPTR) field of the last channel's TCB should point to the first channel's TCB. This is so that when the IIR accelerator is enabled, 1) it first loads the coefficients (Ak, Bk) and state variables (Dk) for all the channels in to it's local coefficient memory and 2) it loops back to first channel again to start fetching the input data for processing.

The accelerator loads the TCB into its internal registers and uses these values to fetch coefficients and data and to store results. After processing a window of data for any channel, the accelerator writes back the IIR_INIDX (input index register) and IIR_OUTIDX (output index register) values to the TCB in memory, so that data processing can begin from where it left off during the next time slot of that channel.

For 32-bit mode, the write back values for the index registers is equal to IIRII + W and IIROI + W.

For 40-bit mode, the write back values are: IIR_INIDX + 2 × W and IIR_OUTIDX + 2 × W.

Accelerator input and output channels connect to system memory.

NOTE: The IIR_CTL2 register is part of the IIR TCB. This allows programs to program individual IIR channels having different control attributes.

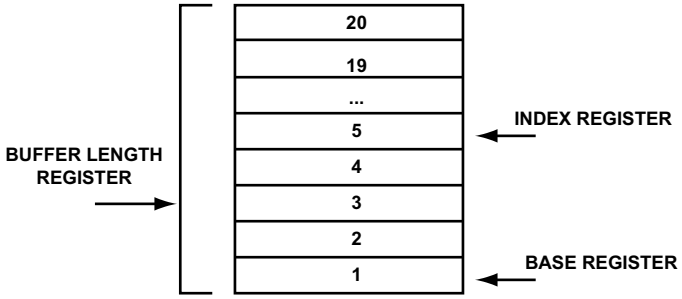

Figure 5: Circular Buffer Addressing

Effect Latency

The total effect latency is a combination of the write effect latency (core access) plus the peripheral effect latency (peripheral specific).

Write Effect Latency

For details on write effect latency, see the *SHARC Processor Programming Reference*.

Interrupts

The **IIR Interrupt Overview** table provides an overview of IIR interrupts.

Table 5: IIR Interrupt Overview

Interrupt	Sources	Masking	Service
IIR_DMA IIR_STAT	Input DMA complete Output DMA complete	N/A	ROC from IIRDMASTAT+ RTI instruction
	MAC IEEE floating point exceptions		ROC from IIRMACSTAT+ RTI instruction

Sources

The IIR module drives two interrupt signals, `IIR_DMA` for the DMA status and `IIR_STAT` for the MAC status. The IIR module generates interrupts as described in the following sections.

Window Complete

This interrupt is generated at the end of each channel when all the output samples are calculated corresponding to a window and updated index values are written back.

All Channels Complete

This interrupt is generated when all the channels are complete or when one iteration of time slots completes. The interrupt follows the access completion rule, where the interrupt is generated when all data are written back to System memory.

Chained DMA

For chained DMA, if the `PCI` bit is cleared (= 0), the DMA complete interrupt is generated only after the entire chained DMA access is complete. If the `PCI` bit is set (= 1), then a DMA interrupt is generated for each TCB.

MAC Status

A MAC status interrupt is generated under these conditions

- Multiplier result zero - Set if Multiplier result is zero
- Multiplier Result Infinity - Set if Multiplier result is Infinity
- Multiply Invalid - Set if Multiply operation is Invalid
- Adder result zero - Set if Adder result is zero
- Adder result infinity - Set if Adder result is infinity
- Adder invalid - Set if Addition is invalid

Service

When a DMA interrupt occurs, programs can find whether the input or output DMA interrupt occurred by reading the DMA status register (IIR_DMASTAT). The DMA interrupt status bits are sticky and are cleared when the DMA status register is read. When a MAC status interrupt occurs, programs can find this by reading the MAC status register (IIR_MACSTAT). The MAC interrupt status bits are sticky and are cleared by a read.

The status interrupt sources are derived from the IIR_MACSTAT register. If the status interrupt occurs as a result of the last set of MAC operations of a processing iteration corresponding to a particular channel, the interrupt is generated continuously and cannot be stopped, even after disabling the accelerator. The interrupt can only be stopped by another processing iteration that results in a non-zero or valid multiply/add result. However, in this situation it is difficult to isolate whether the interrupt corresponds to the previous processing iteration or that of the current one. This makes the use of status interrupts impractical.

An alternate way is to poll status bits of the IIR_MACSTAT register inside the DMA interrupt service routine. However, the behavior of the status bits, as described below, should be kept in mind. The status bits in the IIR_MACSTAT registers are sticky. Once a status bit is set, it gets cleared only when the IIR_MACSTAT register is read and the previous set of MAC operations resulted in a non-zero/valid output. Therefore, if the last set of MAC operations of a particular processing iteration results in a zero/non-valid output, the corresponding status bit won't be cleared, even after reading the IIR_MACSTAT register. To avoid a false indication in the next processing iteration, it is necessary to ensure that all the status bits are cleared after the current iteration finishes.

The solution is to read the IIR_MACSTAT register twice inside the DMA interrupt service routine. The first read is used to identify which status bits are set. The second read is used to discover if the status bit was set because of the last set of MAC operations. If the status bit was not set because of the last set of MAC operations, it provides a zero result.

Otherwise, the bit was set because of the last set of MAC operations. In that case, the status bit must be cleared by performing a simple dummy IIR processing iteration (biquads = 1 and window size = 1) by choosing the appropriate coefficients and input buffer and reading the IIR_MACSTAT register after the processing is complete.

Programming Model

The IIR supports up to 24 channels which are time division multiplexed (TDM). Each channel can have a maximum of 12 cascaded biquads. The window size for each channel is configurable using control registers. A window size of 1 corresponds to sample based operation and the maximum window size is 64.

The coefficients are initially stored in system memory and one TCB per channel is created in system memory with each channels' TCB pointing to the next channels'. The TCB also contains channel specific control registers, input data buffer parameters and output data buffer parameters.

NOTE: The TCB of the last channel should point to the TCB of first channel.

The total number of channels is configured using the `IIR_CTL1` register and DMA is enabled.

The procedure that the accelerator uses to process biquads is shown in the **Biquad Processing Program Flow** figure and described in the following procedure.

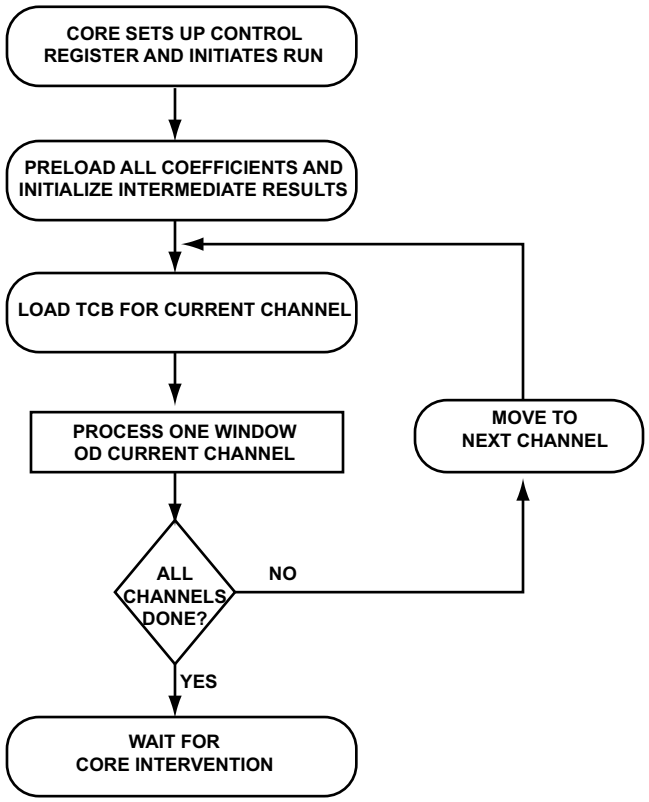

Figure 6: Biquad Processing Program Flow

1. The controller loads all coefficients of all the channels into local storage.
2. Once all the coefficients are loaded, the controller goes to the first biquad of the first channel and calculates the output of the first biquad and updates the intermediate results for that biquad.
3. Then, the accelerator moves to the next biquad of that channel and repeats the process until all the biquads for that channel are completed and the results are stored to memory.

4. This process is repeated with next sample until one window of the corresponding channel is processed.
5. After one window of the channel accelerator is processed, the accelerator moves to the next channel and computes the results.

All the addresses programmed in the TCB should correspond to 32-bit address boundaries and shouldn't contain the lower 2 bits (which are assumed to be zeros).

Dynamic Coefficient Processing Notes

The IIR accelerator loads the coefficients for all the channels only once when the IIR accelerator is enabled. In order to re-load the new coefficients, the accelerator has to be disabled and re-enabled.

Writing to Local Memory

1. Clear the `IIR_CTL1.DMAEN` bit.
2. Set the `IIR_DBG_CTL.EN`, `IIR_DBG_CTL.MEM` and `IIR_DBG_CTL.HLD` bits.
3. Set the `IIR_DBG_CTL.ADRINC` bit for address auto increment.
4. Write start address to the `IIR_DBG_ADDR` register. If bit 11 is set, coefficient memory is selected.
5. Wait at least 4 *CCLK* cycles.
6. Write data to the `IIR_DBG_WRDAT_LO` register.
7. Write data to the `IIR_DBG_WRDAT_HI` register.

Reading from Local Memory

1. Clear the `IIR_CTL1.DMAEN` bit.
2. Set the `IIR_DBG_CTL.EN`, `IIR_DBG_CTL.MEM` and `IIR_DBG_CTL.HLD` bits.
3. Set the `IIR_DBG_CTL.ADRINC` bit for address auto increment.
4. Write start address to the `IIR_DBG_ADDR` register. If bit 11 is set, coefficient memory is selected.
5. Wait at least 4 *CCLK* cycles.
6. Read data from the `IIR_DBG_RDDAT_LO` register.
7. Read data from the `IIR_DBG_RDDAT_HI` register.

Single Step Mode

Single step mode can be used for debug purposes. An additional debug register is used in this mode.

1. Enable stop DMA during breakpoint hit in the emulator settings.
2. Clear the IIR_DBG_CTL.HLD bit and enable the IIR_DBG_CTL.EN and IIR_DBG_CTL.RUN bits.
3. Program the IIR module according to the application.
4. In single step each iteration is updated in the emulator session.

Save Biquad State of the IIR

The following steps are required to resume IIR processing after being interrupted by another accelerator module.

1. When starting the accelerator for the first time, set the IIR_CTL1.EN, IIR_CTL1.DMAEN and IIR_CTL1.SS bits.
2. The core waits for the first set of IIR processing to conclude or performs some other task.
3. The accelerator writes back the updated DMA index registers and the updated Dk values after the processing completes.
4. Disable the accelerator by clearing the IIR_CTL1.EN bit. Optionally, clear the IIR_CTL1.DMAEN bit.
5. The core and accelerator wait for the next set of data to be ready. (The FIR/FFT accelerator can be used for a completely different purpose during this time.)
6. Once the next block is ready for processing, enable the IIR accelerator again by setting the IIR_CTL1.EN and IIR_CTL1.DMAEN bits. The coefficients and the Dk values are re-loaded back into the local memory.
7. The core waits for the current set of IIR processing to conclude or performs some other task.

Programming Example

In this example, an application needs IIR filtering for two channels of data; channel 1 has six biquads and channel 2 has eight biquads. The window size for all channels is 32.

1. Create a circular buffer in system memory for each channel's data. The buffer should be large enough to avoid overwriting data before it is processed by the accelerator.
2. Configure system memory buffers containing the 6×5 coefficients and the 6×2 Dk values for the channel 1 biquads, and the 8×5 coefficients and 8×2 Dk values of the channel 2 biquads.

3. Configure two TCBs in system memory with each channel's chain pointer entry pointing to the next channel's and the last channel's chain pointer entry pointing to the first in a circular fashion.
4. Program the IIR_CTL2 register to use channel 1 TCB for 6 biquads and a window size of 32, and channel 2 for 8 biquads and a window size of 32.
5. Configure the index, modifier, and length entries in the TCBs to point to the corresponding channel's data buffer, coefficient buffer and output data buffer.

   The location of the first channel's TCB is written to the chain pointer register in the accelerator.
6. Program the global control register IIR_CTL1.CH bit for 2 channels.
   a. The accelerator starts and loads the first channel's TCB, loads coefficients and Dk values of all the 6 biquads into local storage, then loads the TCB of the second channel, and finally loads coefficients and Dk values of all the 8 biquads.
   b. Once all the coefficients and Dk values are loaded, the controller loads the TCB of first channel and fetches the input sample. It then starts calculating the first biquad of the first channel.
   c. The accelerator calculates the output of the first biquad and then updates the intermediate results for that biquad. Then it moves to the next biquad of that channel and repeats the biquad processing until all the biquads for that channel are done and the final result is stored to memory.
   d. The accelerator repeats this process with next sample until one window of the corresponding channel is processed. Once the window is done, the accelerator saves the index values to memory and moves to the next channel. After both channels are done, the accelerator waits for core intervention.

ADSP-SC58x IIR Register Descriptions

IIR Register Definitions (IIR) contains the following registers.

Table 6: ADSP-SC58x IIR Register List

Name	Description
IIR_CHNPTR	Chain Pointer Register
IIR_COEFIDX	Coefficient Buffer Index Register
IIR_COEFLEN	Coefficient Buffer Length Register
IIR_COEFMOD	Coefficient Index Modifier Register
IIR_CTL1	Global Control Register
IIR_CTL2	Channel Control Register

Table 6: ADSP-SC58x IIR Register List (Continued)

Name	Description
IIR_DBG_ADDR	IIR Debug Address
IIR_DBG_CTL	IIR Debug Control
IIR_DBG_RDDAT_HI	IIR Debug Read Data High
IIR_DBG_RDDAT_LO	IIR Debug Read Data Low
IIR_DBG_WRDAT_HI	IIR Debug Write Data High
IIR_DBG_WRDAT_LO	IIR Debug Write Data Low
IIR_DMASTAT	DMA Status
IIR_INBASE	Input Buffer Base Register
IIR_INIDX	Input Data Index Register
IIR_INLEN	Input Data Buffer Length Register
IIR_INMOD	Input Data Index Modifier Register
IIR_MACSTAT	MAC Status Register
IIR_OUTBASE	Output Buffer Base Register
IIR_OUTIDX	Output Data Buffer Index Register
IIR_OUTLEN	IIR Output Data Buffer Length
IIR_OUTMOD	IIR Output Data Index Modifier

Chain Pointer Register

The IIR_CHNPTR register should be written with word address without the lower 2 bits.

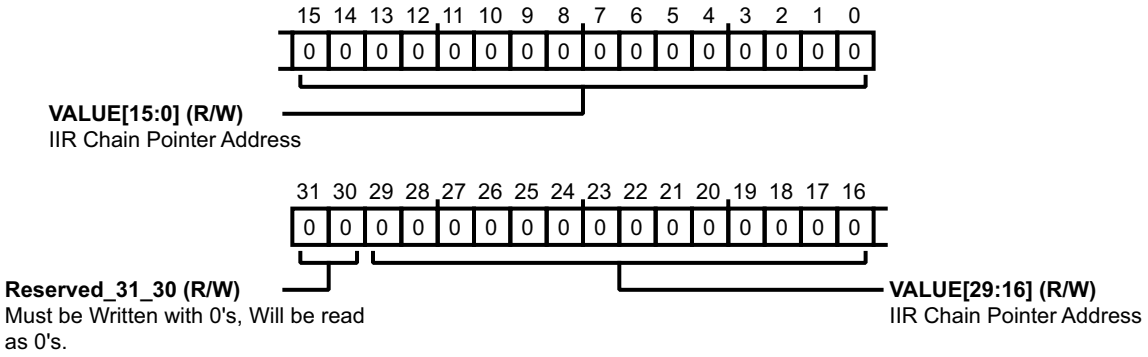

Figure 7: IIR_CHNPTR Register Diagram

Table 7: IIR_CHNPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:30 (R/W)	RESERVED_31_30	Must be Written with 0's, Will be read as 0's..
29:0 (R/W)	VALUE	IIR Chain Pointer Address. The `IIR_CHNPTR.VALUE` bit field contains the chain pointer address.

Coefficient Buffer Index Register

The `IIR_COEFIDX` register should be written with word address without the lower 2 bits

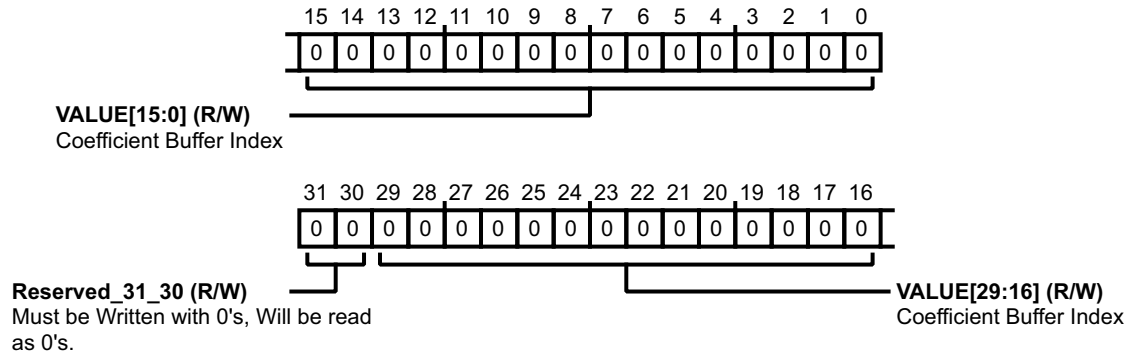

Figure 8: IIR_COEFIDX Register Diagram

Table 8: IIR_COEFIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:30 (R/W)	RESERVED_31_30	Must be Written with 0's, Will be read as 0's..
29:0 (R/W)	VALUE	Coefficient Buffer Index. The `IIR_COEFIDX.VALUE` bit field provides the coefficient buffer index.

Coefficient Buffer Length Register

The `IIR_COEFLEN` register provides the coefficient buffer length.

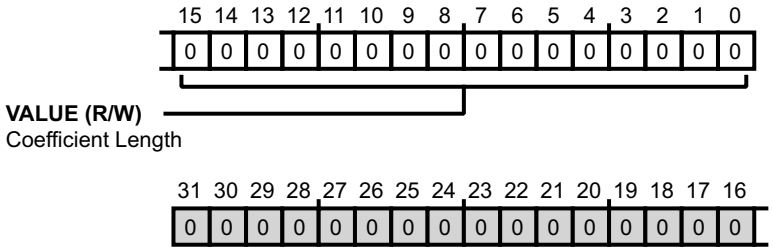

Figure 9: IIR_COEFLEN Register Diagram

Table 9: IIR_COEFLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Coefficient Length. The `IIR_COEFLEN.VALUE` bit field provides the coefficient buffer length.

Coefficient Index Modifier Register

The `IIR_COEFMOD` register provides the coefficient index modifier.

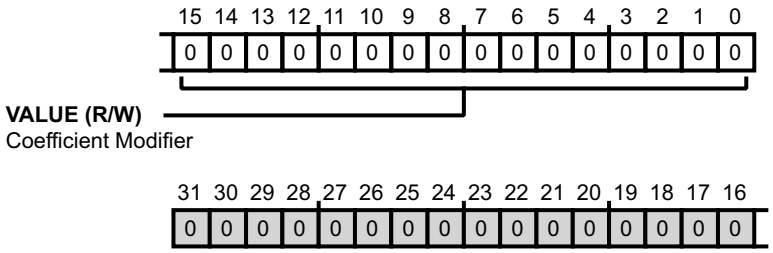

Figure 10: IIR_COEFMOD Register Diagram

Table 10: IIR_COEFMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Coefficient Modifier. The IIR_COEFMOD.VALUE bit field provides the coefficient modifier.

Global Control Register

The IIR_CTL1 register is used to configure the global parameters for the accelerator. These include number of channels, channel auto iterate, DMA enable, and accelerator enable.

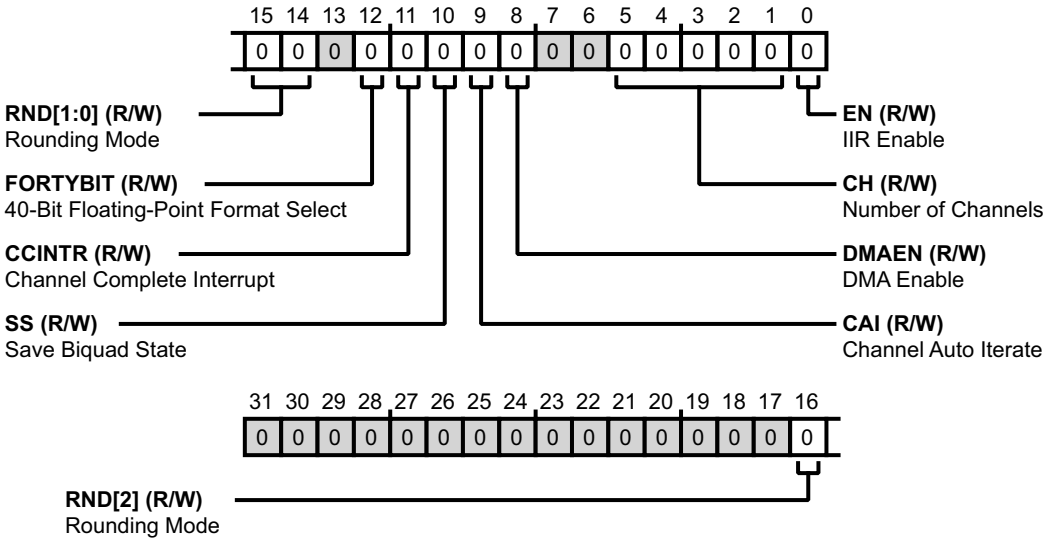

Figure 11: IIR_CTL1 Register Diagram

Table 11: IIR_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16:14 (R/W)	RND	Rounding Mode. The IIR_CTL1.RND bit field selects the rounding mode for floating-point format.	
		0	IEEE round to nearest (even)
		1	IEEE round to zero 010 = IEEE round to +ve infinity 011 = IEEE round to -ve infinity 100 = Round to nearest Up 101 = Round away from zero 110 = Reserved 111 = Reserved

Table 11: IIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	FORTYBIT	40-Bit Floating-Point Format Select. The IIR_CTL1.FORTYBIT bit selects between 32-bit IEEE floating-point format or 40-bit IEEE floating-point format.	
		0	32-bit IEEE floating-point
		1	40-bit IEEE floating-point
11 (R/W)	CCINTR	Channel Complete Interrupt. The IIR_CTL1.CCINTR bit configures the channel complete interrupt to generate when all channels are done or after each channel is done.	
		0	Interrupt is generated only when all channels are done (default)
		1	Interrupt is generated after each channels is done (default)
10 (R/W)	SS	Save Biquad State. The IIR_CTL1.SS bit configures the accelerator to store the Dk register settings into the internal memory. This can be used to save the biquad states before switching to another high priority accelerator task.	
9 (R/W)	CAI	Channel Auto Iterate. The IIR_CTL1.CAI bit sets whether TDM processing stops (idle) once all channels complete processing or moves to first channel and continues TDM processing in a loop when all channels complete processing.	
		0	TDM processing stops (idle) once all channels complete processing
		1	Moves to first channel and continues TDM processing in a loop when all channels complete processing
8 (R/W)	DMAEN	DMA Enable. The IIR_CTL1.DMAEN bit enables DMA on the accelerator.	
		0	Disable
		1	Enable
5:1 (R/W)	CH	Number of Channels. The IIR_CTL1.CH bit field configures the number of channels and is programmable between 0-23 (channels = NCH + 1).	
0 (R/W)	EN	IIR Enable. The IIR_CTL1.EN bit enables or disables the IIR accelerator.	
		0	IIR disabled
		1	IIR enabled

Channel Control Register

The IIR_CTL2 register is used to configure the channel specific parameters. These include the number of biquads and window size.

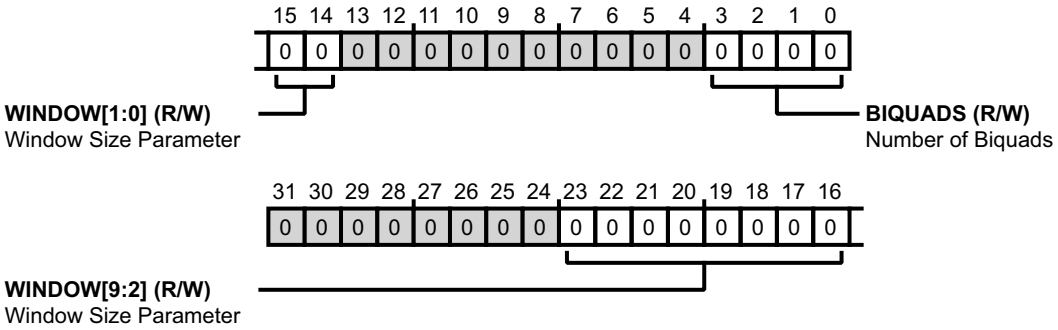

Figure 12: IIR_CTL2 Register Diagram

Table 12: IIR_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:14 (R/W)	WINDOW	Window Size Parameter. The IIR_CTL2.WINDOW bit field configures the window size which specifies the number of samples/blocks to process (sample based processing = window size of 1).
3:0 (R/W)	BIQUADS	Number of Biquads. The IIR_CTL2.BIQUADS bit field configures the number of biquads and is programmable between 0-11 (number of Biquads = BIQUADS + 1).

IIR Debug Address

The IIR_DBG_ADDR register holds the debug address. If bit 11 is set, coefficient memory is selected.

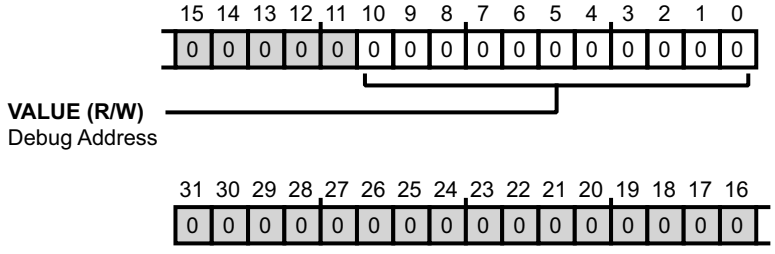

Figure 13: IIR_DBG_ADDR Register Diagram

Table 13: IIR_DBG_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	VALUE	Debug Address. The IIR_DBG_ADDR.VALUE bit field holds the debug address.

IIR Debug Control

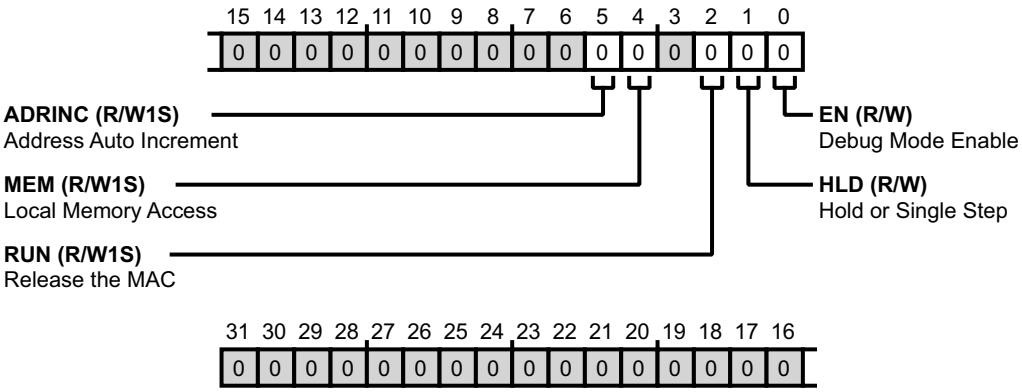

Figure 14: IIR_DBG_CTL Register Diagram

Table 14: IIR_DBG_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/W1S)	ADRINC	Address Auto Increment. The IIR_DBG_CTL.ADRINC bit allows the address register to auto increment on IIR_DBG_WRDAT_HI/IIR_DBG_WRDAT_LO writes and IIR_DBG_RDDAT_HI/IIR_DBG_RDDAT_LO reads.
4 (R/W1S)	MEM	Local Memory Access. The IIR_DBG_CTL.MEM bit allows the data and coefficients memory to be indirectly accessed.
2 (R/W1S)	RUN	Release the MAC. The IIR_DBG_CTL.RUN bit releases the MAC and is is self clearing after one IIR clock cycle.

Table 14: IIR_DBG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	HLD	Hold or Single Step. The IIR_DBG_CTL.HLD bit function is based on the IIR_DBG_CTL.MEM bit setting. For IIR_DBG_CTL.MEM = 0 this bit sets single step. For IIR_DBG_CTL.MEM = 1 this bit sets hold data.	
		0	No effect
		1	Single step (IIR_DBGMEM=0) or Hold data (IIR_DBGMEM=1)
0 (R/W)	EN	Debug Mode Enable. The IIR_DBG_CTL.EN bit enables debug mode. For local memory access, the IIR_CTL1 register can be cleared.	
		0	Disable
		1	Enable

IIR Debug Read Data High

The IIR_DBG_RDDAT_HI register is part of the 40-bit wide debug mode read data register and holds the upper 8 bits.

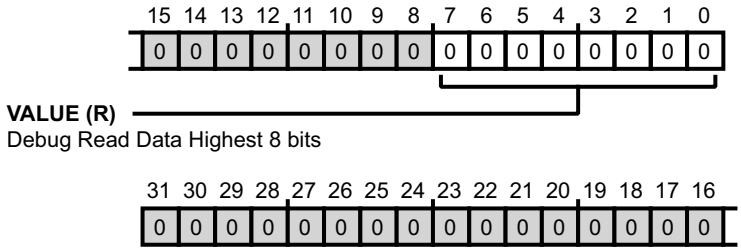

Figure 15: IIR_DBG_RDDAT_HI Register Diagram

Table 15: IIR_DBG_RDDAT_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VALUE	Debug Read Data Highest 8 bits. The IIR_DBG_RDDAT_HI.VALUE bit field holds the upper 8-bit read data.

IIR Debug Read Data Low

The IIR_DBG_RDDAT_LO register is part of the 40-bit wide debug mode read data register and holds the lower 32 bits.

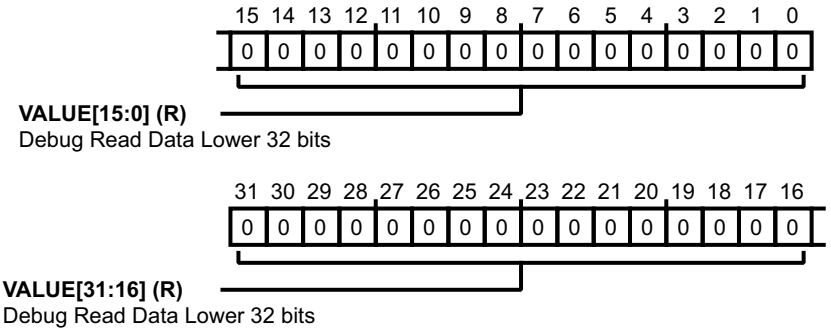

Figure 16: IIR_DBG_RDDAT_LO Register Diagram

Table 16: IIR_DBG_RDDAT_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Debug Read Data Lower 32 bits. The IIR_DBG_RDDAT_LO.VALUE bit field holds the lower 32-bit read data.

IIR Debug Write Data High

The IIR_DBG_WRDAT_HI register is part of the 40-bit wide debug mode write data register and holds the upper 8 bits.

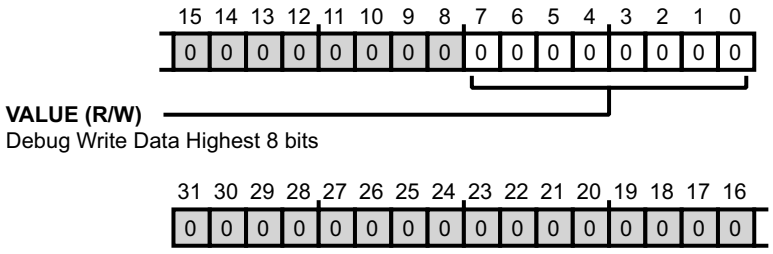

Figure 17: IIR_DBG_WRDAT_HI Register Diagram

Table 17: IIR_DBG_WRDAT_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VALUE	Debug Write Data Highest 8 bits. The IIR_DBG_WRDAT_HI.VALUE bit field holds the upper 8-bit write data.

IIR Debug Write Data Low

The IIR_DBG_WRDAT_LO register is part of the 40-bit wide debug mode write data register and holds the lower 32 bits.

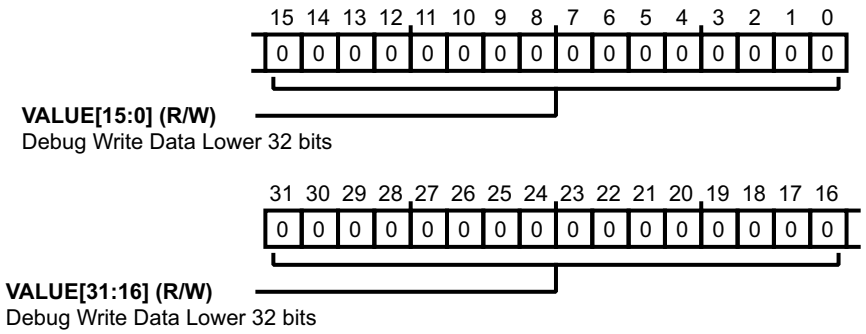

Figure 18: IIR_DBG_WRDAT_LO Register Diagram

Table 18: IIR_DBG_WRDAT_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0	VALUE	Debug Write Data Lower 32 bits.
(R/W)		The IIR_DBG_WRDAT_LO.VALUE bit field holds the lower 32-bit write data.

DMA Status

The IIR_DMASTAT registers indicate the status of DMA operations.

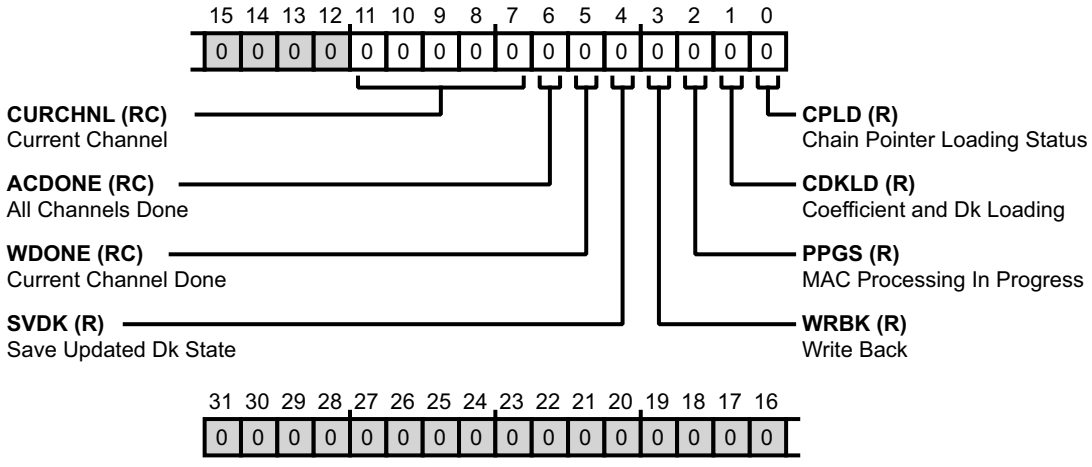

Figure 19: IIR_DMASTAT Register Diagram

Table 19: IIR_DMASTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11:7 (RC/NW)	CURCHNL	Current Channel. The IIR_DMASTAT.CURCHNL bit field indicates the channel that is being processed in the TDM slot. Zero indicates the last slot.	
6 (RC/NW)	ACDONE	All Channels Done. The IIR_DMASTAT.ACDONE bit indicates all channels are done processing. Note that the IIR_CTL1.CCINTR bit does not affect this status bit. This bit is sticky and is cleared on register read.	
5 (RC/NW)	WDONE	Current Channel Done. The IIR_DMASTAT.WDONE bit indicates the processing of the current channel is complete. Note that the IIR_CTL1.CCINTR bit does not affect this status bit. This bit is sticky and is cleared on register read.	
4 (R/NW)	SVDK	Save Updated Dk State. If there is more than one channel (IIR_CTL1.CH>0), the IIR_DMASTAT.SVDK bit toggles between 0 and 1 as it starts and completes the save state operation on one channel at a time. Therefore, this bit is not a reliable indicator of completion of the save state operation for all channels. To ensure graceful completion of the save state operation, programs must poll both the IIR_DMASTAT.CPLD and IIR_DMASTAT.SVDK bits and ensure (IIR_DMASTAT.CPLD OR IIR_DMASTAT.SVDK) = 0 after the IIR_DMASTAT.ACDONE bit is set. The recommended method for minimizing core intervention is to configure the accelerator to generate an interrupt when the processing of all the channels is complete (the IIR_CTL1.CCINTR bit is set), then poll to ensure (IIR_DMASTAT.CPLD OR IIR_DMASTAT.SVDK) = 0 inside the interrupt service routine. To minimize the interrupt service time, the core can perform unrelated tasks before it starts polling for save state operation completion.	
3 (R/NW)	WRBK	Write Back. The IIR_DMASTAT.WRBK bit indicates the accelerator is writing back updated index registers.	
2 (R/NW)	PPGS	MAC Processing In Progress. The IIR_DMASTAT.PPGS bit indicates MAC processing is in progress.	
1 (R/NW)	CDKLD	Coefficient and Dk Loading. The IIR_DMASTAT.CDKLD bit indicates the coefficient and Dk are loading.	
0 (R/NW)	CPLD	Chain Pointer Loading Status. The IIR_DMASTAT.CPLD bit indicates the IIR is in the chain pointer load state.	
		0	state machine not in chain pointer load state
		1	state machine in chain pointer load state

Input Buffer Base Register

The IIR_INBASE register should be written with word address without the lower 2 bits

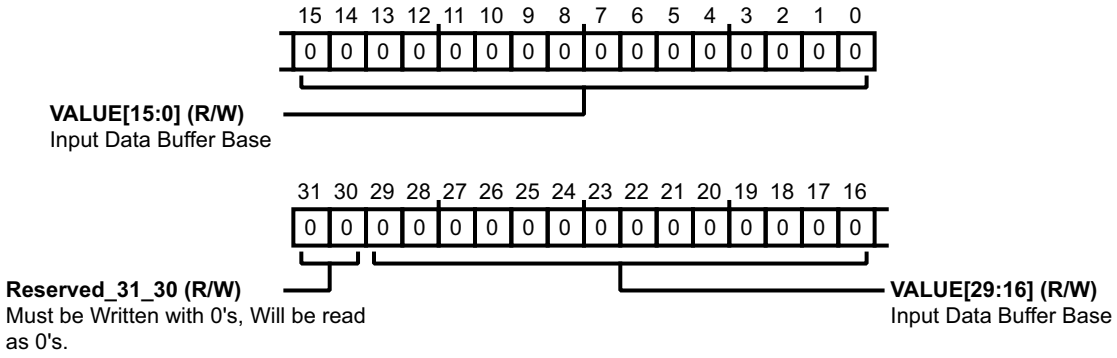

Figure 20: IIR_INBASE Register Diagram

Table 20: IIR_INBASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:30 (R/W)	RESERVED_31_30	Must be Written with 0's, Will be read as 0's..
29:0 (R/W)	VALUE	Input Data Buffer Base. The IIR_INBASE.VALUE bit field value is the input data buffer base address.

Input Data Index Register

The IIR_INIDX register should be written with a word address without the lower 2 bits

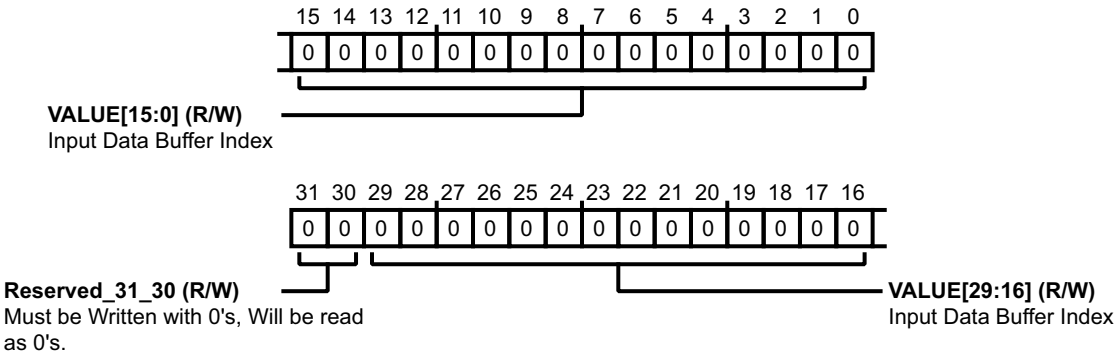

Figure 21: IIR_INIDX Register Diagram

Table 21: IIR_INIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:30 (R/W)	RESERVED_31_30	Must be Written with 0's, Will be read as 0's..
29:0 (R/W)	VALUE	Input Data Buffer Index. The `IIR_INIDX.VALUE` bit field value is the input data buffer index.

Input Data Buffer Length Register

The `IIR_INLEN` register provides the input data buffer length.

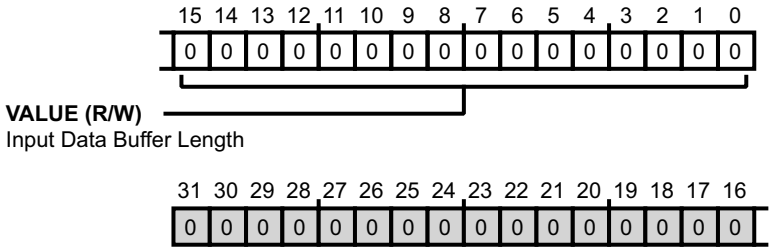

Figure 22: IIR_INLEN Register Diagram

Table 22: IIR_INLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Input Data Buffer Length. The `IIR_INLEN.VALUE` bit field value is the input data buffer length.

Input Data Index Modifier Register

The `IIR_INMOD` register provides the 16-bit input data buffer index modifier.

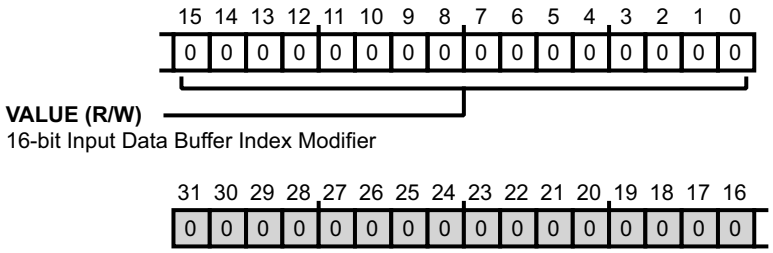

Figure 23: IIR_INMOD Register Diagram

Table 23: IIR_INMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Input Data Buffer Index Modifier. The IIR_INMOD.VALUE bit field value is the 16-bit input data buffer modifier.

MAC Status Register

The IIR_MACSTAT register indicates the status of MAC operations.

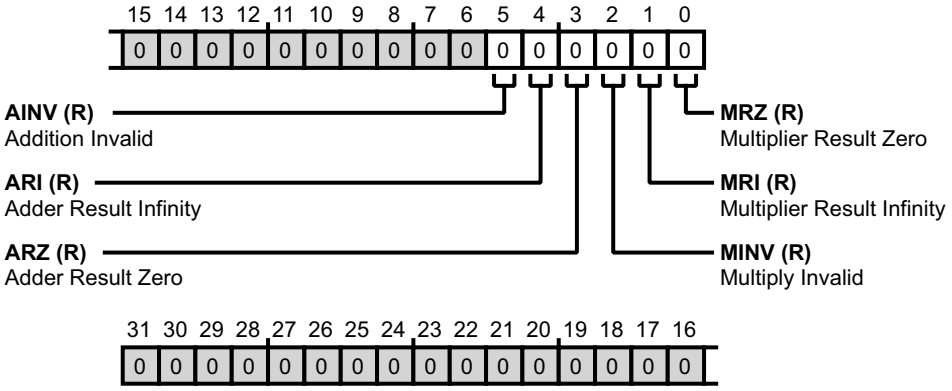

Figure 24: IIR_MACSTAT Register Diagram

Table 24: IIR_MACSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/NW)	AINV	Addition Invalid. The IIR_MACSTAT.AINV bit indicates the addition is invalid.
4 (R/NW)	ARI	Adder Result Infinity. The IIR_MACSTAT.ARI bit indicates the adder result is infinity.
3 (R/NW)	ARZ	Adder Result Zero. The IIR_MACSTAT.ARZ bit indicates the adder result is zero.
2 (R/NW)	MINV	Multiply Invalid. The IIR_MACSTAT.MINV bit indicates the multiply operation is invalid.
1 (R/NW)	MRI	Multiplier Result Infinity. The IIR_MACSTAT.MRI bit indicates the multiplier result is infinity.
0 (R/NW)	MRZ	Multiplier Result Zero. The IIR_MACSTAT.MRZ bit indicates the multiplier result is zero.

Output Buffer Base Register

The IIR_OUTBASE register should be written with word address without the lower 2 bits

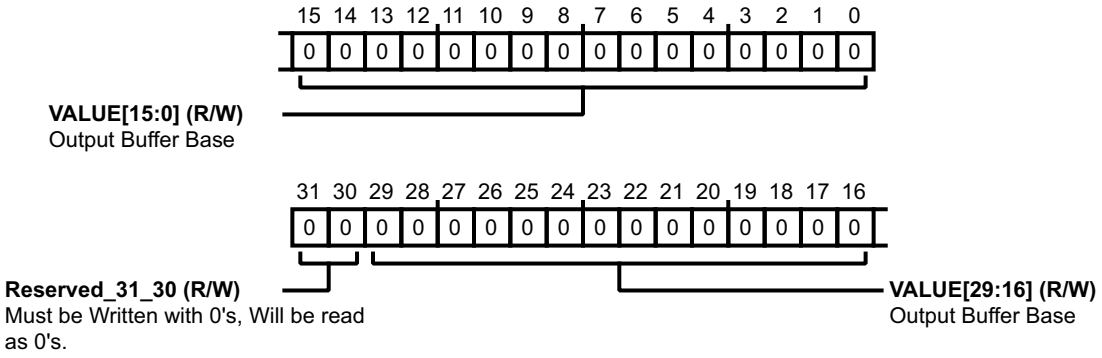

Figure 25: IIR_OUTBASE Register Diagram

Table 25: IIR_OUTBASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:30 (R/W)	RESERVED_31_30	Must be Written with 0's, Will be read as 0's..
29:0 (R/W)	VALUE	Output Buffer Base. The IIR_OUTBASE.VALUE bit field provides the output buffer base address.

Output Data Buffer Index Register

The IIR_OUTIDX register should be written with word address without the lower 2 bits

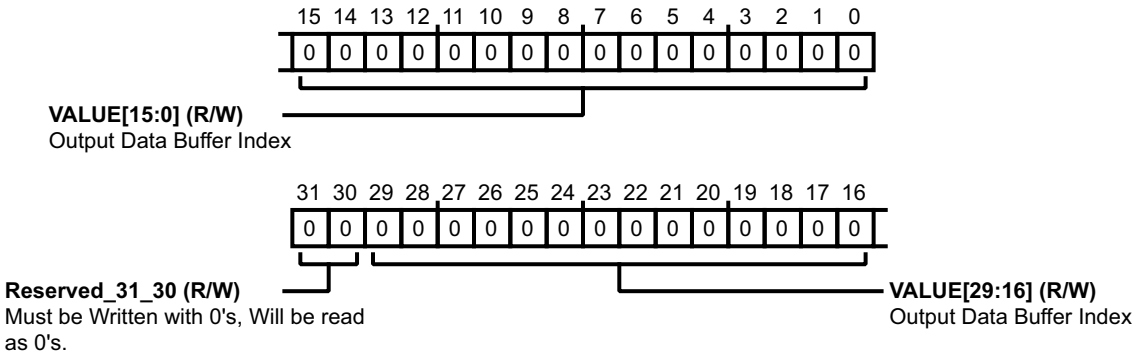

Figure 26: IIR_OUTIDX Register Diagram

Table 26: IIR_OUTIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:30 (R/W)	RESERVED_31_30	Must be Written with 0's, Will be read as 0's..
29:0 (R/W)	VALUE	Output Data Buffer Index. The `IIR_OUTIDX.VALUE` bit field provides the output data buffer index.

IIR Output Data Buffer Length

The `IIR_OUTLEN` register provides the output data buffer length.

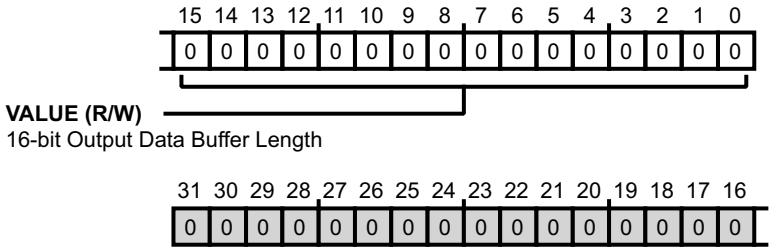

Figure 27: IIR_OUTLEN Register Diagram

Table 27: IIR_OUTLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Output Data Buffer Length. The `IIR_OUTLEN.VALUE` bit field provides the output data buffer length.

IIR Output Data Index Modifier

The `IIR_OUTMOD` register provides the output data index modifier.

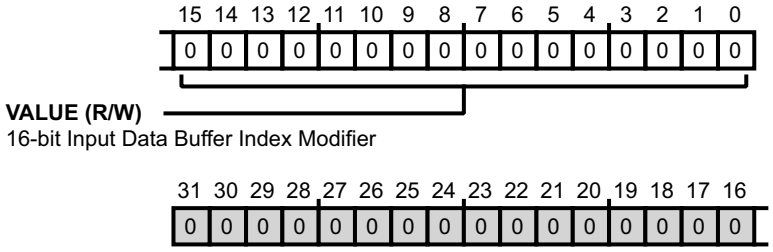

Figure 28: IIR_OUTMOD Register Diagram

Table 28: IIR_OUTMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Input Data Buffer Index Modifier. The `IIR_OUTMOD.VALUE` bit field provides the output data buffer index modifier.

52 Reset Control Unit (RCU)

Reset is the initial state of the whole processor (or one of the cores). It is the result of a hardware or software triggered event. In this state, all control registers are set to their default values and functional units are idle. Exiting a full system reset starts with Core-0 only being ready to boot. Exiting a Core n only reset starts with this Core n being ready to boot.

The reset control unit (RCU) controls how all the functional units enter and exit reset. Differences in functional requirements and clocking constraints define how reset signals are generated. Programs must guarantee that none of the reset functions puts the system into an undefined state or causes resources to stall. This functionality is important when only one of the cores is reset (programs must ensure that there is no pending system activity involving the core that is being reset).

RCU Features

The RCU module supports the following features:

- Hardware reset through the $\overline{\text{SYS\_HWRST}}$ pin
- Software system reset through RCU registers
- Hardware system reset through:
  - TRU module
  - SEC module
  - the oscillator watchdog module of the CGU
- Core reset through RCU registers

RCU Functional Description

This section provides information on the function of RCU module.

Hardware reset using SYS_HWRST pin

Asserting the $\overline{\text{SYS\_HWRST}}$ pin resets all functional units, except the RTC (if present)

Hardware reset through RCU

RCU can perform a full system reset which can be initiated through hardware blocks like the SEC, the TRU, and the oscillator watchdog

Software reset using RCU registers

Asserting the SYS_HWRST bit resets all system units

Core reset RCU registers

Core n can be individually reset by software, or by setting any of CRn (15 ≥ n ≥ 0) bits in the RCU_CRCTL register

ADSP-SC58x RCU Register List

The Reset Control Unit (RCU) controls how all the functional units in the processor enter and exit Reset. Differences in functional requirements and clocking constraints exist (units in different clock domains have to enter reset asynchronously, but units exit reset in a deterministic way), and these differences define how reset signals are generated. Reset signals propagate through all functional units asynchronously. For more information on RCU functionality, see the RCU register descriptions.

Table 1: ADSP-SC58x RCU Register List

Name	Description
RCU_BCODE	Boot Code Register
RCU_CRCTL	Core Reset outputs Control Register
RCU_CRSTAT	Core Reset outputs Status Register
RCU_CTL	Control Register
RCU_MSG	Message Register
RCU_MSG_CLR	Message Clear Bits Register
RCU_MSG_SET	Message Set Bits Register
RCU_REVID	Revision ID Register
RCU_SIDIS	System Interface Disable Register
RCU_SISTAT	System Interface Status Register
RCU_STAT	Status Register
RCU_SVECT0	Software Vector Register 0
RCU_SVECT1	Software Vector Register 1
RCU_SVECT2	Software Vector Register 2
RCU_SVECT_LCK	SVECT Lock Register