Radar Hardware Accelerator

The *Radar Hardware Accelerator User's Guide* (in two parts) describes the Radar Hardware Accelerator architecture, features, and operation of various blocks and their register descriptions. The purpose is to enable the user to understand the capabilities offered by the Radar Hardware Accelerator and to program it appropriately to achieve the desired functionality.

This user's guide is divided into two parts. The first part provides an overview of the overall architecture and features available in the Radar Hardware Accelerator. The main features, such as, windowing, FFT, and log-magnitude are covered in this part.

The second part of the user's guide covers additional features like CFAR-CA and other advanced usage possibilities. The second part of the user's guide is optional and can be skipped if the user is interested only in the FFT computation capability.

Contents

List of Figures

List of Tables

Trademarks

ARM, Cortex are registered trademarks of Arm Limited.
All other trademarks are the property of their respective owners.

1 Radar Hardware Accelerator – Part 1

1.1 Radar Hardware Accelerator – Overview

This section provides an overview of the Radar Hardware Accelerator. The section covers the key features of the accelerator and overall architecture.

1.1.1 Introduction

The Radar Hardware Accelerator (HWA) is a hardware IP that enables off-loading the burden of certain frequently used computations in FMCW radar signal processing from the main processor. It is well known that FMCW radar signal processing involves the use of FFT and log-magnitude computations to obtain a radar image across the range, velocity, and angle dimensions. Some of the frequently used functions in FMCW radar signal processing can be done within the Radar Hardware Accelerator, while still retaining the flexibility of implementing other proprietary algorithms in the main processor.

1.1.2 High Level Architecture

The Radar Hardware Accelerator module is categorized into HWA v1.0 and HWA v1.05 versions. The HWA v1.0 is the original version. HWA v1.05 has both bug fixes and additional safety enhancements. Table 1 summarizes the features relevant to each HWA versions. For complete bug fixes list, see Appendix A. The Hardware Accelerator register map is listed in Appendix B and the Safety feature that is only applicable to HWA v1.05 is discussed in Appendix C. For the HWA version associated with each of the mmWave devices, see the Technical Reference Manual.

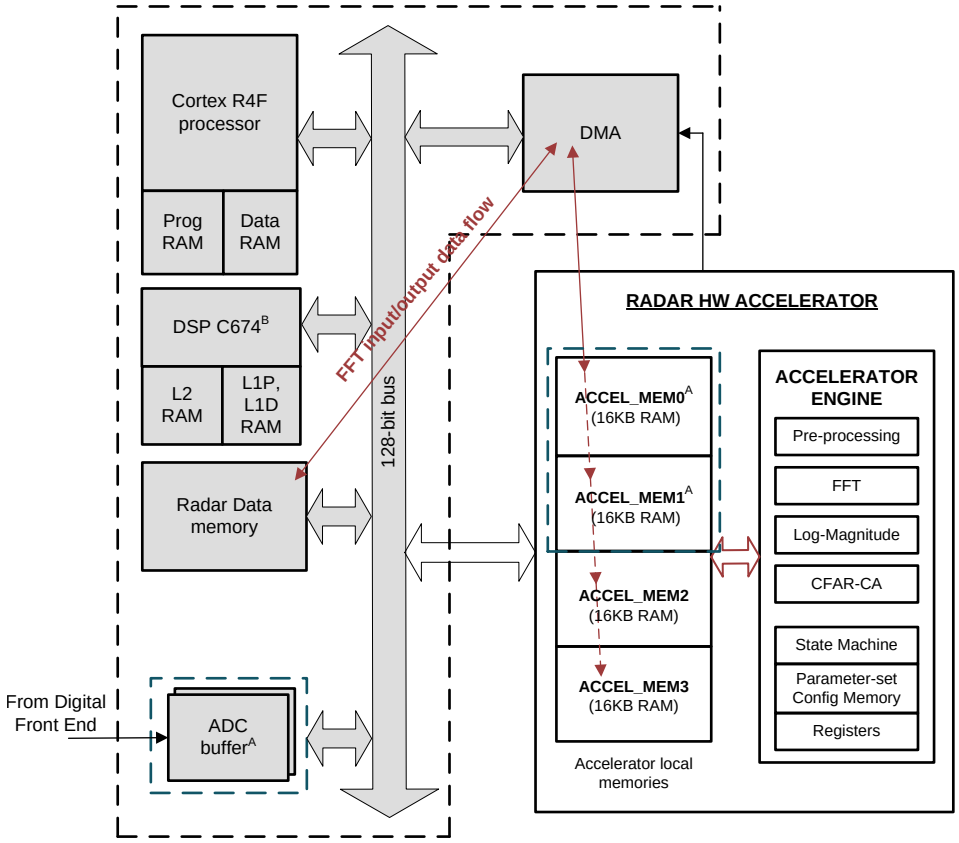
Table 1. Summary of Feature vs HWA version(1)

Section Index	Sections/Components	HWA Version		Comments
		1.0	1.05	
Appendix A	HWA Errata	√	√	Check the errata list for the applicable version
Appendix C	Safety Feature		√	Only applicable for version marked √ here

(1) Sections that are not listed in the table applies to both HWA versions

The HWA module is loosely coupled to the main processor(s). Some device variants have an ARM® Cortex®-R4F as the main processor, whereas other device variants have two main processors, namely a C674 DSP and an ARM® Cortex®-R4F. The Radar Hardware Accelerator module is connected to a 128-bit bus that is present in the main processor system as shown in Figure 1.

The Radar Hardware Accelerator module comprises an accelerator engine and four memories, each of 16KB size, which are used to send input data to and pull output data from the accelerator engine. These memories are referred to as *local memories* of the Radar Accelerator (ACCEL_MEM). For convenience, these four local memories are referred to as ACCEL_MEM0, ACCEL_MEM1, ACCEL_MEM2, and ACCEL_MEM3.



A For HWA v1.0, the analog-to-digital converter (ADC) (ping and pong) buffers are 16KB each. In this device, the ACCEL_MEM0 and the ACCEL_MEM1 memories are shared with the ADC buffers. For HWA v1.05, in addition to the ADC buffer sharing mode, dedicated memories are also available for ACCEL_MEM0 and ACCEL_MEM1.

B DSP C674 is available only in certain device variants.

Figure 1. Radar Hardware Accelerator

1.1.2.1 High-Level Data Flow

The typical data flow is that the DMA module is used to bring samples (for example, FFT input samples) into the local memories of the Radar Hardware Accelerator, so that the main accelerator engine can access and process these samples. Once the accelerator processing is done, the DMA module reads the output samples from the local memories of the Radar Hardware Accelerator and stores them back in the Radar data memory for further processing by the main processor. In Figure 1, the red arrow shows data movement from the Radar data memory into the accelerator local memories for the FFT and other processing steps. The red arrow also shows the output samples from the accelerator being picked up by the DMA and written back into the Radar data memory for further processing by the main processor.

In highly integrated mmWave devices, the Radar Hardware Accelerator is included as part of a single chip along with the mmWave RF and analog Front-end. For HWA v1.0, two of the accelerator local memories, namely ACCEL_MEM0 and ACCEL_MEM1, are directly shared with the ping and pong ADC buffers (which are 16KB each) – such that the ADC output samples for first-dimension FFT processing are directly and immediately available to the Radar Hardware Accelerator at the end of each chirp, without needing a DMA transfer. After the first-dimension FFT processing is complete (typically, at the end of the active transmission of chirps in a frame), it is possible to freely use these memories for (see the FFT1DEN register bit) second-dimension FFT processing by bringing in data to these memories through DMA transfer.

On the other hand, in the devices with HWA v1.05, in addition to the aforementioned ADC buffer sharing mode, there are dedicated memory instances available for ACCEL_MEM0 and ACCEL_MEM1. Note that in the devices with HWA v1.05, the ADC ping and pong buffers are 32KB each. Therefore, when using the ADC buffer sharing mode, the first half of the ADC buffer (the top 16KB) is shared with ACCEL_MEM0 and ACCEL_MEM1. In this mode, the ADC output samples are directly and immediately available to the Radar Hardware Accelerator at the end of each chirp, without needing a DMA transfer. Alternately, if the ADC output samples are required to be pre-processed by the DSP before processing by the Hardware Accelerator, then the ADC buffer sharing can be disabled (see the FFT1DEN register bit), such that the DSP can access the ADC buffer, perform the pre-processing as needed, and then write to the dedicated local memories (ACCEL_MEM0 and ACCEL_MEM1) of the Radar Hardware Accelerator for further processing using the accelerator.

The purpose behind the four separate local memories (16KB each) inside the Radar Hardware Accelerator is to enable the *ping-pong* mechanism, for both the input and output, such that the DMA write (and read) operations can happen in parallel to the main computational processing of the accelerator. The presence of four memories enables such parallelism. For example, the DMA can be configured to write FFT input samples (ping) into ACCEL_MEM0 and read FFT output samples (ping) from ACCEL_MEM2. At the same time, the accelerator engine can be working on FFT input samples (pong) from ACCEL_MEM1 and writing FFT output samples (pong) into ACCEL_MEM3. However, both the DMA and the accelerator cannot access the same 16KB memory at the same time. This would lead to an error (refer to the STATERRCODE register description in Table 4).

The Radar Hardware Accelerator operates on a single clock domain and the operating clock frequency is 200 MHz.

The accelerator local memories are 128-bits wide, for example, each of the 16KB banks is implemented as 1024 words of 128 bits each. This allows the DMA to bring data into the accelerator local memories efficiently (up to a maximum throughput of 128 bits per clock cycle, depending upon the DMA configuration).

It is important to note that any of the four local memories can be the *source* of the input samples to the accelerator engine and any of the four local memories can be the *destination* for the output samples from the accelerator engine – with the important restriction that the source and destination memories cannot be the same 16KB bank. Note also that the accelerator local memories do not necessarily need to be used in ping-pong mode and can instead be used as larger 32KB input and output memories, if the use case requires. The address space for the four 16KB memories is contiguous and thus the source and destination memory can effectively be larger than 16KB.

1.1.2.2 Configuration

The operations of the Radar Hardware Accelerator are configured using registers, which are of two types – *parameter sets* and *common* (common for all parameter sets) registers. The purpose of the parameter sets is to enable a complete sequence of various accelerator operations to be preprogrammed (with appropriate source and destination memory addresses and other configurations specified for each operation in that sequence), such that the accelerator can perform them one after the other, with minimal intervention from the main processor.

The parameter-set register configurations are programmed into a separate 512-byte *parameter-set configuration memory*. A state machine built into the accelerator handles the loading of one parameter-set configuration at a time and sequences the preprogrammed operations one after another. This process is further explained in later sections of this user's guide. Section 1.6 shows a use-case example, which illustrates this well.

1.1.3 Accelerator Engine Block Diagram

As previously mentioned, the Radar Hardware Accelerator module consists of four local memories of 16KB each (ACCEL_MEM) and the main accelerator engine. The accelerator engine has the following five components (as shown in Figure 2) – a state machine, input formatter block, output formatter block, core computational unit, and the 512-byte parameter-set configuration memory.

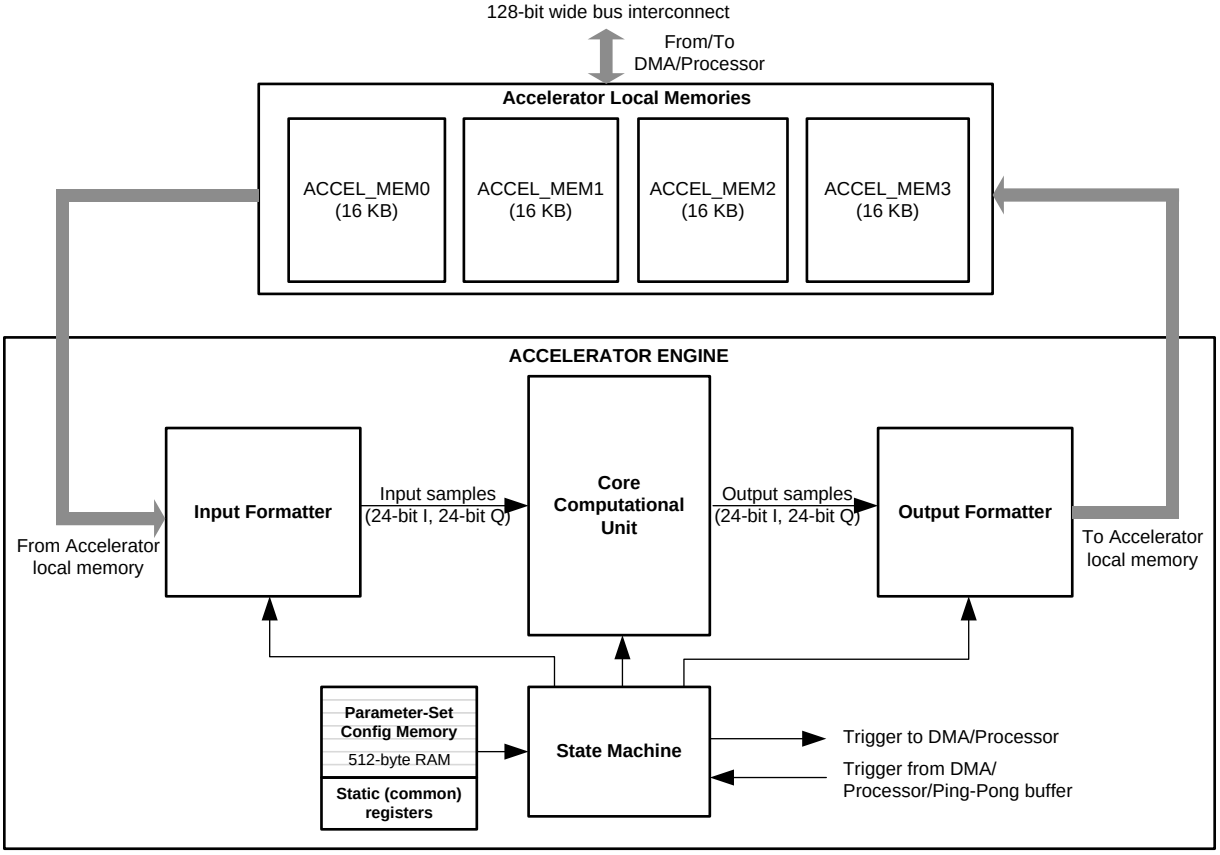


Figure 2. Accelerator Engine Block Diagram

The purpose of these components is as follows.

- State machine: the state machine is responsible for controlling the overall operation of the accelerator – specifically, the starting, looping, stopping, as well as triggering and handshake mechanisms between the accelerator, DMA, and main processor. The state machine is also closely connected to the parameter-set configuration memory and takes care of sequencing and chaining a sequence of multiple accelerator operations as programmed in the parameter-set configuration memory.

- Input formatter: the input formatter block is responsible for reading the input samples from any one of the local memories and feeding them into the core computational unit. In this process, this block provides flexible ways of accessing the input samples, in terms of 16-bit versus 32-bit aligned input samples, transpose read-out, flexible scaling, and sign extension to generate internal bit-width of 24 bits, and so on. Lastly the input formatter block provides 24-bit complex samples as input to the core computational unit. The local memory (memories) from which the input formatter reads the input samples is called the *source* memory.
- Output formatter: the output formatter block is responsible for writing the output samples from the core computational unit into the local memories. This block also provides flexible ways of formatting the output samples, in terms of 16-bit versus 32-bit aligned output samples, transpose write, flexible scaling from internal bit-width of 24 bits, to 16-bit or 32-bit aligned output samples, sign-extension, and so on. The local memory (memories) to which the output formatter writes the output samples is called the *destination* memory.
- Core computational unit: the core computational unit contains the main computational logic for various operations, such as windowing, FFT, magnitude, log2, and CFAR-CA calculations. The unit accepts a streaming input from the input formatter block (at the rate of one input sample per clock cycle), performs computations, and produces a streaming output to the output formatter block (typically at the rate of one output sample per clock cycle), with some initial latency depending on the nature of the computations involved.
- Parameter-set configuration memory: this is a 512-byte RAM that is used to preconfigure the sets of parameters (register settings) for a chained sequence of accelerator operations, which can then be executed by the state machine in a loop. This allows the accelerator to perform a preprogrammed sequence of operations in a loop without frequent intervention from the main processor.

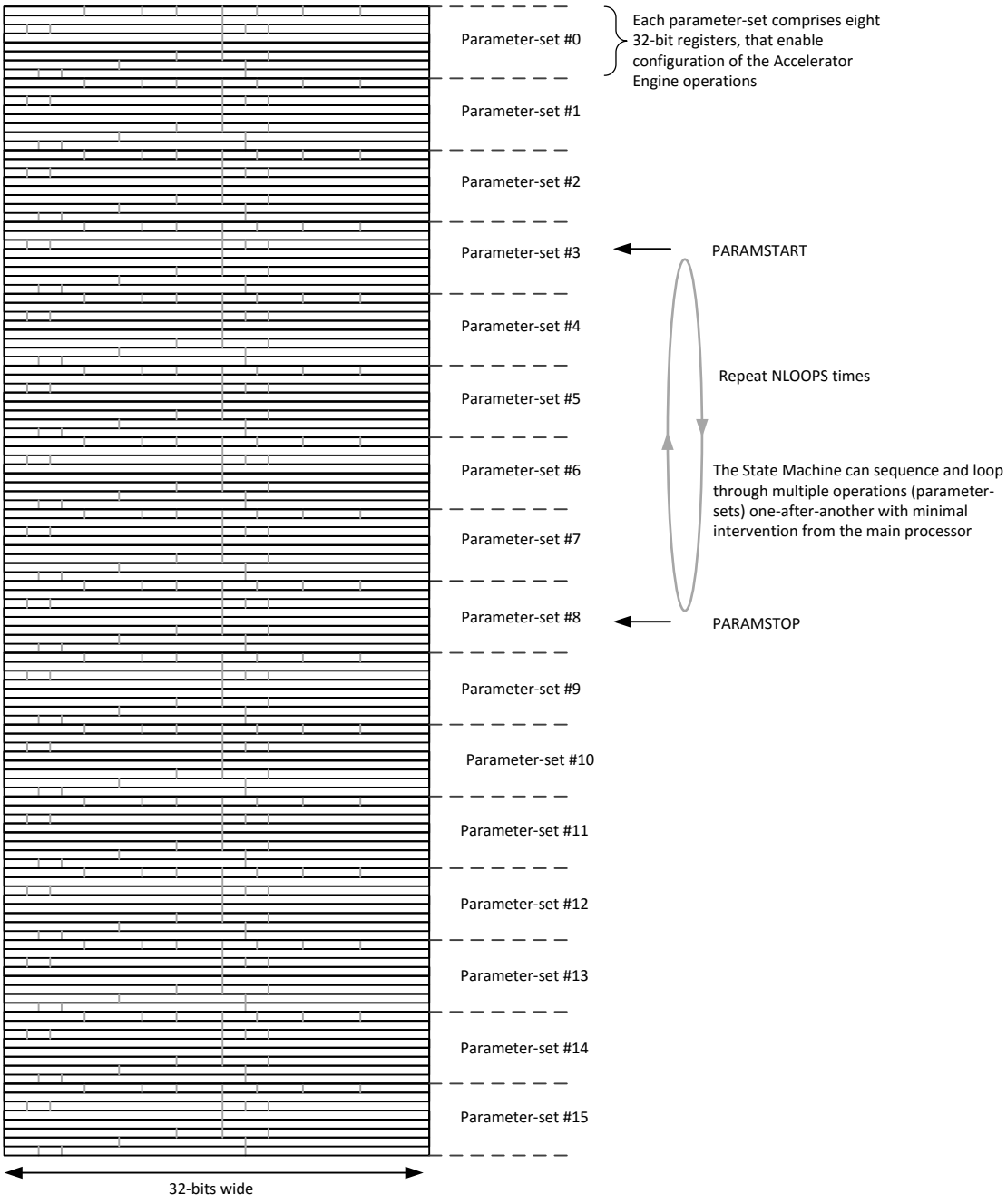


Figure 3. Parameter-Set Configuration Memory (512 Bytes)

The number of parameter sets that can be preconfigured and sequenced (chained) is 16. This means that up to 16 accelerator operations can be chained together and these can then be looped as well, with minimal intervention from the main processor. For example, operations like FFT, log-magnitude, and CFAR-CA detection can be preconfigured in the parameter-set configuration memory and the state machine can be made to sequence them one after another and run them in a loop for specified number of times. There is a provision available to interrupt the main processor and/or trigger a DMA channel at the end of each parameter set if required. This allows various ways by which the accelerator, DMA, and the main processor can work together to establish a data and processing flow. As shown in Figure 3, each parameter set contains the equivalent of eight 32-bit registers, which corresponds to total RAM size of 16 × 8 × 32 bits = 512 bytes for the parameter-set configuration memory.

The layout of the parameter-set register map is provided in Appendix B. The detailed descriptions of the registers is provided in the various sections, as and when the functionality of each component is presented.

1.1.4 Accelerator Engine Operation

The accelerator engine and the local memories run on a single clock domain. The overall operation of the accelerator can be summarized as follows. The accelerator engine is configured by the main processor through common configuration registers (common for all parameter sets), as well as the parameter-set configuration memory. As explained earlier, the former comprises common register settings for overall control of the accelerator engine, and the latter comprises the 16 parameter-set specific settings which control the functioning of the accelerator for each of its *chained* sequence of operations.

When the accelerator engine is enabled, the state machine kicks off and controls the overall operation of the accelerator, which involves loading the parameter sets one at a time from the parameter-set configuration memory into various internal registers of the accelerator engine and running the accelerator as per the programmed configuration for each parameter set one after another. The entire procedure then repeats in a loop for a programmed number of times (NLOOPS described later).

Each parameter set includes various configuration details such as the accelerator mode of operation (FFT, Log2, and so on), the source memory address, number of samples, the destination memory address, input formatting, output formatting, trigger mode for controlling the start of computations to ensure proper handshake with the DMA, and so on.

1.1.4.1 Data Throughput

Once the state machine has loaded the registers corresponding to the current parameter set to be executed, the data flow happens as follows: at each clock cycle, one sample from the source memory is read by the input formatter and fed into the core computational unit with appropriate scaling and formatting as configured. The data interface between the input formatter and the core computational unit is a 24-bit complex bus (24-bit for each I and Q) which streams one input sample every clock cycle. The core computational unit processes this streaming sequence of input samples and in general, produces a streaming output also at one sample every clock cycle, after an initial latency period. Thus for most operations (FFT, log-magnitude, CFAR-CA, and so on), in steady state the core computational unit maintains a streaming data rate of one sample per clock cycle. The data interface between the core computational unit and the output formatter is also a 24-bit complex bus (24-bit for each I and Q) and the output formatter is responsible for writing into the destination memory, with appropriate scaling and formatting as configured.

The next section provides more details regarding the state machine, including its detailed operation, registers, trigger mechanisms, and so on.

1.2 Accelerator Engine – State Machine

This section describes the state machine block present in the accelerator engine (see Figure 4). This block, together with the input formatter and output formatter blocks described in the next two sections, provides the overall framework for establishing the data flow and using the accelerator for various computations.

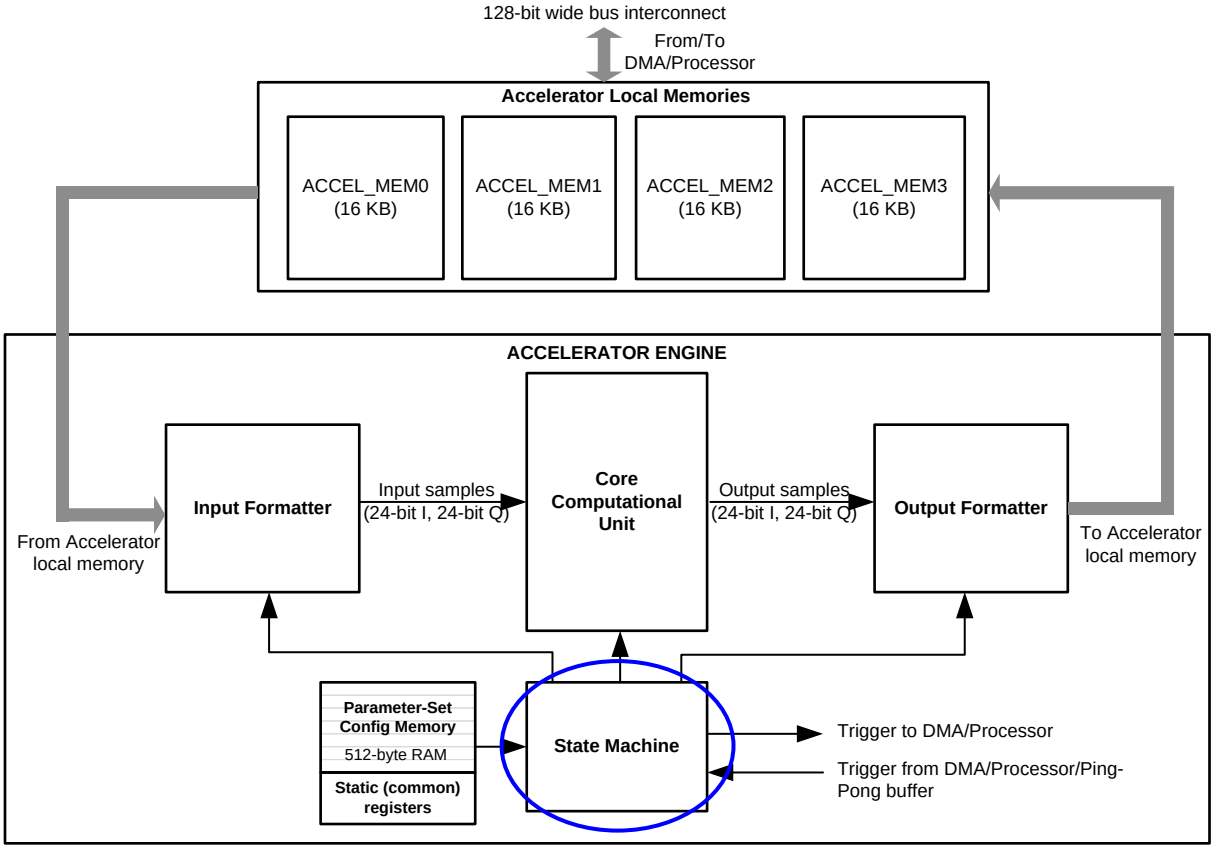


Figure 4. State Machine

1.2.1 State Machine

The state machine controls the overall functioning of the Radar Hardware Accelerator. The state machine controls the enabling and disabling of the accelerator, as well as supports sequencing an entire set of operations (configured using parameter-set configuration memory), and looping through those operations one after another without needing frequent intervention from the main processor.

1.2.1.1 State Machine – Operation

The state machine block and the entire accelerator remain in reset and disabled state by default. The state machine (and hence the accelerator in general) is enabled by setting the ACCCLKEN register bit, followed by writing 111b into the ACCENABLE register.

Note that a complete list of registers pertaining to the state machine is provided in Table 2. Some of the registers are common (common for all parameter sets) registers, whereas some other registers are parameter-set registers, which as explained in the previous section means that they can be uniquely programmed for each of the 16 parameter sets. For each register, Table 2 lists whether it is part of the parameter set or not. Table 2 also provides a brief description of each register.

When enabled, the state machine steps through (one after another) the parameter sets programmed in the parameter-set configuration memory and executes the computations as per the configuration of each parameter set. The registers PARAMSTART and PARAMSTOP define the starting index and ending index within the 16 parameter sets, so that only those parameter sets between the start and end indices are executed by the accelerator, as shown in Figure 3. The state machine also loops through these parameter sets for a total of NLOOPS times (unless NLOOPS is programmed as 0 or 4095, in which case the loop does not run or runs infinite times respectively). As an example, if the state machine needs to be configured to run the first four parameter sets in a loop 64 times, then the registers should be programmed as follows: PARAMSTART = 0, PARAMSTOP = 3, and NLOOPS = 64.

For each parameter set, there is a TRIGMODE register, which is used to control when the state machine starts executing the computations for that parameter set. This control is useful, for example, to ensure that the input data is ready in the accelerator local memory (source memory) before the computations are started. Specifically, it is possible to trigger the start of computations after completion of a DMA transfer, or, after a ping-pong switch happens in the ADC buffer, and so on. The TRIGMODE register setting thus controls when the accelerator operation is triggered for the current parameter set and there are four trigger mechanisms supported as listed in the next subsection. Once triggered, the state machine loads all the registers from the parameter-set configuration memory for the current parameter set into corresponding internal registers of the accelerator and starts the actual computations for that parameter set. After completion of computations of the current parameter set, it moves to the next parameter set.

After a sequence of operations as programmed in the parameter set(s) for the specified number of loops is complete, the accelerator provides a completion interrupt (ACC_DONE_INTR) to the processor. The accelerator can be reconfigured as desired. For reconfiguration, the following procedure must be followed. The accelerator must be disabled by writing 000b to the ACCENABLE register. Then, a reset must be asserted by writing 111b followed by 000b to the ACCRESET register. The new configurations can now be written in to the accelerator, and then the accelerator can be enabled again by writing 111b to ACCENABLE.

1.2.1.2 State Machine – Trigger Mechanisms (Incoming)

As mentioned in the previous subsection, for each parameter set, the start of the computations can be triggered based on specific events. Four trigger mechanisms are supported as follows.

- Immediate trigger (TRIGMODE = 000b): In this case, the state machine does not wait for any trigger and starts the accelerator computations immediately for the current parameter set. This mode is applicable when chaining (sequencing) a set of operations one after another in the accelerator without any need for control handshake or data exchange outside the accelerator (for example, when chaining FFT and log-magnitude operations) with no need to wait for a trigger in between.
- Wait for processor-based software trigger (TRIGMODE = 001b): This is a software-triggered mode that is useful when the main processor must directly control the data flow and start or stop of accelerator computations. In this trigger mode, the state machine waits for a software-based trigger, which involves the main processor setting a separate self-clearing bit in a CR42ACCTRIG register (single-bit register). The state machine keeps monitoring that register bit and waits as long as the value is zero. When the value becomes 1 (set), the state machine gets triggered to start the accelerator operations for the current parameter set.
- Wait for ADC buffer ping-to-pong or pong-to-ping switch (TRIG_MODE = 010b): This trigger mode is applicable when RF and analog front-end is integrated in the same chip with the main processor and the Radar Hardware Accelerator. Recall that in these devices, the ADC ping and pong buffers can be shared with the accelerator local memories (ACCEL_MEM0 and ACCEL_MEM1), such that the ADC data is directly available to the accelerator for processing during active chirping portion of the frame. This sharing mode is enabled by setting FFT1DEN register bit before the start of the frame. In this trigger mode, the accelerator's State Machine starts the computations for the current parameter-set as soon as the ADC buffer switches from ping-to-pong or pong-to-ping. As an example, during the active chirping portion of a frame, the digital front-end and ADC buffer can be configured to switch from ping-to-pong or pong-to-ping buffer at the end of every chirp or at the end of every few chirps or at the end of every specified number of ADC samples. (This digital front-end configuration is accomplished using other registers not related to the Radar Hardware Accelerator and not described in this document).

Now, using this trigger mode (TRIG_MODE = 010b) allows the accelerator's computations to start whenever the ping-to-pong or pong-to-ping switch happens in the ADC buffer, thus enabling inline per-chirp processing. It is important to mention here that the user needs to take care to ensure that processing of the current 'ping' data is completed by the accelerator, before the next switch/trigger happens on the ADC buffer. In other words, the chirp duration (ping-pong switch frequency) should not be configured to be so fast that the accelerator cannot complete its configured operations within that duration.

- Wait for the DMA-based trigger (TRIGMODE = 011b): This trigger mode is useful when a DMA transfer completion must be used to trigger the start of the accelerator computations for the current parameter set. The primary purpose of this trigger mode is as follows; when performing second dimension FFT, the DMA is used to bring the FFT input samples from the Radar data memory to the local memory of the accelerator. Upon completion of each DMA transfer, it is useful to automatically trigger the accelerator to perform the FFT.

  To achieve this, the state machine of the accelerator has a 16-bit register called the DMA2ACCTRIG register, where each register bit maps to one of 16 DMA channels that are associated with the accelerator. To use the DMA-based trigger mode, the DMA2ACC_CHANNEL_TRIGSRC register in the current parameter set must be programmed to the DMA channel whose completion we wish to monitor. The state machine then monitors the corresponding register bit in the DMA2ACCTRIG register, and triggers the execution of the current parameter set only when that register bit gets set. If DMA2ACC_CHANNEL_TRIGSRC is programmed to 5, then the current parameter set will execute only once the register bit #5 gets set in DMA2ACCTRIG.

  The user may utilize the EDMA's linking capability to set the appropriate register bit in DMA2ACCTRIG. Linking is a programmable feature of the EDMA, where the completion of a DMA transfer can automatically trigger a second DMA transfer. In the present context, the DMA transfer that moves data to the local memory of the accelerator can be linked to a second DMA whose purpose is to write a one-hot signature into DMA2ACCTRIG to set a specific register bit and trigger the accelerator. Note that there are 16 read-only, one-hot, signature registers (SIG_DMACH1_DONE, SIG_DMACH2_DONE, and more) that are available. These registers are simply read-only registers which contain hard-coded values (each register is a one-hot signature – 0x0001, 0x0002, 0x0004, 0x0008, and so on). For convenience, these hard-coded 16 read-only signatures can be used, so that the second DMA can simply copy from one of these SIG_DMACHx_DONE registers into the DMA2ACCTRIG register to set the appropriate register bit.

1.2.1.3 State Machine – Trigger Mechanisms (Outgoing)

After the accelerator computations for the current parameter set are triggered (using one of the four incoming trigger mechanisms mentioned in the previous subsection), it performs the actual computation operations for that parameter set. These computations typically take several tens or hundreds of clock cycles, depending on the nature of the configuration programmed. Once the accelerator completes its computation operations for the current parameter set, the state machine advances to the next parameter set and repeats the same process. But before advancing to the next parameter set, it can interrupt the main processor and/or trigger a DMA channel. This provision is useful if the main processor is required to read or write registers or memory locations at the end of the current parameter set. Also, this provision is useful for triggering a DMA channel, so that the output of the accelerator can be copied out of the accelerator local memories.

There are two trigger mechanisms provided as follows:

- Interrupt to main processor (INTR_EN = 1): The accelerator will interrupt the main processor at the end of completion of computations for the current parameter-set, if the register bit INTR_EN is set, the interrupt signal from the accelerator is sent to any coupled main processor. The user can enable interrupt response and perform interrupt processing in either of these processors.

- Trigger to DMA (DMATRIGEN = 1): The accelerator gives a trigger to a DMA channel at the end of completion of computations for the current parameter set, if the register bit DMATRIGEN is set. If DMATRIGEN is set, then the particular DMA channel as specified in a separate ACC2DMA_CHANNEL_TRIGDST register (valid values are 0 to 15, for the 16 DMA channels dedicated for the accelerator) is triggered. Thus, it is possible to preconfigure up to 16 DMA channels and trigger the appropriate one at the end of the computations of the current parameter set. The trigger from accelerator to the DMA channels can also be faked by the processor, by writing to a CR42DMATRIG register.

  This can be used by the processor to kick-start a full/repetitive chain of operations, that are then subsequently managed between the DMA and the accelerator without further processor involvement – for example, the processor writes to the CR42DMATRIG register to trigger a DMA channel for the first time, and this kicks off a series of back-to-back data transfers and accelerator computations, with the DMA and accelerator hand-shaking with each other.

1.2.1.4 State Machine – Register Descriptions

Table 2 lists all the registers of the state machine block. As explained previously, some of the registers are common (common for all parameter sets) registers, whereas some others are *part of each parameter set*. For each register, this distinction is captured as part of the register description in Table 2.

Table 2. State Machine Registers

Register	Width	Parameter Set	Description
ACCENABLE	3	No	Enable and Disable Control: This register enables or disables the entire Radar Hardware Accelerator. The reason for a 3-bit register (instead of 1-bit) is to avoid an accidental bit-flip (for example, transient error caused by a neutron strike) from unintentionally turning on the accelerator engine. A value of ACCENABLE = 111b enables the Radar Hardware Accelerator and any other value of the register keeps the accelerator engine in disabled state.
ACCCLKEN	1	No	Clock-gating Control: This register bit controls the enable/disable for the clock of the Radar Accelerator. This register bit can be set to 0 to clock-gate the accelerator when not using the accelerator. Before enabling the accelerator or before configuring the accelerator's registers, this register bit should be set first, so that the clock is available.
ACCRESET	3	No	Software Reset Control: This register provides software reset control for the Radar Hardware Accelerator. The assertion of these register bits by the main processor will bring the accelerator engine to a known reset state. This is mostly applicable for resetting the accelerator in case of unexpected behavior. Under normal circumstances, it is expected that whenever the accelerator is enabled (from disabled state), it always comes up in a known reset state automatically. The recommended sequence to be followed in case software reset is desired is to write 111b to this register and then a 000b, before the clock is enabled to the accelerator.
NLOOPS	12	No	Number of loops: This register controls the number of times the state machine will loop through the parameter sets (from a programmed start index till a programmed end index) and run them. The maximum number of times the loop can be made is run is 4094. A value of 4095 (0xFFF) programmed in this register should be considered as a special case and it should be interpreted as an infinite loop mode, for example, keep looping and never stop the accelerator engine unless reset by the main processor. A value of zero programmed in this register means that the looping mechanism is disabled. In this case, the accelerator engine can still be used under direct control of the main processor (without the state machine looping provision coming into the picture).
PARAMSTART	4	No	Parameter-set Start and Stop Index: These registers are used to control the start and stop index of the parameter set through which the state machine loops through. The state machine starts at the parameter set specified by PARAMSTART and loads each parameter set one after another and runs the accelerator as per that configuration. When the state machine reaches the parameter set specified by PARAMSTOP, it loops back to the start index as specified by PARAMSTART.
PARAMSTOP	4	No	

Table 2. State Machine Registers (continued)

Register	Width	Parameter Set	Description
FFT1DEN	1	No	ADC buffer sharing mode: This register is relevant when the Radar Hardware Accelerator is included in a single device along with the mmWave RF front-end. In such a case, during active chirp transmission and inline first dimension FFT processing, the ACCEL_MEM0 and ACCEL_MEM1 memories of the accelerator are shared as ping-pong ADC buffers. This register bit needs to be set during this time, so that while the digital front end writes ADC samples to the ping buffer, the accelerator automatically accesses (only) the pong buffer, and vice versa. At the end of the active transmission portion of a frame, this bit can be cleared, so that the accelerator has access to all the four local memories independently.
TRIGMODE	3	Yes	Trigger mode control: This parameter-set register is used to control how the state machine and the operations of the accelerator are triggered for each parameter set. The following modes are supported: • 000b – Immediate trigger • 001b – Software trigger • 010b – Ping-pong switch based trigger (applicable only when FFT1DEN is set) • 011b – DMA-based trigger The trigger modes are described in Section 1.2.1.2.
CR42ACCTRIG	1	No	Software trigger bit: This register bit is relevant whenever software triggered mode is used (for example, TRIGMODE = 001b). Whenever software triggered mode is configured for a parameter set, the state machine keeps monitoring this register bit and waits as long as the value is zero. The main processor software can set this register bit, so that the state machine gets triggered and starts the accelerator operations for that parameter set.
DMA2ACCTRIG	16	No	DMA trigger register: This register is relevant whenever DMA triggered mode is used (for example, TRIGMODE = 011b). Whenever a DMA channel has finished copying input samples into the local memory of the accelerator and wants to trigger the accelerator, the procedure to follow is to use a second linked DMA channel to write a 16-bit one-hot signature into this register to trigger the accelerator. In DMA triggered mode, the state machine keeps monitoring this 16-bit register and waits as long as a specific bit (see DMA2ACC_CHANNEL_TRIGSRC) in this register is zero. The second linked DMA channel writes a one-hot signature that sets the specific bit, so that the state machine gets triggered and starts the accelerator operations for that parameter set.
DMA2ACC_CHANNEL_TRIGSRC	4	Yes	DMA channel select for DMA completion trigger: This parameter-set register is relevant whenever DMA triggered mode is used (for example, TRIGMODE = 011b). This register selects the bit number in DMA2ACCTRIG for the state machine to monitor to trigger the operation for that parameter set.
CR4INTREN	1	Yes	Completion interrupt to main processor: This parameter-set register is used to enable/disable interrupt to the main processor upon completion of the accelerator operation for that parameter set. If enabled, the main processor receives an interrupt from the Radar Hardware Accelerator at the end of operations for that parameter set, so that the main processor can take any necessary action.
PARAMDONESTAT (read-only)	16	No	Parameter-set done status: This read-only status register can be used by the main processor to see which parameter sets are complete that led to the interrupt to the main processor. The individual bits in this 16-bit status register indicate which of the 16 parameter sets have completed. These status bits are not automatically cleared, but they can be individually cleared by writing to another 16-bit register PARAMDONECLR.
PARAMDONECLR	16	No	
DMATRIGEN	1	Yes	Completion trigger to DMA: This parameter-set register is used to enable DMA channel trigger upon completion of the accelerator operation for that parameter set. This trigger mechanism enables the accelerator to hand-shake with the DMA so that output data samples are copied out of the accelerator local memory. If enabled, the accelerator triggers a specified DMA channel, so that the output samples can be shipped from the local memory to Radar data memory.

Table 2. State Machine Registers (continued)

Register	Width	Parameter Set	Description
ACC2DMA_CHANNEL_ TRIGDST	4	Yes	DMA channel select for accelerator completion trigger: This parameter-set register is used to select which of the 16 DMA channels allocated to the accelerator should be triggered upon completion of the accelerator operation for that parameter set. This register is to be used in conjunction with DMATRIGEN.
CR42DMATRIG	16	No	Trigger from processor to DMA: This register can be used by the processor to trigger a DMA channel for the first time, so that a full sequence of repeated operations between the DMA and the accelerator gets kick-started.
PARAMADDR	4	No	Debug register for current parameter-set index: This read-only status register indicates the index of the current parameter set that is under execution. This is useful for debug, where parameter sets can be executed in single-step manner (one-by-one) using SW trigger mode for each of them. In such a debug, this register indicates which parameter set is currently waiting for the SW trigger.
LOOPCNT	12	No	Debug register for current loop count: This read-only status register indicates what is the loop count that is presently running. When the state machine is programmed for NLOOPS loops, this register shows the current loop count that is running.
ACC_TRIGGER_IN_STAT	19	No	Debug register for trigger status: This is a read-only status register, which indicates the trigger status of the accelerator, for example, whether a specific DMA trigger or a Ping-pong trigger or a SW trigger was ever received (refer TRIGMODE). The MSB 16 bits of this register indicate whether a trigger was received via DMA trigger method. The next two bits (for example, bit indices 2 and 1) indicate the status of DFE ping-pong switch-based trigger and SW trigger respectively. The LSB bit is always 1 and can be ignored.
ACC_TRIGGER_IN_CLR	1	No	Clear trigger status read-only register: This register-bit when set clears the trigger status register ACC_TRIGGER_IN_STAT described above

The next two sections cover the Input Formatter and Output Formatter blocks, including their detailed operation, registers and usage procedure.

1.3 Accelerator Engine – Input Formatter

This section describes the input formatter block present in the accelerator engine (see Figure 5).

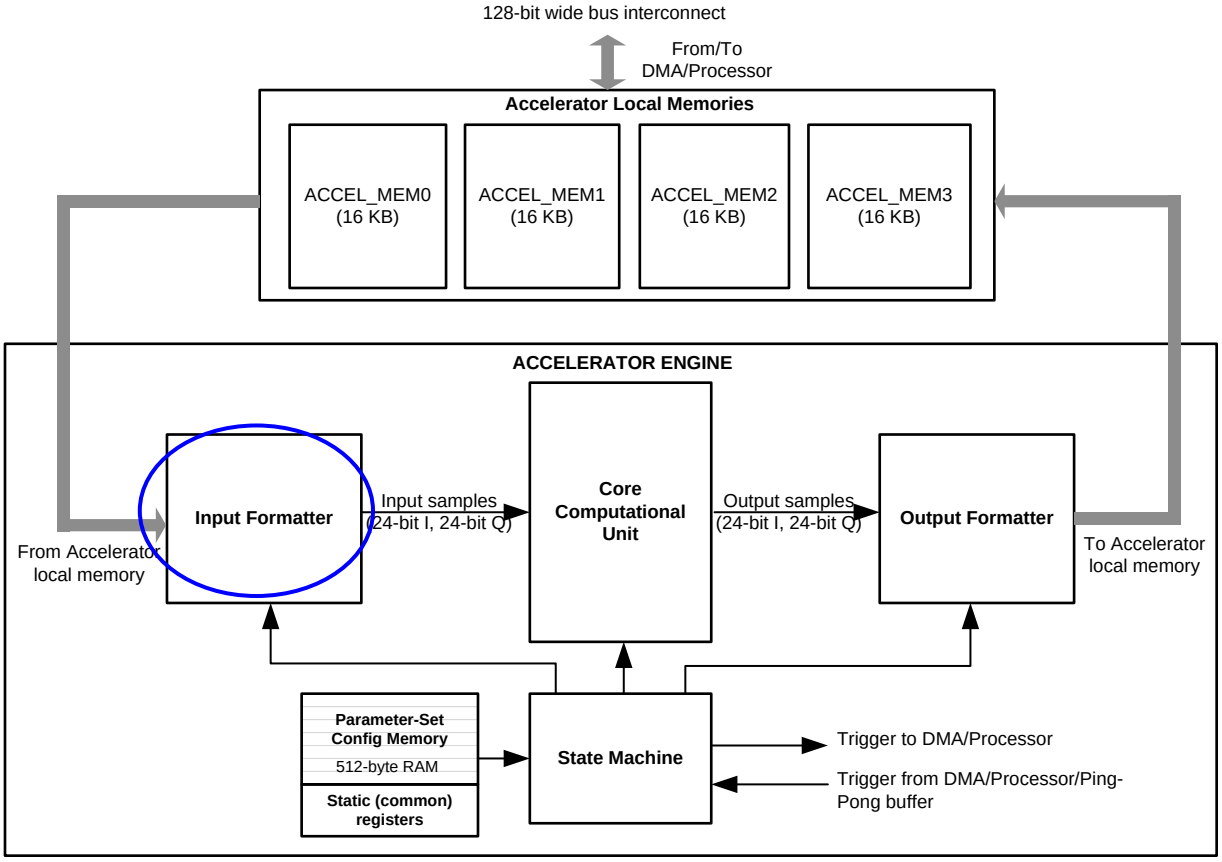


Figure 5. Input Formatter

1.3.1 Input Formatter

The input formatter is used to access, format, and feed the data from the local memories of the accelerator as 24-bit I and 24-bit Q samples into the core computational unit. The input formatter provides various capabilities to access and format the samples from the local memories – especially, various multidimensional access patterns (for example transpose access), 16-bit or 32-bit aligned word access, scaling using bit-shifts to generate 24-bit wide samples from 16-bit or 32-bit words, real versus complex input, sign extension, conjugation, and more.

1.3.1.1 Input Formatter – Operation

The input formatter block is responsible for reading the input samples from the accelerator local memory and feeding them into the core computational unit (see Figure 2). The data flow from the input formatter, through the core computational unit, to the output formatter is designed to sustain a steady-state throughput of one complex sample per clock cycle. The input formatter thus feeds one sample (24-bit I and 24-bit Q) into the core computational unit every clock cycle.

To make the best use of the capabilities of the core computational unit and to allow meaningful chaining of radar signal processing operations with minimal intervention from the R4F processor, the input formatter supports flexibility in how the input samples are accessed from the memory and how they are formatted and fed into the core computational unit.

The memory from which the input formatter picks up the data is referred to as *source memory*. Note that any of the four accelerator local memories can be the source memory. However, as will be described in a subsequent section, there is an important restriction which explains that the source memory cannot be the same as the destination memory (which is the memory to which the output formatter writes the output data).

1.3.1.2 Input Formatter – 2D Indexed Addressing for Source Memory Access

The 16-bit parameter-set register SRCADDR specifies the start address at which the input samples must be accessed. This register is a byte-address, and a value of 0x0000 corresponds to the first memory location of ACCEL_MEM0 memory. The 16-bit SRCADDR register maps to the entire 64KB address space of the four accelerator local memories (4x16KB).

The input data can be read from the memory as either 16-bit wide samples or 32-bit wide samples. Also, they can be read as real samples or complex samples. These two aspects are configured using register bits SRC16b32b and SRCREAL. See Table 3 for a description of these and other registers pertaining to the input formatter block. As an example, if SRC16b32b = 0 and SRCREAL = 0, then the input samples are read from the memory as 16-bit complex samples (16-bit I and 16-bit Q), shown in Figure 6. In the mmWave devices, the ADC buffer is always filled with complex samples from the digital front end – this is true even if the device is configured for real-only operation, in which case the Q-channel output is written with zero values. Therefore, for all purposes of part one of the user guide, SRCREAL can be configured as 0.

An important feature of the input formatter block is that it supports flexible access pattern to fetch data from the source memory, which makes it convenient when the data corresponding to multiple RX channels are interleaved or when performing multi-dimensional (FFT) processing. This feature is facilitated through the SRCAINDX, SRCACNT, SRCBINDX, and REG_BCNT registers, which are part of each parameter-set configuration.

The register SRCAINDX specifies how many bytes separate successive samples to be fetched from the source memory and the register SRCACNT specifies how many samples need to be fetched per iteration. An iteration is typically one computational routine, such as one FFT operation. It is possible to perform multiple iterations back-to-back – for example, four FFT operations corresponding to four RX channels. The register SRCBINDX specifies how many bytes separate the start of input samples for successive iterations and REG_BCNT specifies how many iterations to perform back-to-back. These registers can be better understood using the example given in Figure 6. Also, a complete use case is illustrated in Section 1.6, which provides further clarity on this aspect.

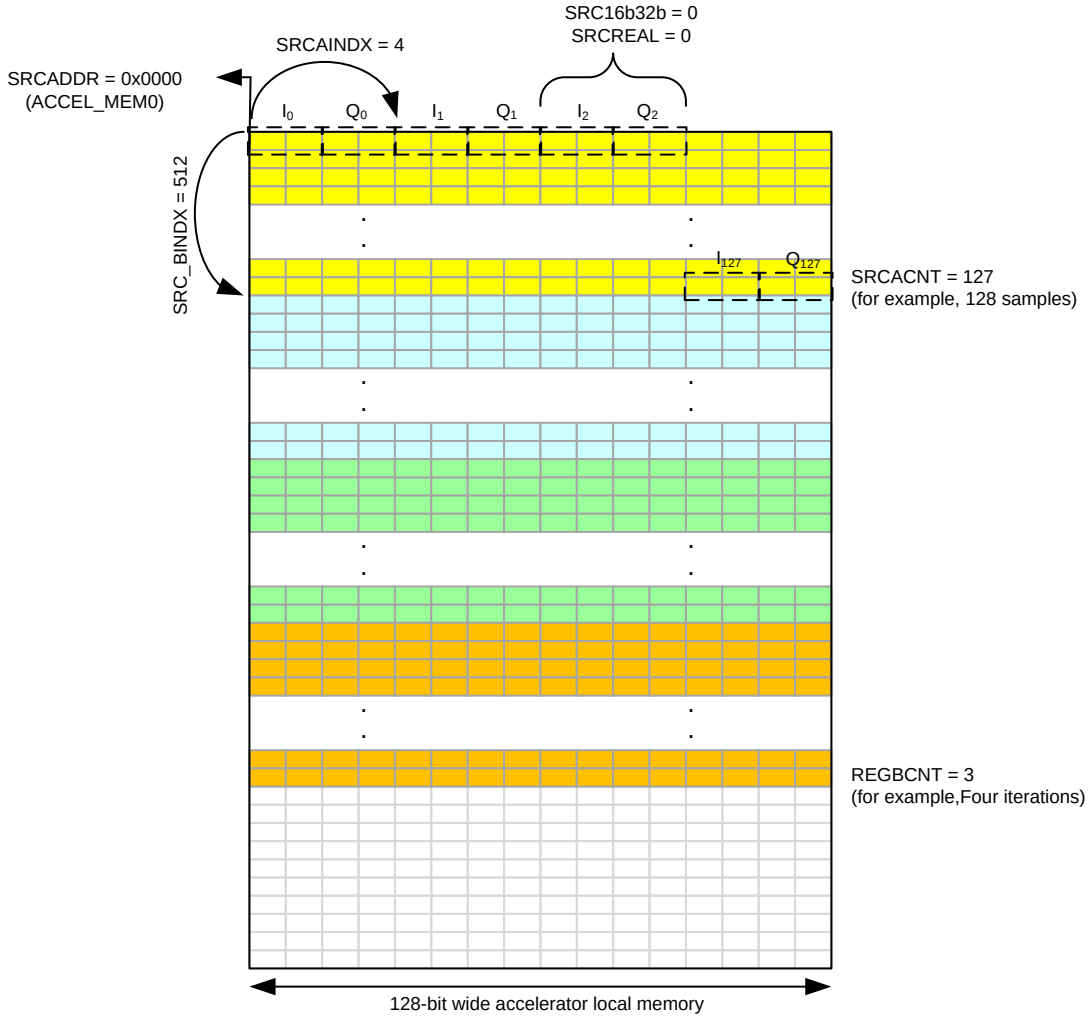


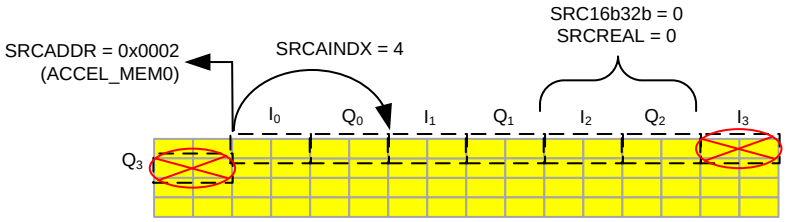
Figure 6. Input Formatter Source Memory Access Pattern (Example)

In Section 1.6, the input data consists of complex data (16-bit I and 16-bit Q) that is contiguously present in ACCEL_MEM0. The data in memory consists of four sets of 128 samples each (say, corresponding to four RX antennas) and these are shown in four different colors. Because each sample occupies 4 bytes and the samples are contiguously placed in the memory starting at the beginning of ACCEL_MEM0, values of SRCADDR = 0x0000 and SRCAINDX = 4 are used to fetch these samples.

In each clock cycle, the input formatter fetches one complex sample from the memory and feeds it into the core computational unit (with appropriate scaling, as described later). Because there are 128 samples to be fed for the first iteration (computational routine), a value of SRCACNT = 127 is used. For the second iteration, the samples are fetched starting from a memory location that is SRCBINDX (=128 × 4 = 512) bytes away from SRCADDR.

This process repeats for the programmed number of iterations as per the REG_BCNT register. For example, the value of REG_BCNT = 3 used in this example corresponds to four iterations. Note that the registers shown here are part of parameter-set configuration registers and the four iterations described here can be performed using a single parameter set.

An important restriction in programming the registers related to source memory access pattern is that the input formatter can only read data from one memory row (128-bit memory location) in a clock cycle. Therefore, if a sample is placed in memory such that the real-part (I value) is at the end of one memory location and the imaginary part (Q value) is at the beginning of the next memory location, then that would be an invalid configuration (see Figure 7).



(1) Single sample getting divided into two rows

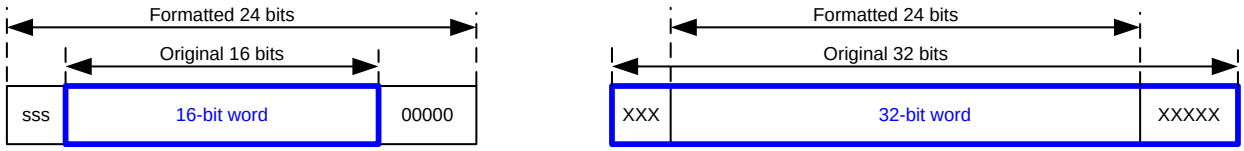
Figure 7. Invalid Configuration Example

1.3.1.3 Input Formatter – Scaling and Formatting

The input formatter allows the input samples read from the source memory to be scaled and formatted before feeding them as 24-bit complex samples into the core computational unit.

Even though the data read from the source memory is initially 16-bits or 32-bits wide (for each I and Q), the samples expected by the core computational unit are 24-bit complex samples (24-bits each for I and Q). There is a REG_SRCSCAL register which provides scaling options using bit-shift to generate 24-bit samples from the original 16- or 32-bit data (see Figure 8).

For the 16-bit case, the 24-bit sample is generated by padding (8-REG_SRCSCAL) zeros at the LSB and REG_SRCSCAL redundant MSBs. For the 32-bit case, the 24-bit sample is generated by dropping REG_SRCSCAL bits at the LSB and clipping (8-REG_SRCSCAL) bits at the MSB. Note that the register bit SRCSIGNED is used to indicate whether the input samples are signed or unsigned. When this register bit is set, the input samples are treated as signed numbers and hence any extra MSB bits are sign-extended and any clipping of MSB bits takes care of signed saturation. In most cases of interest in part one of this user guide (for example, when performing FFT operation), the input samples would be signed and hence SRCSIGNED should be set (for example, equal to 1).



For 16-bit case, if REG_SRCSCAL = 3, then 5 zeros are padded at the LSB, and 3 redundant (extension) bits are padded at the MSB

For 32-bit case, if REG_SRCSCAL = 3, then 5 bits are dropped at the LSB, and 3 bits are clipped (with saturation) at the MSB

(1) 16- or 32-bit words to 24-bit samples

Figure 8. Input Formatter Data Scaling

When the input samples are complex (for example, SRCREAL = 0), there is a provision to conjugate the input samples. Setting the register bit SRCCONJ conjugates the input samples before feeding them to the core computational unit. This feature (together with a corresponding DSTCONJ register bit in the output formatter block) enables an IFFT mode from the FFT engine. Note that conjugating the input and output of an FFT block is equivalent to an IFFT function.

There are other registers in the input formatter, such as BPM_EN, BPMPATTERNLSB and BPMPATTERNMSB, BPMRATE, CIRCIRSHIFT, CIRCSHIFTWRAP, and so on, which are beyond the scope of part one of this user's guide and these registers are described in part two. For the immediate purpose of the first part of the user's guide, it is important to note that BPM_EN and CIRCIRSHIFT registers must be kept 0.

1.3.1.4 Input Formatter – Zero Padding

The input formatter has provision for *zero padding*, which is important when performing FFT of a set of samples whose length is not a power of 2. The input formatter automatically feeds the required number of zeros into the core computational unit, whenever the FFT size (as programmed using the FFTSIZE register, which is described in a later section) does not match the SRCACNT setting.

For example, if the number of input samples read by the input formatter is 56 (for example, SRCACNT = 55) and the FFT size is programmed to be 64 (for example, FFTSIZE = 6), then the input formatter feeds 8 zeros at the end of each iteration, before starting to read the input samples for the next iteration from the source memory. This zero-padding provision enables the core computational unit to perform 64-point FFT with the correct set of zero-padded input samples. It is important for the user to note that SRCACNT should never be larger than 2^FFTSIZE-1.

The zero padding is effective only when performing FFT operation in the core computational unit (when FFT_EN = 1) and not otherwise. Please refer to section 6 for further information regarding the registers relevant for FFT operation.

1.3.1.5 Input Formatter – Register Descriptions

Table 3 lists all the registers of the input formatter block.

Table 3. Input Formatter Registers

Register	Width	Parameter Set	Description
SRCADDR	16	Yes	Source start address: This register specifies the starting address of the input samples, for example, it specifies the source memory start address from which input samples have to be fetched by the input formatter. This is a byte-address and this 16-bit register covers the entire address space of the four local memories (4 × 16KB = 64 KB). The four accelerator local memories are contiguous in the memory address space and any of them can act as the source memory (as long as the same memory bank is not configured to be used as destination memory at the same time).
SRCACNT	12	Yes	Source sample count: This register specifies the number of samples (minus 1) from the source memory to process for every iteration. The sample count is in number of samples, not number of bytes. For example, the sample count can be specified as 255 (SRCACNT = 0x0FF) in a case where a 256-point FFT is required to be performed. Note however that the sample count register does not always match the FFT size. This can happen when zero-padding of input samples is required. For example, a sample count of 192 could be used with an FFT size of 256, in which case, the input formatter will automatically append 64 zeros.
SRCAINDX	16	Yes	Source sample index increment: This register specifies the number of bytes separating successive samples in the source memory. For example, a value of SRCAINDX = 16 means that successive samples are separated by 16 bytes in memory. The maximum value allowed for this register is 32767.
REG_BCNT	12	Yes	Number of iterations: This register specifies the number of times (minus 1) the processing should be repeated. This register can be used to process the four RX chains back-to-back – for example, a value of REG_BCNT = 3 means that the processing (say first dimension FFT processing) is repeated four times. Note the distinction between the NLOOPS register of the state machine block and the REG_BCNT register of the input formatter block. The NLOOPS register specifies how many times the state machine loops through all the configured parameter sets (with each time possibly awaiting a trigger), whereas the register REG_BCNT specifies how many times the input formatter and the computational processing of the accelerator is iterated back-to-back for the current parameter set (without any intermediate triggers).
SRCBINDX	16	Yes	Source offset per iteration: This register specifies the number of bytes separating the starting address of input samples for successive iterations. For example, when using four iterations to process the four RX chains, this register can be used to specify the offset in the starting address between the successive RX chains. Note the distinction that SRCAINDX specifies the number of bytes separating successive samples for a particular iteration, whereas SRCBINDX specifies the number of bytes separating the starting address of the first sample for successive iterations. The maximum value allowed for this register is 32767.

Table 3. Input Formatter Registers (continued)

Register	Width	Parameter Set	Description
SRCREAL	1	Yes	Complex or real Input: This register-bit specifies whether the input samples are real or complex. A value of SRCREAL = 0 implies complex input and a value of SRCREAL = 1 implies real input. When real input is selected, the input formatter block automatically feeds zero for the imaginary part into the core computational unit.
SRC16b32b	1	Yes	16-bit or 32-bit input word alignment: This register-bit specifies whether the input samples fetched from source memory are to be read as 16-bits or 32-bits wide. A value of SRC16b32b = 0 implies that the input samples are 16-bits wide each (in case of complex input, real and imaginary parts are each 16 bits wide). A value of SRC16b32b = 1 implies that the input samples are 32-bits wide each.
SRCSIGNED	1	Yes	Input sign-extension mode: This register-bit, when set, specifies that the input samples are signed numbers and hence, sign-extension or signed-saturation at the MSB is required when converting 16-bit or 32-bit input words to the 24-bit wide samples to be fed into the core computational unit.
SRCCONJ	1	Yes	Input conjugation: This register-bit specifies whether the input samples should be conjugated before feeding them into the core computational unit. If SRCCONJ is set, then the input samples are conjugated. Setting this register-bit only makes sense if the samples are complex numbers (for example, SRCREAL = 0). This register, together with its counterpart in the output formatter block, enable an IFFT mode for the FFT engine. Note that conjugating the input and output of an FFT block is equivalent to an IFFT function.
REG_SRCSCAL	8	Yes	Input scaling: This register specifies a programmable scaling using bit-shift, when converting the 16-bit or 32-bit wide input data to 24-bit wide samples before feeding into the core computational unit. See Figure 8 and its description for more details regarding this register.
CIRCIRSHIFT	–	–	Described in part two of this user's guide. For the immediate purposes relevant to part one of this user's guide, all of these registers must be kept as 0.
CIRCSHIFTWRAP	–	–	
BPMPATTERNLSB and BPMPATTERNMSB	–	–	
BPMRATE	–	–	
BPMPHASE	–	–	

1.4 Accelerator Engine – Output Formatter

This section describes the output formatter block present in the accelerator engine (see Figure 9).

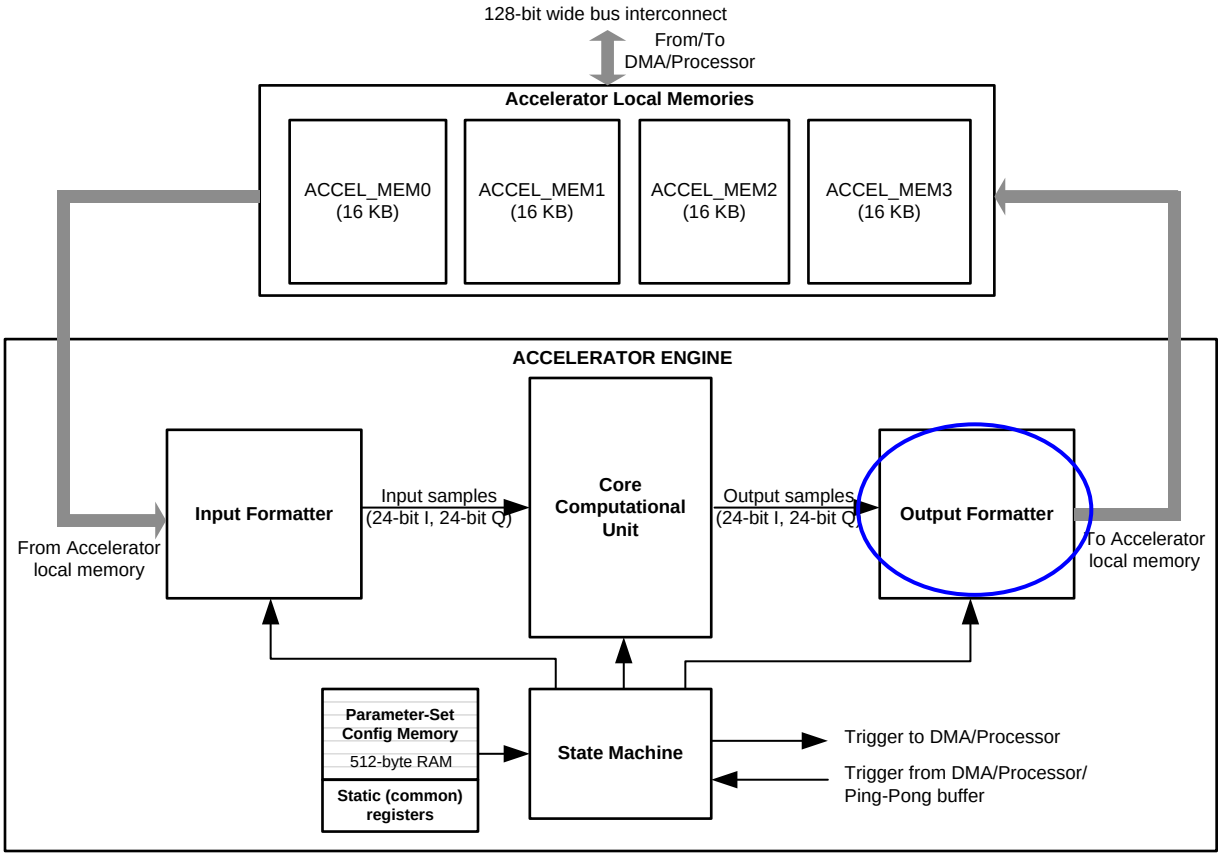


Figure 9. Output Formatter

1.4.1 Output Formatter

The output formatter is used to format and write the data coming out of the core computational unit into the accelerator local memory. Similar to the input formatter block discussed in the previous section, the output formatter block also provides various capabilities to format and write the samples written to the local memory – especially, various multidimensional access patterns (for example, transpose writes), 16-bit or 32-bit aligned word writes, scaling using bit-shifts to generate 16-bit or 32-bit words from 24-bit wide samples, real versus complex output write, and more.

1.4.1.1 Output Formatter – Operation

The output formatter block is responsible for storing the samples coming out of the core computation unit into the accelerator local memory (see Figure 2). As mentioned in the previous section, the data flow from the input formatter, through the core computational unit, to the output formatter, is designed to sustain a steady-state throughput of one complex sample per clock cycle. Thus, typically, the output formatter accepts one sample (24-bit I and 24-bit Q) from the core computational unit every clock cycle and writes it to the accelerator local memory. Just like the input formatter, the output formatter also supports lot of flexibility in how the samples are formatted and written into the memory.

The memory into which the output formatter writes the data is referred to as *destination memory*. Note that any of the four accelerator local memories can be the destination memory, with the important restriction that the source memory cannot be same as the destination memory. In other words, each of the four 16KB memory banks can either function as source memory, or as destination memory at any time (for example, in any given parameter set).

1.4.1.2 Output Formatter – 2-D Indexed Addressing for Destination Memory Access

The 16-bit parameter-set register DSTADDR specifies the start address at which the output samples must be written into the accelerator local memory. Similar to the SRCADDR register of the input formatter, the DSTADDR register of the output formatter is a byte-address and a value of 0x0000 corresponds to the first memory location of ACCEL_MEM0 memory. The 16-bit DSTADDR register maps to the entire 64KB address space of the four accelerator local memories (4 × 16KB). As mentioned in the previous paragraph, in a given parameter set, SRCADDR and DSTADDR cannot be configured such that the input samples being fetched and the output samples being written out are accessing the same memory bank.

Even though the core computational unit produces a 24-bit complex output stream, this output data can be written to the memory as either 16-bit wide samples or 32-bit wide samples. Also, they can be written out as complex samples or real samples (for example, drop imaginary part – applicable when performing log-magnitude computation). These two aspects are configured using register bits DST16b32b and DSTREAL. See Table 4 for a description of these and other registers pertaining to the output formatter block. As an example, if DST16b32b = 0 and DSTREAL = 0, then the output samples are written to the memory as 16-bit complex samples (16-bit I and 16-bit Q), shown in Figure 10.

Similar to the input formatter block, the output formatter block also supports flexible patterns to write multidimensional data to the destination memory and this makes it convenient when the data corresponding to multiple RX channels must be interleaved, or when performing multidimensional (FFT) processing. This feature is facilitated through the DSTAINDX, DSTACNT, DSTBINDX, and REG_BCNT registers, which are part of each parameter-set configuration.

The register DSTAINDX specifies how many bytes separate successive samples to be written to the destination memory and the register DSTACNT specifies how many samples must be written per iteration. Note that DSTACNT can be different from SRCACNT – this is useful when only a subset of the output samples need to be stored in the output memory (for example, if some FFT output bins must be discarded). The register DSTBINDX specifies how many bytes separate the start of output samples for successive iterations and REG_BCNT specifies the number of iterations. The REG_BCNT register is common for input formatter and output formatter. These registers can be better understood using the example given in Figure 10. Also, a complete use case is illustrated in Section 1.6 which provides further clarity on this aspect.

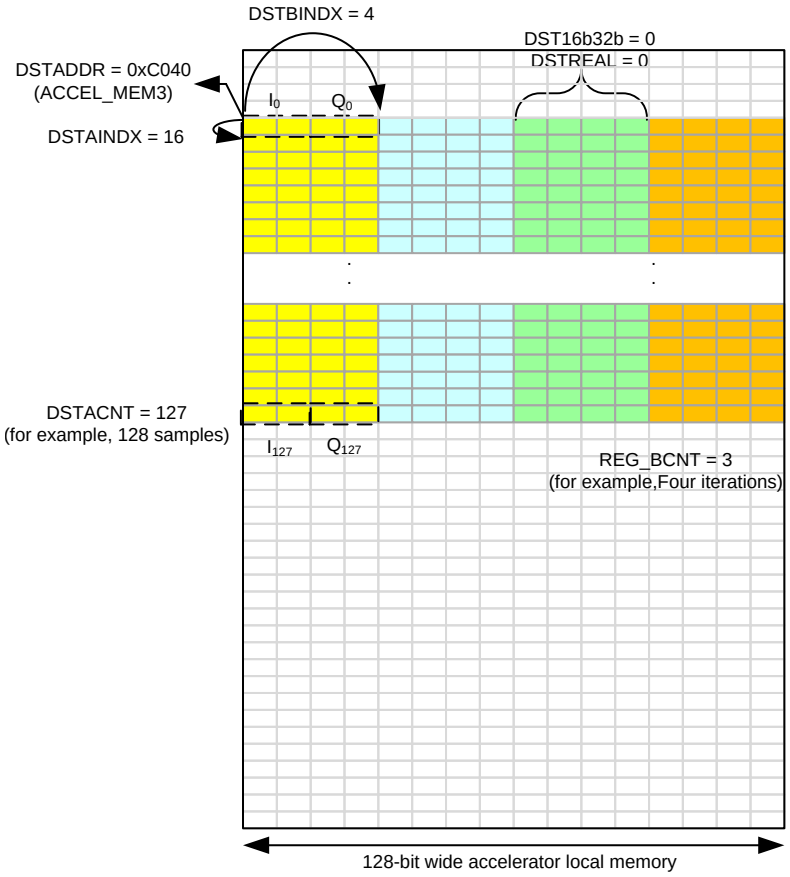


Figure 10. Output Formatter Destination Memory Access Pattern (Example)

In the example shown in Figure 10, the output data consists of complex data (16-bit I and 16-bit Q) that is written to ACCEL_MEM3. The output data consists of four sets of 128 samples each (say, corresponding to FFT output of four RX antennas) and these are shown in four different colors. Each sample occupies 4 bytes and the samples are written to the output memory at a specific start address inside ACCEL_MEM3, as shown in Figure 10. The samples for the four RX antennas are written to the memory in an interleaved manner. Thus, for this example, a value of DSTADDR = 0xC040, DSTAINDX = 16, DSTACNT = 127, and DSTBINDX = 4 are used. The register REG_BCNT (common for input formatter and output formatter) is configured with a value of 3, corresponding to the four iterations required (for the four RX antennas). In steady state, for each clock cycle, the output formatter accepts one complex sample from the core computational unit and writes it into the memory as per the 2-D indexed addressing pattern programmed.

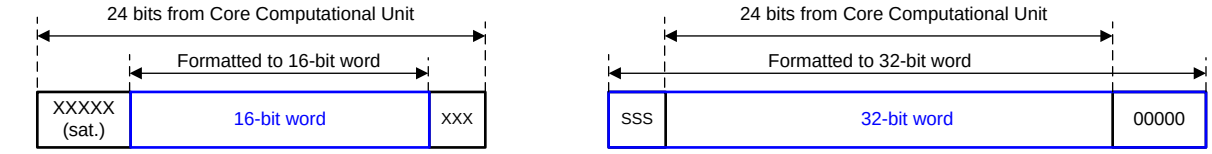
The register DSTACNT, which corresponds to the number of samples written to the destination memory for each iteration does not need to be equal to SRCACNT. This is useful in cases where some of the output samples (for example, some FFT bins at the end) can be dropped and do not need to be written into the destination memory. Another register, REG_DST_SKIP_INIT is also available, which can be used to skip some samples in the beginning as well. The number of samples written to the destination memory for each iteration is equal to (DSTACNT + 1) – REG_DST_SKIP_INIT.

Note that when performing FFT operations, internally the core computational unit sends out FFT output data in bit-reversed addressing order, but this is automatically handled in the output formatter, such that when the FFT output samples are written into the destination memory, they are written out in the correct normal order. Therefore, no special procedure is required on the part of the main processor to read the FFT output samples in the right sequence.

1.4.1.3 Output Formatter – Scaling and Formatting

The output formatter allows the 24-bit output samples from the core computational unit to be scaled and formatted before writing them to the destination memory as 16-bit or 32-bit words. There is a REG_DSTSCAL register which provides scaling options using bit-shift, to take the 24-bit samples and convert them to 16-bit or 32-bit data.

For the 16-bit case, the 24-bit sample (24-bits for each I and Q) is converted to 16-bit word by dropping REG_DSTSCAL bits at the LSB and by clipping with saturation (8-REG_DSTSCAL) bits at the MSB. For the 32-bit case, the 24-bit sample is padded with REG_DSTSCAL extra bits at the MSB and with (8-REG_DSTSCAL) extra zeros at the LSB. Note that the register bit DSTSIGNED is used to indicate whether the output samples are signed or unsigned. When this register bit is set, the output samples are treated as signed numbers and therefore any extra MSB bits are sign-extended and any clipping of MSB bits handles signed saturation. In most cases of interest in part one of this user's guide (for example, when performing FFT operation), the output samples would be signed and therefore DSTSIGNED should be set (for example, equal to 1). However, if the log-magnitude operation in the core computational unit is enabled, then the output samples are unsigned and therefore DSTSIGNED is cleared (for example, equal to zero).



For 16-bit case, if REG_DSTCAL = 3, then 3 btis are dropped at the LSB, and 5 bits are clipped (saturated) at the LSB

For 32-bit case, if REG_DSTCAL = 3, then 5 zeros are padded at the LSB, and 3 bits are extended at the MSB

(1) 24-bit samples to 16- or 32-bit words

Figure 11. Output Formatter Data Scaling

When the output samples are complex (for example, DSTREAL = 0), there is a provision to conjugate the output samples. Setting the register bit DSTCONJ conjugates the output samples before writing them to the destination memory. This feature (together with a corresponding SRCCONJ register bit in the input formatter block) enables an IFFT mode from the FFT engine.

1.4.1.4 Output Formatter – Register Descriptions

Table 4 lists all the registers of the output formatter block.

Table 4. Output Formatter Registers

Register	Width	Parameter Set	Description
DSTADDR	16	Yes	Destination start address: This register specifies the starting address of the output samples, for example, it specifies the destination memory start address at which the output samples have to be written by the output formatter. This is a byte-address and this 16-bit register covers the entire address space of the four local memories (4 × 16KB = 64 KB). The four accelerator local memories are contiguous in the memory address space and any of them can act as the destination memory (as long as the same memory bank is not configured to be used as source memory at the same time).
DSTACNT	12	Yes	Destination sample count: This register specifies the number of samples (minus 1) to be written to the destination memory for every iteration. The sample count is in number of samples, not number of bytes. For example, the sample count can be specified as 191 (DSTACNT = 0x0BF) in a case where 192 samples must be written. Note that the DSTACNT register can be different from SRCACNT or even the FFT size. This is useful when only a part of the FFT bins must be written to memory and the remaining (far-end FFT bins) can be discarded. This register description is true when the REG_DST_SKIP_INIT register value is zero (see further for more information related to REG_DST_SKIP_INIT).

Table 4. Output Formatter Registers (continued)

Register	Width	Parameter Set	Description
DSTAINDX	16	Yes	Destination sample index increment: This register specifies the number of bytes separating successive samples to be written to the destination memory. For example, a value of DSTAINDX = 16 means that successive samples written to the destination memory should be separated by 16 bytes. The maximum value allowed for this register is 32767.
DSTBINDX	16	Yes	Destination offset per iteration: This register specifies the number of bytes separating the starting address of output samples for successive iterations. For example, when using four iterations to process four RX chains, this register can be used to specify the offset in the starting address between the successive RX chains. Note the distinction that DSTAINDX specifies the number of bytes separating successive samples for a particular iteration, whereas SRCBINDX specifies the number of bytes separating the starting address of the first sample for successive iterations. The maximum value allowed for this register is 32767.
REG_DST_SKIP_INIT	10	Yes	Destination skip sample count: This register specifies how many output samples should be skipped in the beginning, before starting to write to the destination memory. This is useful if only a certain part of the FFT output (skipping the first several bins) need to be stored in memory. The total number of samples written to destination memory is equal to DSTACNT+1-REG_DST_SKIP_INIT.
DSTREAL	1	Yes	Complex or real output: This register-bit specifies whether the output samples are real or complex. A value of DSTREAL = 0 implies complex output and a value of DSTREAL = 1 implies real output. When real output is selected, the output formatter block automatically stores only the real part into the destination memory. This is useful when the core computational unit is configured to output magnitude or log-magnitude values.
DST16b32b	1	Yes	16-bit or 32-bit output word alignment: This register-bit specifies whether the output samples are to be written as 16-bits or 32-bits wide in the destination memory. A value of DST16b32b = 0 implies that the output samples are to be written as 16-bit words (in case of complex output, real and imaginary parts are each 16 bits wide). A value of DST16b32b = 1 implies that the output samples are 32-bits wide each.
DSTSIGNED	1	Yes	Output sign-extension mode: This register-bit, when set, specifies that the output samples are signed numbers and therefore, sign-extension or signed-saturation at the MSB is required when converting the 24-bit wide samples coming from the core computational unit into 16-bit or 32-bit output words to be written to the destination memory.
DSTCONJ	1	Yes	Output conjugation: This register-bit specifies whether the output samples must be conjugated before writing them into the destination memory. If DSTCONJ is set, then the output samples are conjugated. Setting this register-bit only makes sense if the samples are complex numbers (for example, DSTREAL = 0). This register, together with its counterpart in the output formatter block, enables an IFFT mode for the FFT engine.
REG_DSTSCAL	8	Yes	Output scaling: This register specifies a programmable scaling using bit-shift, when converting the 24-bit samples coming from the core computational unit into 16-bit or 32-bit wide words to be written to the destination memory. See Figure 11 and its description for more details regarding this register.
STATERRCODE	4	No	Memory access error: This 4-bit read-only register indicates if there is a memory access error caused by incorrect configuration or usage of the accelerator, where both the DMA and the accelerator are attempting to access the same 16KB memory at the same time. The 4-bit register indicates the error status for the 4 16KB memories (MSB bit corresponds to ACCEL_MEM0).
ERRCODEMASK	4	No	Mask for memory error: This register can be used to mask the memory access error. If set, the memory access error indication is disabled.
ERRCODECLR	4	No	Clear memory access error: This register can be used to clear the memory access error indication. Setting this register clears the error indication.

1.5 Accelerator Engine – Core Computational Unit

This section describes the core computational unit present in the accelerator engine (see Figure 12).

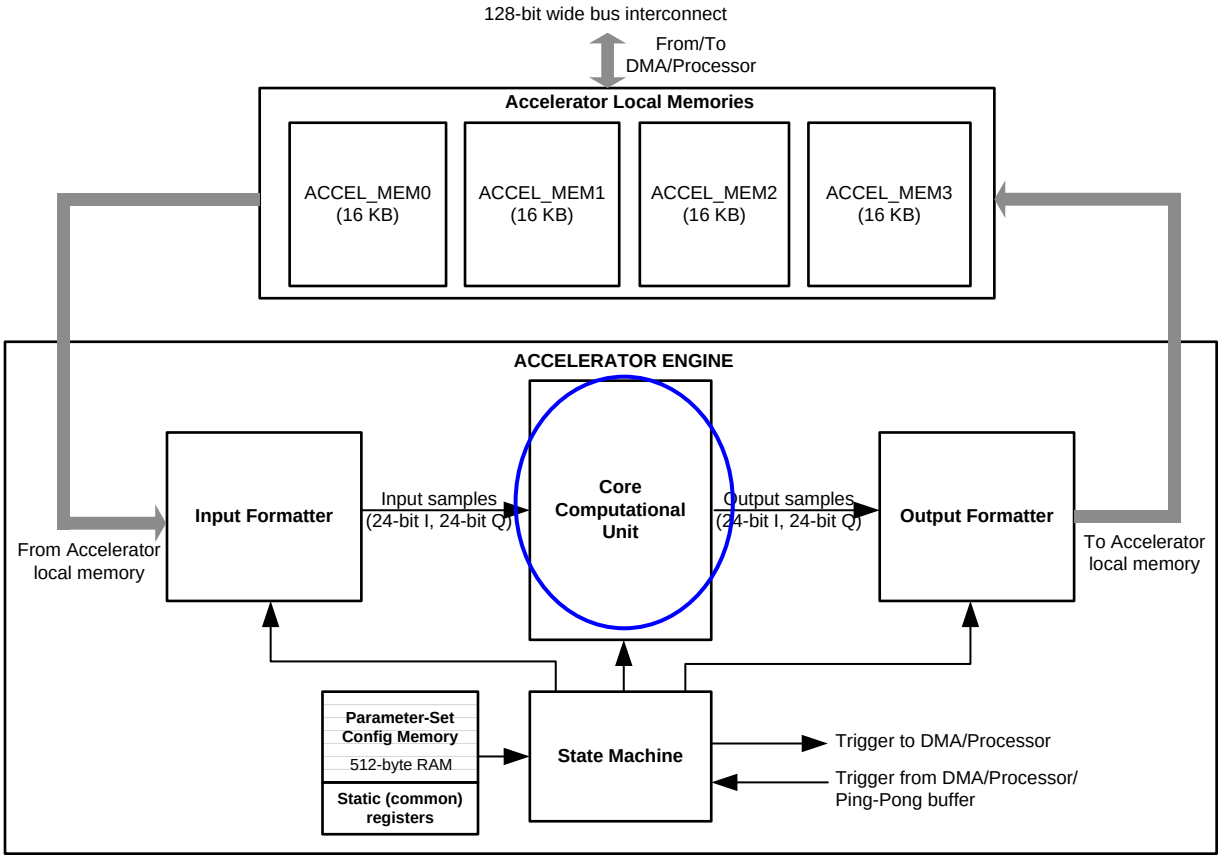


Figure 12. Core Computational Unit

1.5.1 Core Computational Unit

The core computational unit performs the mathematical operations required for the key functions, such as FFT, log-magnitude, and so on. The core computational unit accepts a streaming 24-bit complex input (24 bits for each I and Q) from the input formatter block and it outputs a streaming 24-bit complex output (24 bits for each I and Q) to the output formatter block. In addition to FFT and log-magnitude, the core computational unit has provision for simple pre-FFT processing, such as zeroing out large interference samples, complex derotation, and windowing prior to FFT. The core computational unit also contains a CFAR-CA detector unit for detecting peak samples (for example, radar targets).

Figure 13 shows the block diagram of the core computational unit. The core computational unit has two main paths – namely the FFT Engine path and the CFAR Engine path. Only one of these two paths can be operational at any given instant. However, in separate parameter sets, different paths can be configured and used, so that multiple parameter sets executing one after another can accomplish a sequence of computational operations as desired. The register ACCEL_MODE controls which path gets used in a given parameter set.

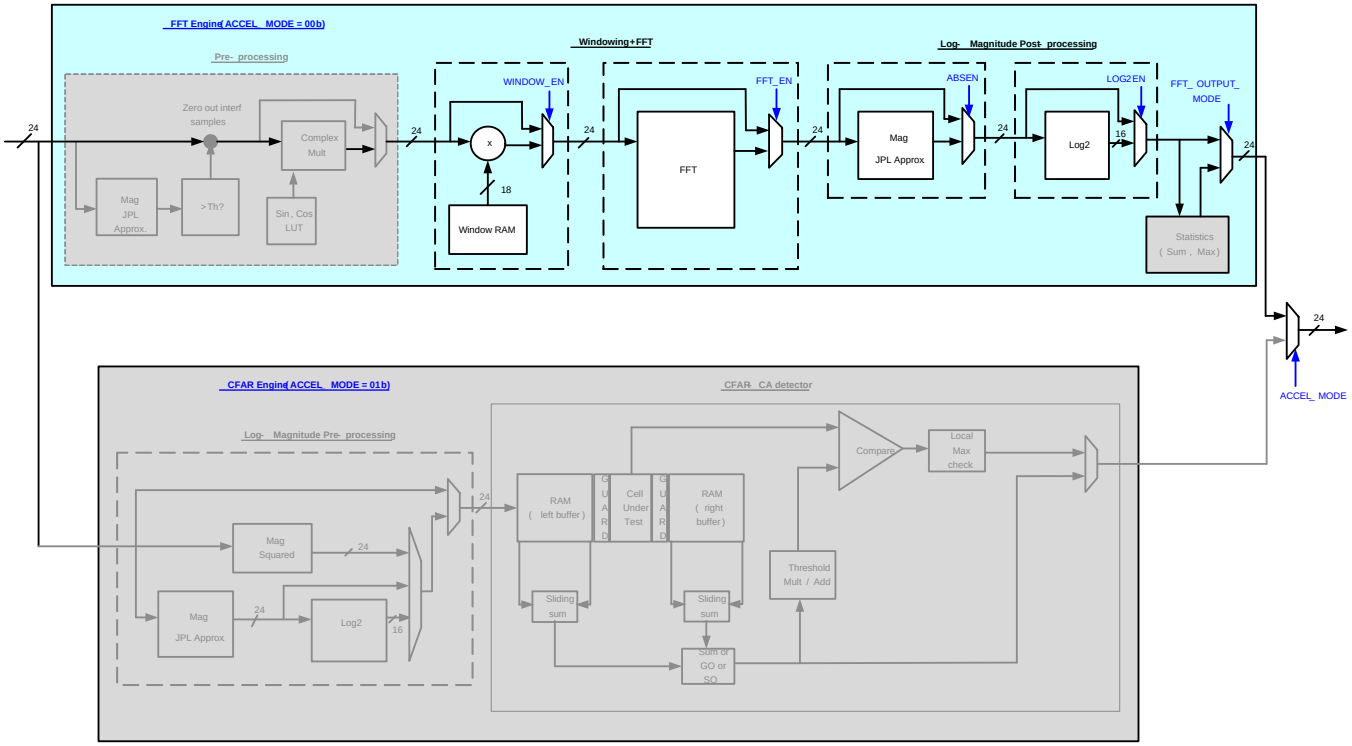


Figure 13. Core Computational Unit Block Diagram

For the purpose of part one of the user's guide, only the FFT Engine path is described. Specifically, the windowing, FFT, and log-magnitude operations are covered in this document. The greyed-out blocks in Figure 13, namely the Pre-processing, Statistics, and CFAR Engine, are covered in part two of the user's guide and can be ignored for the present purpose.

1.5.1.1 Core Computational Unit – Operation

The core computational unit operates on the streaming input of samples coming from the input formatter block, and in general outputs a stream of samples (after an initial latency in some cases) to the output formatter block. In general, at steady-state, one input sample is processed and one output sample is produced every 200-MHz clock.

The core computational unit has the ability to perform windowing, FFT, and log-magnitude computations. Each of these computational subblocks operate on a streaming input and produce a streaming output at the throughput of one sample per clock. These computational subblocks are stitched together one after the other in a series, as shown in Figure 13. This architecture allows multiple operations to be done in a streaming manner (for example, windowing and FFT can be done together), while at the same time, providing the user flexibility to choose one operation at a time.

The parameter-set registers WINDOW_EN, FFT_EN, ABSEN, and LOG2EN control the multiplexers (see Figure 13), which decide what operations are performed on the input samples for that parameter set.

Note that for the purpose of part one of the user's guide, the registers ACCEL_MODE and FFTOUT_MODE must be kept at zero. The purpose of these registers is covered in part two.

1.5.1.2 Core Computational Unit – Windowing

The incoming samples from the input formatter to the core computational unit are passed through the (optional) windowing operation (see Figure 13). Windowing operation is often required prior to performing FFT, to mitigate the sinc roll-off leakage from one strong FFT bin to the adjacent bins.

The implementation of the windowing operation in the accelerator is very straightforward. The window coefficients are preloaded by the Cortex-R4F processor into a dedicated Window RAM. The purpose of this RAM is to provide a fully programmable window (for example, Hann, Kaiser, or any proprietary window) to the user.

As the incoming samples from the input formatter stream in, each sample is multiplied by the appropriate window coefficient read from the RAM. The window coefficients must be real numbers and they are stored as 18-bit, signed, twos-complement numbers in the Window RAM. Because the incoming samples are complex 24-bits wide (24-bits for each I and Q), the windowing operation involves multiplying the 24-bit I and 24-bit Q of the incoming sample with the 18-bit real window coefficient (see Figure 14). The output of this multiplication is rounded back to 24-bit I and 24-bit Q by dropping excess LSBs.

Note that windowing can be enabled or disabled by using the register bit WINDOW_EN.

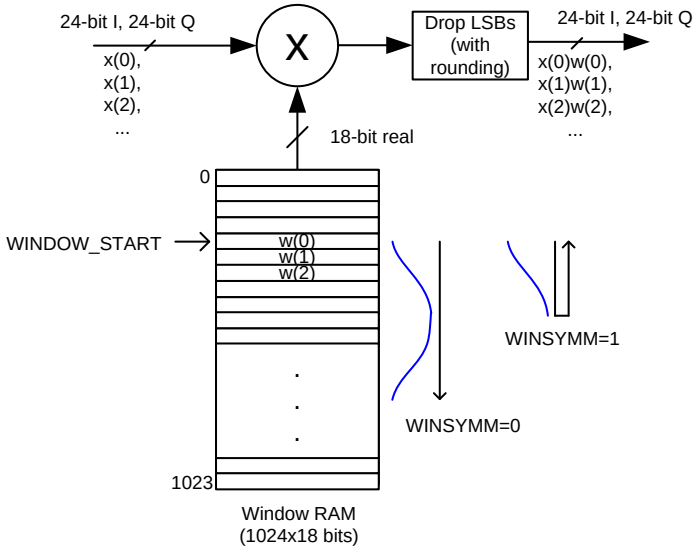


Figure 14. Windowing Computation

The Window RAM can hold a maximum of 1024 coefficients. It is possible to store more than one window function in the Window RAM. For example, two separate windows for first-dimension FFT and second-dimension FFT can be preloaded and kept in the Window RAM, as long as the total number of coefficients is 1024 or less.

The start address (for example, starting coefficient index between 0 to 1023) is programmed in a 10-bit register WINDOW_START as part of the parameter set, so that the windowing computation can pick the appropriate window coefficients starting from that index. For each incoming sample, the index keeps incrementing, so that each successive sample is multiplied by the successive window coefficient. At the end of each iteration (for example, when SRCACNT number of samples have been processed), the index resets back to the starting coefficient index programmed for the parameter set, so that the next iteration can be performed. At the end of all the iterations of the current parameter set, the next parameter set can use a different window if desired. For example, when performing second- and third-dimension FFTs one after another (in two parameter sets), the window functions for both these FFTs can be prestored in the Window RAM and appropriate start index can be provided for each of the FFT operation dimensions.

If the window function is symmetric, only one half of the set of window coefficients needs must be stored in the Window RAM. The register bit WINSYMM, when set, indicates that after SRCACNT / 2 samples (or, if SRCACNT is odd, (SRCACNT + 1) / 2 samples) are processed, the window coefficients read-indexing must be reversed, so that the same set of coefficients used for the first SRCACNT / 2 samples are reused in the reverse order for the next SRCACNT / 2 samples. (See Figure 14). If SRCACNT is odd, then the last window coefficient is read only once, when the direction is reversed. If SRCACNT is even, then the last window coefficients is read twice, when the direction is reversed.

The output of the windowing computation is 24-bit I and 24-bit Q, which is streamed into the FFT subblock.

1.5.1.3 Core Computational Unit – FFT

The FFT subblock performs FFT on the incoming 24-bit I and 24-bit Q data stream. The FFT sizes supported are all powers of 2 until 1024, for example, FFT sizes of 2, 4, 8, 16, … 512 and 1024 are supported. The lowest FFT size of 2 is mostly useful as a *complex add-subtract* feature or while using the *FFT stitching* feature. FFT sizes of 4, 8, 16, and 32 can be used for third dimension (angle estimation) FFT.

Note that FFT stitching is a feature that enables large FFT sizes, specifically, 2048 and 4096, using a two-step process (this feature is not covered here and is discussed in part two of the user's guide).

The FFT operation can be enabled or disabled by using the register bit FFT_EN. When enabled, the FFT subblock computes the FFT of the input data stream and produces a 24-bit I and 24-bit Q output stream. This output stream is initially in bit-reversed order, but the output formatter handles appropriately writing the output to the destination memory in the correct order.

The FFT implementation comprises ten butterfly stages. Depending on the FFT size needed, an appropriate number of butterfly stages are employed. The FFT size is programmed using the FFTSIZE register – for example, FFTSIZE = 5 means 32-point FFT, FFTSIZE = 7 means 128-point FFT, and so on. Note that the FFT size must be equal to or larger than SRCACNT, and the input formatter block automatically zero-pads extra samples to account for the difference between FFT size and SRCACNT. For example, if SRCACNT = 99 (for example, 100 samples) and FFTSIZE = 7 (for example, 128-point FFT), then the input formatter automatically appends 28 zero-pad samples for each iteration.

1.5.1.4 Core Computational Unit – FFT Quantization and Speed performance

As is well known, a butterfly stage typically consists of add-subtract and twiddle multiplication operations. At the output of each add-subtract structure, the bit-width would increase by 1 bit (for example, 24-bit input would grow to 25-bit output). To handle this one-bit growth due to add-subtract operation, there is a provision at the output of each butterfly add-subtract stage to scale the result back to 24 bits, by either dividing the output by 2 (round off one LSB) or by saturating one MSB, shown in Figure 15.

The 10-bit register BFLY_SCALING is used to control this divide-by-2 scaling operation at each stage, so that the user has full flexibility to control the signal level through the different butterfly stages. If BFLY_SCALING = 0 for a particular stage, then the 25-bit output is saturated at the MSB to get back to 24 bits. Otherwise, it is convergent-rounded at the LSB to get back to 24 bits. The user can thus control the scaling at each of the ten butterfly stages. The LSB of this 10-bit register corresponds to the last stage and the MSB of this register corresponds to the first stage. For an FFT size of 64, only the LSB 6 bits are relevant.

There is a 10-bit read-only register FFTCLIP which indicates whether there was any clipping in any of the butterfly stages. This register is a sticky register that gets set when a clipping event occurs and remains set until it is cleared using the CLR_FFTCLIP register bit. See the register description of FFTCLIP in Table 6.

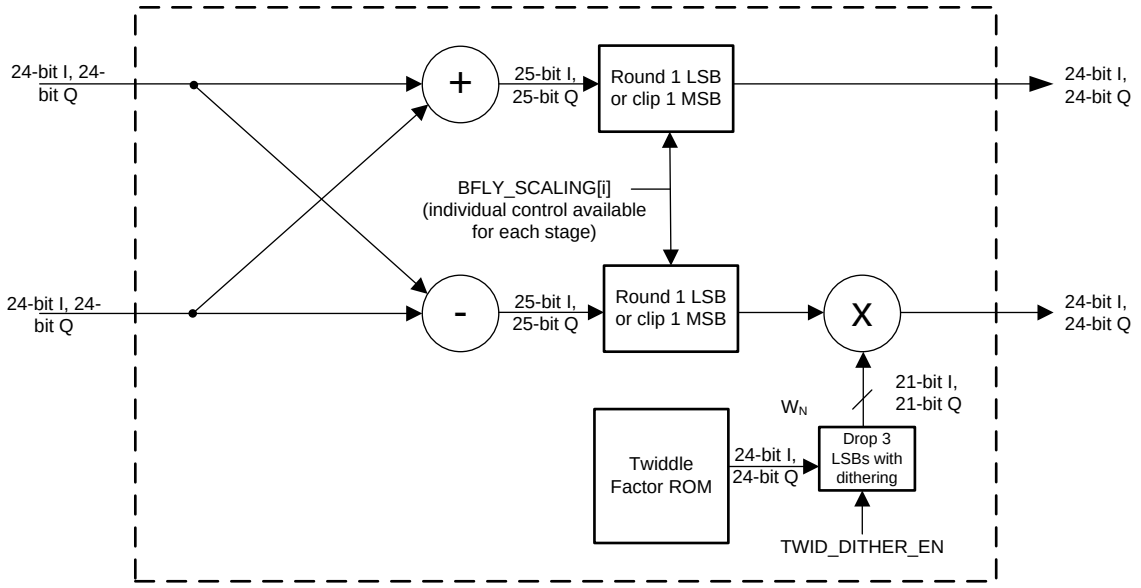


Figure 15. Butterfly Stage Fixed-Point

The twiddle factors are stored as 24-bit I and 24-bit Q coefficients. Prior to twiddle factor multiplication, the coefficients are reduced to 21-bit I and 21-bit Q by dropping three LSBs (with optional dithering). The purpose of dithering is to eliminate any repetitive quantization noise patterns from degrading the SFDR of the FFT. TI recommends that dithering be enabled (DITHERTWIDEN should be set). For dithering, an LFSR is used to generate a random pattern, for which the LFSR seed must be loaded with a non-zero value (see LFSRSEED in the register descriptions).

The SFDR performance of the FFT, with dithering enabled, is better than –140 dBc, as shown in Figure 16.

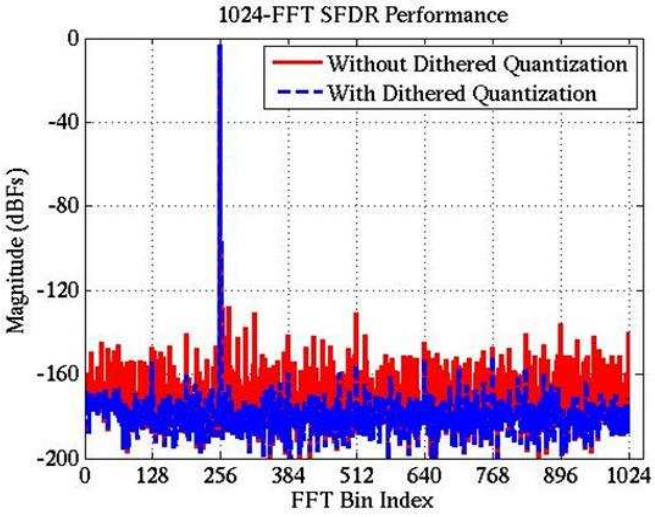


Figure 16. FFT SFDR Performance With and Without Dithering

The architecture of the FFT is such that it can take a streaming input (one sample per clock) and produce a streaming FFT output (one sample per clock), in steady-state. There is an initial latency of approximately *FFT size* number of clocks. This latency only comes into picture once for a given parameter set. Within a parameter set, multiple FFT iterations can be performed back-to-back (for example, for four RX) with no additional latency between iterations.

Because the implementation uses 200-MHz clock, a 256-point complex FFT for four RX chains would take 256 + 256 × 4 clock cycles to complete, which corresponds to 6.4 µs (plus a few clocks of implementation latencies, which are not accounted here). Table 5 lists the approximate computation time needed for various FFT sizes.

Table 5. FFT Computation Time

Example	FFT Size	Number of Back-to-Back Iterations	Number of Clock Cycles (Initial latency + Computation)	Total Duration
1	256	4	256 + (256 × 4)	6.4 µs
2	128	4	128 + (128 × 4)	3.2 µs
3	8	64	8 + (64 × 8)	2.6 µs

The output of the FFT can be fed to the output formatter or it can be sent to the magnitude/log-magnitude computation subblock.

NOTE: The FFT is a complex FFT implementation. If the input samples are real-only, then the SRCREAL register bit can be set, such that the imaginary part (Q-part) will be forced to zero by the input formatter block.

1.5.1.5 Core Computational Unit – Magnitude and Log-Magnitude Post-Processing

The magnitude and log-magnitude post-processing block computes absolute value or log2 of the absolute value of its input. Because this block is connected to the output of the FFT engine, the computation of absolute value (and log2) can be directly performed on the streaming FFT output. Alternately, the FFT block can be bypassed and only the magnitude and log-magnitude block can be employed.

The processing in this block first involves computation of magnitude (absolute value) of the input samples in the magnitude subblock (using JPL approximation). The result of the magnitude computation is fed into a Log2 computation subblock, which uses a look-up table-based approximation to compute logarithm-base-2 of the magnitude.

As shown in Figure 13, if the register-bit ABSEN is set, the magnitude computation subblock is enabled. In addition, if the register-bit LOG2EN is set, then the Log2 computation subblock is also enabled. Note that setting LOG2EN makes sense only when ABSEN is also set.

The magnitude computation uses JPL (Levitt and Morris) approximation. This approximation for magnitude of a complex number (I + jQ) is defined as follows, let U = max($|I|, |Q|$) and V = min($|I|, |Q|$).

Then, the magnitude can be approximated as follows in Equation 1.

$$\text{Magnitude} \approx \max(U + V / 8, 7U / 8 + V / 2) \quad (1)$$

The magnitude output is 24-bits wide (real number).

Next, the log2 computation of the magnitude value is achieved as follows. Any unsigned input number N can be written as N = 2k(1 + f) and the log2(N) can then be written as follows in Equation 2.

$$\log2(N) = k + \log2(1+f) \quad (2)$$

The implementation of log2 computation uses the previous formula, where a look-up table approximation is used to generate the second term, for example, log2(1 + f). The accuracy of the log2 computation is shown in Figure 17. The log2 output is 16-bits wide. The 16-bit logarithm output consists of 5 bits of integer part and 11 bits of fractional part.

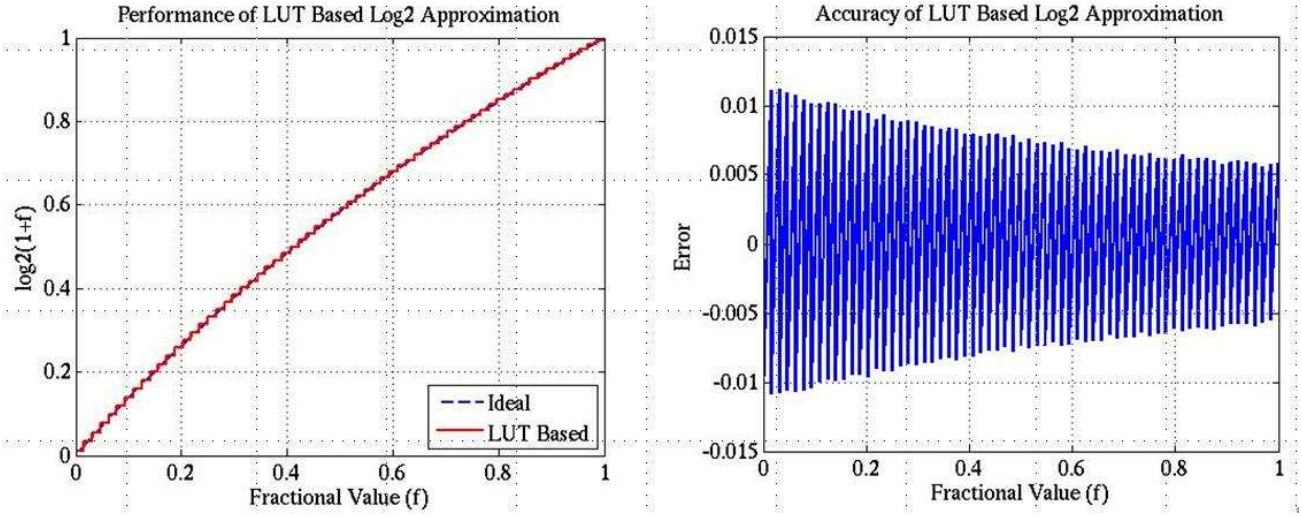


Figure 17. Accuracy of Log2 Computation

Depending on the settings of ABSEN and LOG2EN, either the magnitude or the log-magnitude is sent as the final output of the core computational unit. The final output of the core computational unit going to the output formatter is 24-bits I and 24-bits Q. Thus, if either magnitude or log-magnitude is enabled, the Q-values are just made zeros. Similarly, when log2 is enabled, because the output is 16-bits, 8 MSBs are filled as zero.

The output formatter handles writing the samples to the destination memory as per the configured destination memory access pattern described in a previous section.

1.5.1.6 Core Computational Unit – Register Descriptions

Table 6 lists all the registers of the core computational unit.

Table 6. Core Computational Unit Registers

Register	Width	Parameter Set	Description
WINDOW_EN	1	Yes	Windowing Enable: This register-bit enables or disables the pre-FFT windowing operation. If this register is set to 1, then the windowing is enabled, otherwise, it is disabled. The exact window function (coefficients) to be applied is specified in a dedicated Window RAM, which is 1024 × 18 bits in size.
FFT_EN	1	Yes	FFT Enable: This register-bit is used to enable the FFT computation. If FFT_EN = 1, then the FFT computation is enabled. Otherwise, it is disabled (bypassed).
ABSEN	1	Yes	Magnitude Enable: This register-bit is used to enable the magnitude calculation. If this register bit is set, then the magnitude calculation is enabled, else it is bypassed. When enabled, the magnitude (absolute value) of the input complex samples are calculated using JPL approximation and the resulting magnitude value is sent on the I-arm of the output. The Q-arm is made zeros.
LOG2EN	1	Yes	Log2 Enable: This register-bit is used to enable the Log2 computation. If this register bit is set, then the Log2 computation is enabled, else it is bypassed. Note that setting this register bit only makes sense if the inputs to the Log2 computation are unsigned real numbers, such as when the Magnitude Enable bit (ABSEN) is also set. When enabled, the Log2 of the magnitude of the input samples is calculated and sent out on the I-arm of the output. The Q-arm is made zeros.

Table 6. Core Computational Unit Registers (continued)

Register	Width	Parameter Set	Description
WINDOW_START	10	Yes	Windowing coefficients start index: This register specifies the starting index of the window coefficients within the Window RAM. The value of this register ranges from 0 to 1023. The purpose of this register is to allow multiple windows (for example, one window of 512 coefficients and another window of 256 coefficients) to be stored in the Window RAM and one of these windows can be used by programming this start index register appropriately in the current parameter set.
WINSYMM	1	Yes	Window symmetry: This register-bit indicates whether the complete set of window coefficients are stored in the Window RAM or whether one half of the coefficients are stored. If this register bit is set, it means that the window function is symmetric and therefore, only one half of the window function coefficients are stored in the Window RAM. See the description section related to Windowing computation for more details.
FFTSIZE	4	Yes	FFT size: This register specifies the FFT size. The mapping of the FFTSIZE register to the actual FFT size is as follows: Actual FFT size = 2 ^ (FFTSIZE). For example, a register value of 0110b specifies that the FFT size is 64. The maximum FFT size that is supported is 1024. Therefore, this register value is never expected to exceed 1010b. Note that the FFT size should be equal to or larger than SRCACNT and the Input Formatter block will automatically zero-pad extra samples to account for the difference between FFT size and SRCACNT. For large-size FFT (> 1024 point) that might be useful for industrial level-sensing applications, an FFT stitching procedure is supported, which is based on performing multiple smaller size FFTs in a first step and then stitching them in a second step (using a subsequent parameter set). This FFT stitching feature is covered in part two of the user's guide.
BFLY_SCALING	10	Yes	Butterfly scaling: This register is used to control the butterfly scaling at each stage of the FFT structure. Because the maximum FFT size is 1024, there are up to ten butterfly stages. Each butterfly stage has an add-and-subtract structure, at the output of which the bit-width would temporarily increase by 1 (from 24 to 25 bits wide). If BFLY_SCALING = 0, then the 25-bit output is saturated at the MSB to get back to 24 bits. Otherwise, it is convergent-rounded at the LSB to get back to 24 bits. The user can thus control the scaling at each of the 10 butterfly stages. The LSB of this register corresponds to the last stage and the MSB of this register corresponds to the first stage. For an FFT size of 64, only the LSB 6 bits are relevant.
DITHERTWIDEN	1	No	Twiddle factor dithering enable: This register-bit is used to enable and disable dithering of twiddle factors in the FFT. The twiddle factors are 24-bits wide (24-bits for each I and Q), but they are quantized to 21-bits before twiddle factor multiplication. This quantization is implemented with dithering on the LSB, to avoid periodic quantization pattern affecting SFDR performance of the FFT. TI recommends keeping this register bit set to 1 (for example, dithering enabled), with appropriate LSFR seed loaded (see the following).
LFSRSEED	29	No	Seed for LFSR (random pattern): For twiddle factor dithering, there is an LFSR that is used, whose seed value is loaded by writing to this 29-bit LFSRSEED register. The LFSRSEED register should be set to any non-zero value, say 0x1234567. To load the LFSR seed, a pulse signal needs to be provided, by writing a 1 followed by a 0 (by setting and clearing) the LFSRLOAD register-bit.
LFSRLOAD	1	No	
FFTCLIP	10	No	FFT Clip Status (read-only): This is a read-only status register, which indicates any saturation/clipping events that have happened in the FFT butterfly stages. Note that each of the 10 butterfly stages in the FFT can be programmed to either saturate the MSB or round the LSB. Whenever saturation of MSB is used in any stage, there is a possibility that that stage can saturate or clip samples. In that case, this saturation event is indicated in the corresponding bit in this status register, so that the Cortex-R4F processor can read it. If multiple FFTs are performed, this status register includes any saturation events happening in any of them. This status register can only be cleared by the R4F, by setting another single-bit register CLR_FFTCLIP, so that the saturation status indication gets cleared back to 0 and any subsequent saturation events can be freshly monitored.
CLR_FFTCLIP	1	No	Clear FFT Clip Status register: This register bit, when set, clears the FFTCLIP register.

Table 6. Core Computational Unit Registers (continued)

Register	Width	Parameter Set	Description
ACCEL_MODE	2	Yes	Select Core Computational Unit Data Path: This register selects the data-path mode of the accelerator's core computational unit – for example, it selects whether the FFT engine path or the CFAR engine path is active. This register will be covered in part two. For the purpose of part one of the user's guide, this register should be zero.
INTERFTHRESH	–	–	Described in part two of this user's guide. For the immediate purposes relevant to part one of this user's guide, all of these registers should be kept as 0.
INTERFTHRESH_EN	–	–	
WINDOW_INTERP_FRACTION	–	–	
CMULT_MODE	–	–	
TWID_INCR	–	–	
STG1LUTSELWR	–	–	
FFT_OUT_MODE	–	–	
FFTSUMDIV	–	–	
MAXn_VALUE	–	–	
ISUMn, ISUMn	–	–	
CFAR_AVG_LEFT	–	–	
CFAR_AVG_RIGHT	–	–	
CFAR_GUARD_INT	–	–	
CFAR_THRESH	–	–	
CFAR_LOG_MODE	–	–	
CFAR_INP_MODE	–	–	
CFAR_ABS_MODE	–	–	
CFAR_OUT_MODE	–	–	
CFAR_GROUPING_EN	–	–	
CFAR_NOISE_DIV	–	–	
CFAR_CA_MODE	–	–	
CFAR_CYCLIC	–	–	
FFTPEAKCNT	–	–	

1.6 Radar Hardware Accelerator – Use Case Example

This section presents a use-case example that illustrates how to configure and use the Radar Hardware Accelerator to achieve some of the frequently used computations in FMCW radar signal processing.

1.6.1 Ultra-Short-Range Radar Use Case

This example illustrates a typical end-to-end radar signal processing flow and how it can be accomplished using the Radar Hardware Accelerator and Cortex-R4F processor. The use case assumes a two-TX, two-RX configuration, with a chirp profile as in Table 7.

Table 7. Chirp Configuration Used for Illustration

Parameter	Value	Comments
Chirp duration	50 µs (active) + 10 µs (idle)	–
Sweep bandwidth	2 GHz	7.5-cm range resolution
Ramp slope	40 MHz/µs	–
Maximum range	15 m	–
Maximum beat frequency	4 MHz	–
ADC sampling rate	4.5 MHz	Complex I,Q sampling
Number of samples per chirp	225	–
First-dimension FFT size	256	225 samples + 31 zeros
Number of chirps per frame	64 × 2 = 128	TX1, TX2 alternating (64 chirps each)
Number of channels	Two TX, two RX	Effective four channels (assuming sparse antenna array with TX's λ-separated and RX's λ / 2-separated)
Radar cube data memory	256 KB	256 × (64 × 2) × 2 × 2 × 2 = 262144 bytes
Frame time	128 × 60 µs = 7.68 ms	–
Second-dimension FFT size	64	Every alternate chirp
Third-dimension FFT size	4 × 2 = 8	Four channels, four zero pads

1.6.1.1 Use Case Illustration – First-Dimension FFT Processing Configuration

During active chirp transmission, the digital front-end (DFE) writes ADC samples to the ADC buffer in ping-pong manner. This example assumes that the ADC data is complex – for example, the RF/Analog is configured as a complex baseband (instead of real-only) chain. The DFE is configured to write two chirps at a time into the ping ADC buffer and two chirps at a time into the pong ADC buffer. This allows effective four parallel channels (two TX and two RX) worth of ADC data to be captured in ping or pong buffer at any time. The DFE configuration details are outside the scope of this user's guide.

To achieve inline, first-dimension, FFT processing using the Radar hardware accelerator, the FFT1DEN register-bit is set, such that the ADC buffer is shared with the accelerator input memories, and the DFE output is directly available to the accelerator for processing at the end of every ping-pong switch.

The data layout in the ADC buffer and in the accelerator local memories after each processing step is shown in Figure 18, Figure 19, Figure 20, Figure 21, Figure 22, and Figure 23. The suffix *0, RX1* means the first ADC sample from the RX1 antenna, *6, RX2* means the seventh ADC sample from the RX2 antenna, and so on. In the following figures, the data corresponding to different chirps is shown in different background colors. All the pictures represent the data layout in either ping or pong memory (for example, 16KB).

$I_{0,RX1}$	$Q_{0,RX1}$	$I_{1,RX1}$	$Q_{1,RX1}$	$I_{2,RX1}$	$Q_{2,RX1}$	$I_{3,RX1}$	$Q_{3,RX1}$
$I_{4,RX1}$	$Q_{4,RX1}$	...	...	...	...	...	...
...	...	...	...	...	...	...	...
$I_{224,RX1}$	$Q_{224,RX1}$						
$I_{0,RX1}$	$Q_{0,RX1}$	$I_{1,RX1}$	$Q_{1,RX1}$	$I_{2,RX1}$	$Q_{2,RX1}$	$I_{3,RX1}$	$Q_{3,RX1}$
$I_{4,RX1}$	$Q_{4,RX1}$	...	...	...	...	...	...
...	...	...	...	...	...	...	...
$I_{224,RX1}$	$Q_{224,RX1}$						
$I_{0,RX2}$	$Q_{0,RX2}$	$I_{1,RX2}$	$Q_{1,RX2}$	$I_{2,RX2}$	$Q_{2,RX2}$	$I_{3,RX2}$	$Q_{3,RX2}$
$I_{4,RX2}$	$Q_{4,RX2}$	...	...	...	...	...	...
...	...	...	...	...	...	...	...
$I_{224,RX2}$	$Q_{224,RX2}$						
$I_{0,RX2}$	$Q_{0,RX2}$	$I_{1,RX2}$	$Q_{1,RX2}$	$I_{2,RX2}$	$Q_{2,RX2}$	$I_{3,RX2}$	$Q_{3,RX2}$
$I_{4,RX2}$	$Q_{4,RX2}$	...	...	...	...	...	...
...	...	...	...	...	...	...	...
$I_{224,RX2}$	$Q_{224,RX2}$						

Figure 18. Layout of Samples in ADC Buffer (Ping)

The first-dimension FFT input is directly picked up from the ADC buffer and it consists of samples from each antenna placed consecutively in memory. The key register configurations required for first-dimension FFT processing are listed in Table 8.

Table 8. Key Register Configurations for First-Dimension FFT

Register	Value	Comments
FFT_EN	1	Enable FFT computation
FFTSIZE	8	FFT size = 2 ^ 8 = 256 225 valid samples + zero padding
SRCACNT	224	225 valid samples (zero-based count)
SRCAINDX	4	Adjacent samples spaced 4 bytes apart
REG_BCNT	1	Two RX antennas processed back-to-back (zero-based count)
SRCBINDX	4096	Samples of RX1 and RX2 are spaced 4KB apart
SRCADDR	0 (parameter sets 0, 3) 1024 (parameter sets 1, 2)	Start at beginning of ACCEL_MEM0 for first chirp. Use another parameter set for second chirp, which starts 1KB away from first chirp. Two additional parameter sets are required for pong operation (this is actually required for destination memory, not for source memory which does automatic ping-pong memory selection).
DSTADDR	32KB (parameter set 0) 32KB + 8B (parameter set 1) 48KB (parameter set 2) 48KB + 8B (parameter set 3)	Destination is MEM2 or MEM3 (ping-pong). There is a separation of 8 bytes between RX1 and RX2 *interlaced* samples. See layout picture for first-dimension FFT output.
DSTAINDX	16	See layout picture for first-dimension FFT output
DSTBINDX	4	See layout picture for first-dimension FFT output
SRC16b32b	0	FFT input samples are 16-bit word aligned
DST16b32b	0	FFT output samples are 16-bit word aligned
TRIGMODE	010b (parameter set 0) 000b (parameter set 1) 010b (parameter set 2) 000b (parameter set 3)	Trigger is based on ping → pong or pong → ping switch. For chained parameter sets to process the two chirps, immediate trigger is used.

Note from Table 8 that the two chirps are processed here using two parameter sets. This is because the separation of ADC samples for the two chirps is different (intentionally kept different in this example, just for illustration) from the separation of the ADC samples for the two RX antennas. Becaue the REG_BCNT and BINDX mechanisms inherently assume uniform spacing, it is not possible to perform the first-dimension FFT processing for both RX antennas (two RX) and both chirps (two TX) using a single parameter set. To avoid this problem, it is possible to place the ADC samples for the two RX antennas 2KB apart (instead off 4KB apart) so that the spacing is uniform, or alternately, just use two parameter sets as done in this example.

The first-dimension FFT processing using ping-pong mechanism is continued for all chirps inline, during active transmission and reception of data. Due to the ping-pong mechanism, another two parameter sets would be required, so that the FFT output memories are also ping-ponged for efficient DMA transfer. In other words, a total of four parameter sets are used for first-dimension processing.

The layout of the first-dimension FFT output samples in the destination memory (accelerator local memories ACCEL_MEM2 and ACCEL_MEM3) are shown in Figure 19. Note that the alignment here is chosen to be *interlaced*, such that the two RX antennas and the two chirps are all interlaced with each other. This allows efficient 128-bit data transfer of each range-bin (FFT bin) to the Radar Cube memory.

$I_{0, RX1}$	$Q_{0, RX1}$	$I_{0, RX2}$	$Q_{0,RX2}$	$I_{0, RX1}$	$Q_{0, RX1}$	$I_{0, RX2}$	$Q_{0,RX2}$
$I_{1,RX1}$	$Q_{1,RX1}$	$I_{1, RX2}$	$Q_{1,RX2}$	$I_{1,RX1}$	$Q_{1,RX1}$	$I_{1, RX2}$	$Q_{1,RX2}$
$I_{2,RX1}$	$Q_{2,RX1}$	$I_{2, RX2}$	$Q_{2,RX2}$	$I_{2,RX1}$	$Q_{2,RX1}$	$I_{2, RX2}$	$Q_{2,RX2}$
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
$I_{255,RX1}$	$Q_{255,RX1}$	$I_{255, RX2}$	$Q_{255,RX2}$	$I_{255,RX1}$	$Q_{255,RX1}$	$I_{255, RX2}$	$Q_{255,RX2}$

Figure 19. Layout of First-Dimension FFT Output Samples in Accelerator Local Memory

1.6.1.2 Use Case Illustration – Radar Data Cube Storage

The first-dimension FFT outputs are stored as 16-bit complex samples in the Radar Data Cube memory, as per the layout shown in Figure 20. Different chirps are shown with different background color.

Because DMA-writes are less efficient for noncontiguous access, the transpose is not yet done, while DMA'ing data into the Radar Data Cube memory. The transpose is done only when reading data back for second and third dimension processing.

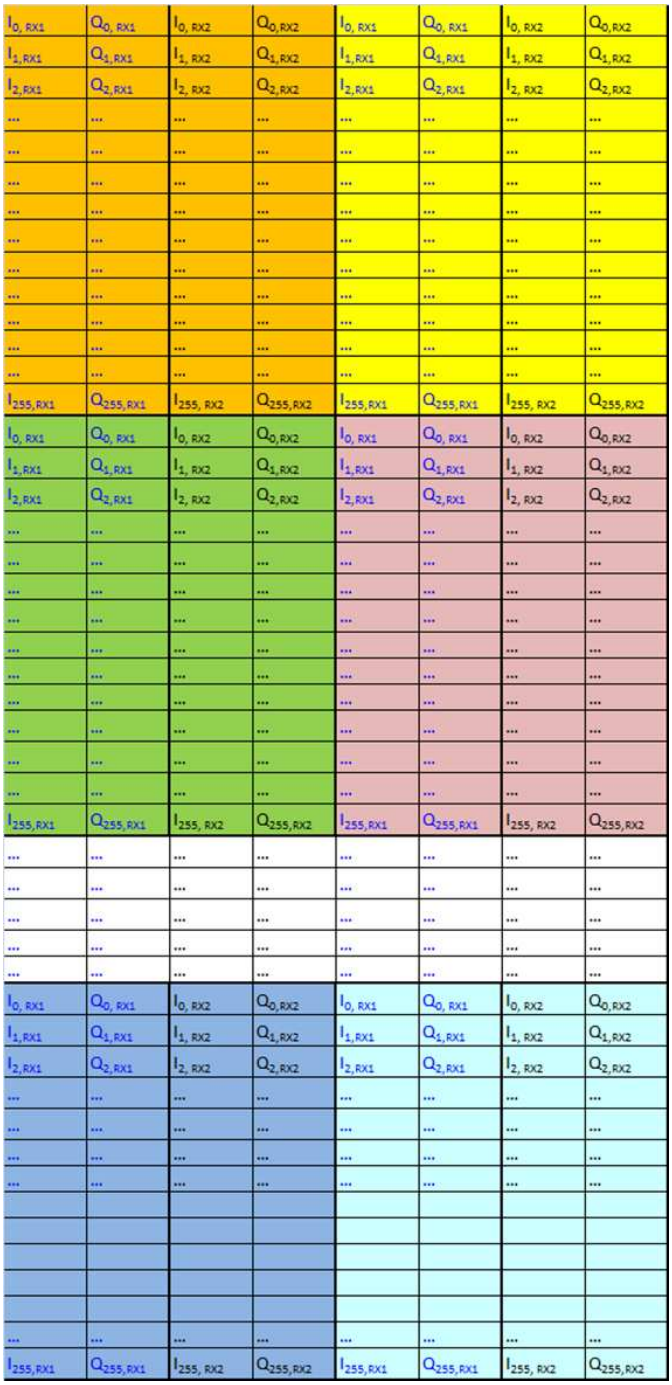


Figure 20. Layout of First-Dimension FFT Output Samples in Radar Data Cube Memory

1.6.1.3 Use Case Illustration – Second (and Third-Dimension) FFT Processing

At the end of active chirp transmission, the ADC buffer is switched over to be directly under the control of the accelerator (FFT1DEN register-bit is cleared). The first-dimension FFT output samples from the Radar Data Cube memory are brought in through DMA to the accelerator local memories (ACCEL_MEM0 and ACCEL_MEM1). This can also be done with a ping-pong mechanism to achieve best overall throughput. The accelerator must be triggered based on (additional) linked DMA channels writing the appropriate register bit into DMA2ACCTRIG to signal completion of the DMA and trigger the appropriate parameter set of the accelerator.

In the example use case shown here, three range-bins are processed at a time for second and third-dimension FFT. This is not really necessary (one range bin can be done at a time instead), but is shown here for the purpose of illustration. Note also that in some implementations, instead of third-dimension FFT, a noncoherent accumulation of the second-dimension FFT output is done across antennas and this result is used for detection. Subsequently, the third-dimension FFT is done only for detected objects. However, the example shown here assumes third-dimension FFT processing is done for all range-velocity cells and the detection is performed only after third-dimension FFT. Another thing to note is that prior to third-dimension FFT, some complex derotations for phase (angle-of-arrival) calibration may be needed. For most practical purposes, this derotation can be achieved inside the DFE and is therefore not needed in the accelerator. However, if desired, this derotation can be achieved through the Complex Vector Multiplication feature in the accelerator (described in part two of the user's guide).

Continuing with the present example, because there are 64 × 2 × 2 data samples for each range bin, the number of bytes per range bin is 1KB. This means that the number of bytes for three range bins is 3KB. Note that as part of the second and third-dimension FFT processing, this 3KB number will grow four times bigger to 12 KB. This is because the second and third-dimension FFT outputs are stored as 32-bit wide samples (2× increase) and also because the third-dimension FFT is zero-padded from 4 to 8 (another 2× increase).

The layout of the data samples at the input to the second-dimension FFT processing is shown in Figure 21. Note that the gaps left in the data sample arrangement in Figure 21 are not really required in the source memory and are shown just for illustration. The three range bins are separated by 5KB each – for example, range 1 starts at 0 KB, range 2 starts at 5KB, and range 3 starts at 10KB.

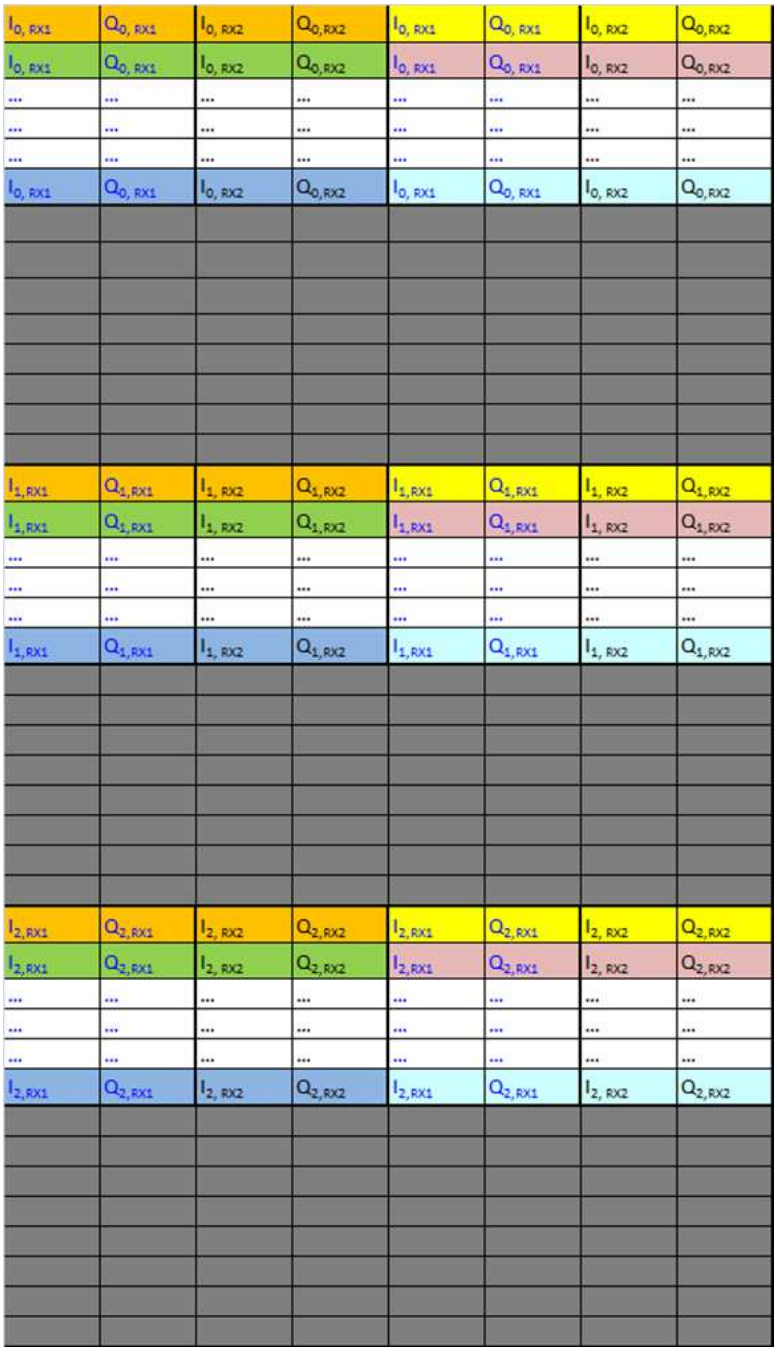
The parameter sets 0, 1, 2, and 3 used for first-dimension FFT processing are overwritten for second and third-dimension processing, which uses 12 parameter sets of its own (for illustration). The reason for using 12 parameter sets here is because of the fact that three parameter sets are required for processing the three range bins, multiplied by a factor of two for second and third dimensions, and another factor of 2 for ping-pong mechanism.

- Parameter sets 0 – 2 → second-dimension FFT processing for three range bins, ping buffer
- Parameter sets 3 – 5 → third-dimension FFT processing for three range bins, ping buffer
- Parameter sets 6 – 8 → second-dimension FFT processing for three range bins, pong buffer
- Parameter sets 9 – 11 → third-dimension FFT processing for three range bins, pong buffer

Table 9. Key Register Configurations for Second-Dimension FFT

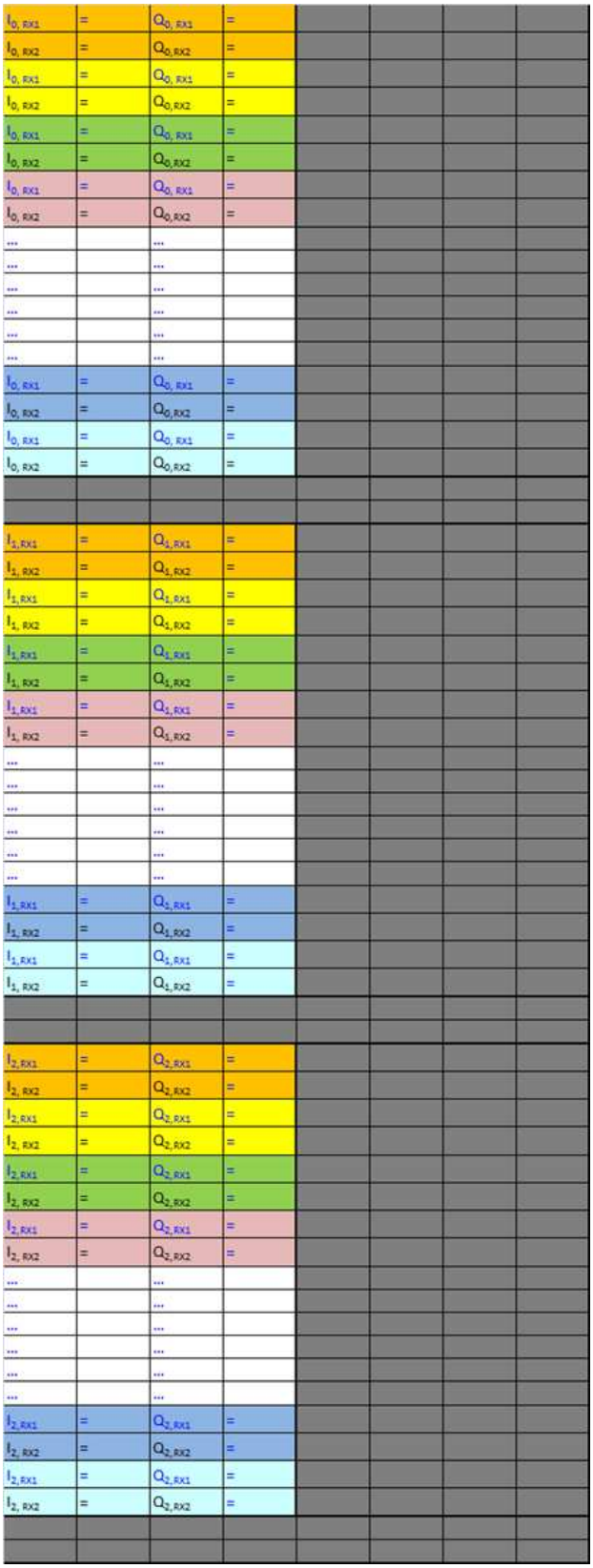
Parameter	Value	Comments
FFT_EN	1	Enable FFT computation
FFTSIZE	6	FFT size = 64 64 chirps for each TX and RX combination
SRCACNT	63	64 samples (no zero padding)
SRCAINDX	16	Adjacent samples spaced 16 bytes apart
REG_BCNT	3	Two TX, two RX processed back-to-back
SRCBINDX	4	See layout picture
SRCADDR	0 (parameter set 0) 5120 (parameter set 1) 10240 (parameter set 2)	Three range bins are processed using three parameter sets. Ping-pong (not shown here) means additional three parameter sets. 5KB separation between range bins is not really required, they can be packed closer at the input memory if required.
DSTADDR	32KB (parameter set 0) 32KB+5KB (parameter set 1) 32KB+10KB (parameter set 2)	Destination is ACCEL_MEM2 (or ACCEL_MEM3 – ping-pong not shown here explicitly). Three range bins separated by 5KB in output memory.
DSTAINDX	64	Refer layout picture for second dim FFT output – (Note that output samples are 32-bits wide.)
DSTBINDX	16	Refer layout picture for second dim FFT output
SRC16b32b	0	FFT input samples are 16-bit word aligned
DST16b32b	1	FFT output samples are 32-bit word aligned
TRIGMODE	011b (parameter set 0, 6) 000b (all others)	DMA-based trigger is used to start the accelerator second-dimension FFT operations
DMA2ACC_CHANNEL_TRIGSRC	0 (parameter set 0) 1 (parameter set 6)	Bit 0 (LSB) of DMA2ACCTRIG is monitored for triggering parameter set 0. Note that a linked DMA channel can be used to set the Bit0 of DMA2ACCTRIG when the DMA transfer into ping memory is complete. Bit 1 of DMA2ACCTRIG is monitored for triggering parameter set 6, which can also be set using a linked DMA channel when the DMA transfer into pong memory is complete.

Note that even though it is not explicitly listed in Table 9, the second-dimension FFT processing uses parameter sets 0, 1, and 2, as well as parameter sets 6, 7, and 8.



(1) Three range bins at a time, in the accelerator local memory

Figure 21. Layout of Second-Dimension FFT Input Samples



(1) 3 range bins at a time in the accelerator local memory – these are stored as 32-bit wide I and Q samples

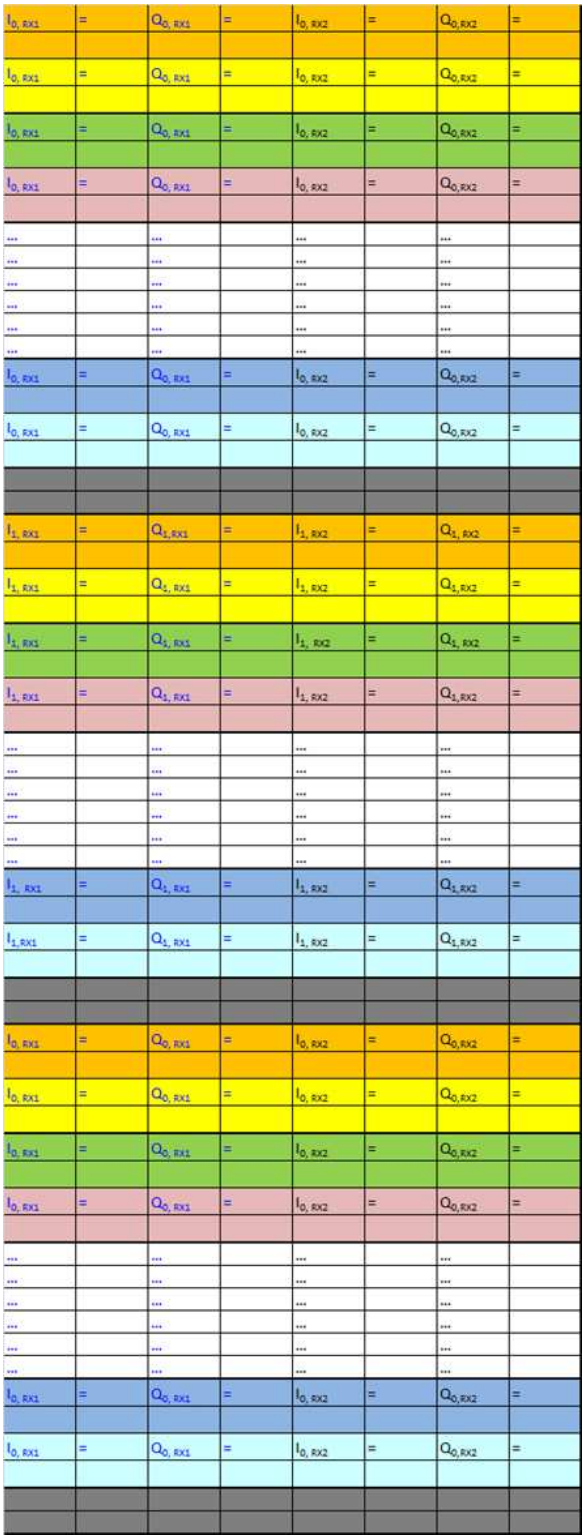
Figure 22. Layout of Second-Dimension FFT Output Samples

The second-dimension processing is immediately followed by third-dimension processing (immediate trigger). Table 10 lists the configuration for this.

Table 10. Key Register Configurations for Third Dimension FFT

Parameter	Value	Comments
FFT_EN	1	Enable FFT computation
FFTSIZE	3	FFT size = 2 ^ 3 = 8 Two TX, two RX gives four channels, add four zeros
SRCACNT	3	Four samples (zero-based count), four zero-pads
SRCAINDX	16	Adjacent input samples spaced 16 bytes apart
REG_BCNT	63	64 chirps to process
SRCBINDX	64	See layout picture
SRCADDR	32KB (parameter set 3) 32KB + 5KB (parameter set 4) 32KB + 10KB (parameter set 5)	Three range bins are processed using three parameter sets. Ping-pong (not shown here) means additional three parameter sets.
DSTADDR	0KB (parameter set 3) 5KB (parameter set 4) 10KB (parameter set 5)	Destination is MEM0 or MEM1 (ping-pong not shown here). Three range bins are separated by 5KB at input and output. Note that the original second-dimension FFT input samples get overwritten here with third-dimension FFT output.
DSTAINDX	8	See layout picture for third-dimension FFT output – (Note that output samples are 32-bits wide.)
DSTBINDX	64	See layout picture for third-dimension FFT output
SRC16b32b	1	FFT input samples are 32-bit word aligned
DST16b32b	1	FFT output samples are 32-bit word aligned
TRIGMODE	000b	Immediate trigger after second dimension.

Note that even though it is not explicitly listed in Table 10, the third-dimension FFT processing uses parameter sets 3, 4, and 5, as well as parameter sets 9, 10, and 11.



(1) 3 range bins at a time in the accelerator local memory – these are also stored as 32-bit wide I and Q samples

Figure 23. Layout of Zero-Padded, Third-Dimension, FFT Output Samples

1.6.1.4 Use Case Illustration – Log-Magnitude Processing

At the end of second and third-dimension FFT processing, the data can either be shipped back to the Radar Data Cube memory or it can be taken through log-magnitude and Cortex-R4F-based detection processing immediately. Alternately, CFAR-CA processing can be done in the accelerator using the CFAR Engine.

In this example, only the log-magnitude processing is assumed to be done in the accelerator and the CFAR (or some other proprietary) detection is assumed to be done in the Cortex-R4F processor. Since the Cortex-R4F must be involved for performing the detection processing anyway, TI recommends having the Cortex-R4F processor interrupted after the third-dimension FFT and log-magnitude processing is done (by setting CR4INTREN for the last parameter set).

The configuration for log-magnitude processing is very straightforward. The Log-Magnitude processing can simply take the input from ACCEL_MEM0 (or ACCEL_MEM1 for pong), compute the log-magnitude, and store the output as 16-bit unsigned numbers in ACCEL_MEM2 (or ACCEL_MEM3 for pong).

Table 11. Key Register Configurations for Log-Magnitude Processing

Parameter	Value	Comments
FFT_EN	0	Disable FFT computation
ABSEN	1	Enable Magnitude computation
LOG2EN	1	Enable Log2 computation
SRCACNT	511	512 samples (each 32-bit I, 32-bit Q) per range bin, where the 512 samples correspond to 64 chirps × 8 angle bins
SRCAINDX	8	Adjacent input samples spaced 8 bytes apart (see the third-dimension FFT output sample layout)
REG_BCNT	2	Three range bins to process
SRCBINDX	5120	Adjacent range bins are spaced 5KB apart. See layout picture.
SRCADDR	0	Source is ACCEL_MEM0 (or ACCEL_MEM1 – ping-pong not shown explicitly here). Three range bins separated by 5KB at input.
DSTADDR	32KB	Destination is ACCEL_MEM2 (or ACCEL_MEM3 – ping-pong not shown explicitly here). Log-magnitude results of three range bins separated by 1KB at output. Note that the original third-dimension FFT input samples get overwritten here with log-magnitude output.
DSTAINDX	2	Log-magnitude output is 16 bits (2 bytes) wide.
DSTBINDX	1024	Log-magnitude results for adjacent range bins are separated by 1KB.
SRC16b32b	1	Input samples are 32-bit word aligned.
DST16b32b	0	Log-magnitude output samples are 16-bit word aligned
DSTREAL	1	Log-magnitude output is real.
TRIGMODE	000b	Immediate trigger after third-dimension FFT.
CR4INTREN	1	Configures hardware accelerator to interrupt the Cortex-R4F processor after completion of log-magnitude calculations.

Note that in this approach, because the Cortex-R4F is being interrupted for performing detection processing at this stage, the ping-pong mechanism for the DMA transfer of second-dimension FFT input from the Radar Data Cube to the accelerator memories, and for the DMA transfer of third-dimension FFT and log-magnitude outputs to the Radar Data Memory, is not really required and therefore only six parameter sets (instead of 12) are required for second and third-dimension FFT processing for the three range bins. The seventh parameter set can be configured for log-magnitude processing, as given in Table 11.

After log-magnitude is computed, the Cortex-R4F must perform CFAR detection processing on the log-magnitude output and identify the cells which have detected objects. This may take up several microseconds for each range bin. Then, for each detected cell, the corresponding third-dimension FFT complex outputs are accessed by the Cortex-R4F from the input memories (ACCEL_MEM0 or ACCEL_MEM1) to estimate the angle of arrival.

2 Radar Hardware Accelerator - Part 2

The second part of the user's guide is organized as follows:

- Section 2.1 covers some additional features of the core computational unit related to pre-FFT processing.
- Section 2.2 covers details of CFAR-CA detection feature.
- Section 2.3 covers other miscellaneous capabilities such as statistics computation are discussed.
- Section 2.4 covers the specific use-case of FFT stitching is described using an example.

2.1 Core Computational Unit – Pre-Processing

This section provides an overview of the pre-processing block inside the core computational unit. Specifically, the features of interference zero-ing out and complex multiplication.

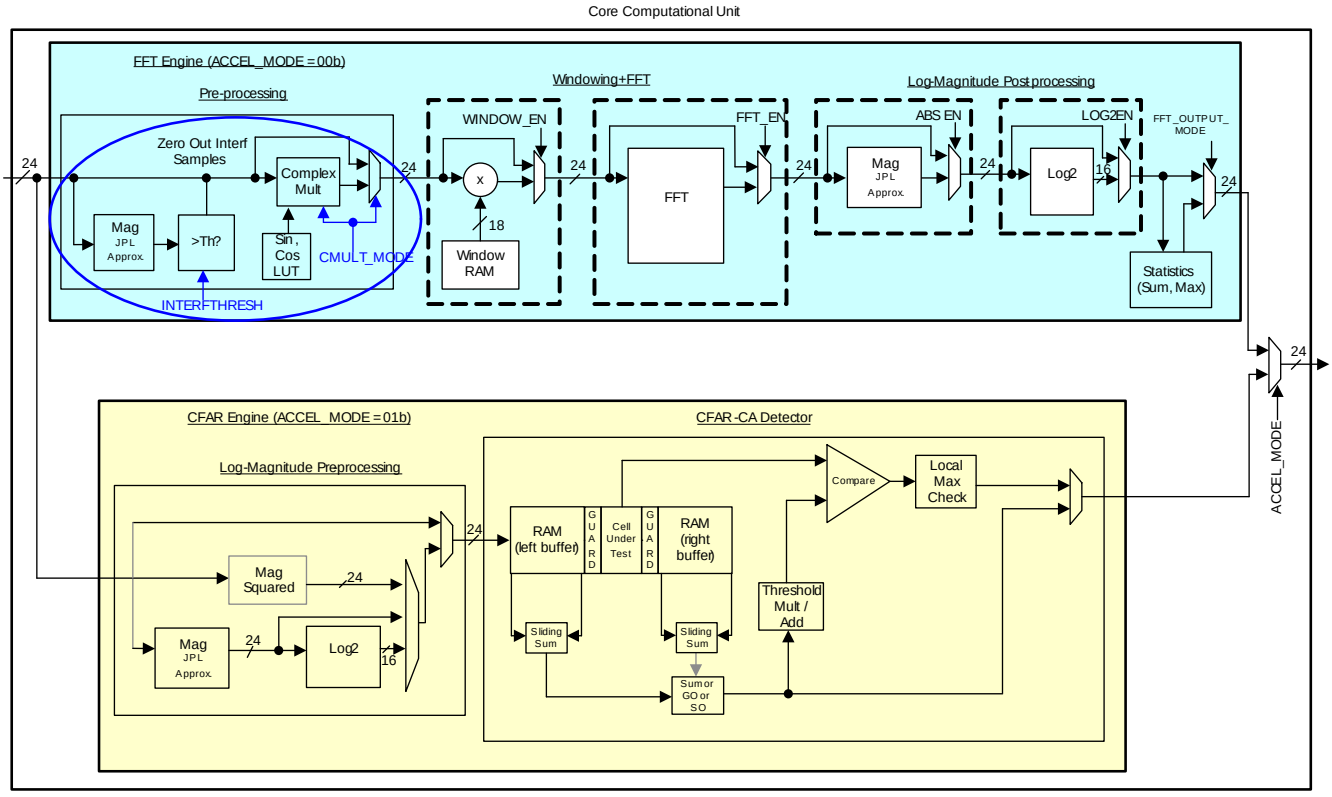


Figure 24. Core Computational Unit

2.1.1 Pre-Processing Block

The pre-processing block (see Figure 24) provides the ability to perform some simple pre-FFT manipulations of the samples. As shown in Figure 24, the pre-processing block, the windowing block, FFT block and log-magnitude post-processing block are connected to each other, so that the output from one block can be fed into the next block, if required. One or more of these blocks can be enabled in each parameter-set to perform the desired operations.

The pre-processing block comprises provision for interference zero-out and complex multiplication. These sub-blocks are described in the following sections.

2.1.1.1 Interference Zero-Out

In an FMCW radar transceiver, interference from another radar typically manifests itself as a time-domain spike in a few samples. This spike corresponds to the time duration when the chirping frequency of both radars overlap with each other. Such a time-domain spike caused by interference can lead to degradation in the noise floor at the FFT output, causing loss of detection.

In order to mitigate the impact of interference, one common technique is to zero-out any large glitches in the time-domain samples. The pre-processing block provides capability to perform this operation. Specifically, the input samples are fed through a magnitude calculation (based on JPL approximation), which computes a 24-bit magnitude of the 24-bit input complex sample. For definition of this approximation, see part 1 of this user's guide Section 1.5.1.5. Any sample whose magnitude exceeds a programmable threshold is zero-ed out, both in real and imaginary part. The programmable threshold is a 24-bit register named INTERFTHRESH. This threshold register is not part of the parameter-set and is a common register. Note that a value of 0xFFFFFF can be programmed to disable the zero-out feature, since the magnitude can never exceed this value anyway. Alternately, a separate INTERFTHRESH_EN register is provided as part of the parameter-set to control when the interference zero-ing out should be enabled.

Note the limitation that the threshold is a static configuration and there is no specific provision in the accelerator to automatically calculate the threshold based on the RMS of the current set of data samples. However, there are possible ways of accomplishing this indirectly, by performing a first-pass of the input samples to compute the signal statistics (with a bit-shift scaling on the statistic) and then using the main processor or DMA to copy this scaled statistic value to the interference threshold register, so that in a subsequent second pass, the interference zeroing out can be accomplished. On a separate note, if the user is interested just to know the indices of the time-domain spikes, then it is possible to use the CFAR engine path to look for spikes in the samples and record the indices of the spikes. The CFAR engine is covered in Section 2.2.

2.1.1.2 Complex Multiplication

In addition to interference zero-out, the pre-processing block contains a complex multiplication sub-block. The purpose of this sub-block is to enable several assorted capabilities that require complex multiplication of the input samples. The CMULT_MODE register is used to enable and configure the complex multiplication functionality. The complex multiplication sub-block can be disabled (bypassed) by the setting cmult_mode to 000b. any other value of this register will enable the complex multiplication sub-block and configure it to perform specific operation as described in the next few paragraphs.

There are seven modes of the complex multiplier supported, as follows. They are frequency shifter mode, frequency shifter with auto-increment mode (a slow DFT mode), FFT stitching mode, magnitude squared mode, scalar multiplication mode, vector multiplication modes 1 and 2.

- Frequency shifter mode: If the register value is CMULT_MODE = 001b, then the complex multiplier functions as a frequency shifter, which can be used to de-rotate the input samples by a certain frequency. This de-rotation is accomplished using cos, sin values from a twiddle factor look-up table (LUT). This LUT contains the (compressed) equivalent of the cos, sin values corresponding to the 16384 long sequence exp(j*2*pi*(0:16383)/16384). Another register (TWIDINCR) is used to specify the de-rotation frequency, by specifying how much the phase should change for each successive input sample (that register controls how much the LUT read index increments every sample). In effect, the input samples x(n) for n = 0 to SRCACNT-1 are multiplied by the sequence, exp(j*2*pi*TWIDINCR*(0:SRCACNT-1)/16384).

- Frequency shifter with auto-increment mode (a slow DFT mode): If the register value is CMULT_MODE = 010b, then the complex multiplier functions in a mode which enables Discrete Fourier Transform (DFT) computation. In this case, the complex multiplier performs a function that is very similar to frequency shifter mode, except that, at the end of each iteration, the de-rotation frequency is automatically incremented for the next iteration. Note that DFT computation for a given set of input samples involves de-rotating the samples by one frequency at a time, and computing a sum of the de-rotated samples for each such frequency. To achieve DFT computation, the Input Formatter should be configured to send the same set of input samples to the complex multiplier for multiple iterations (as many as the number of DFT bins required) and the complex multiplier de-rotates the samples by one frequency at a time and auto-increments to the next frequency for the next iteration. Also, the statistics block (explained in a later section) is used to compute the sum of the de-rotated samples corresponding to each iteration, which then becomes the final DFT value.

  The DFT computation is 'slow' in the sense that in each 200 MHz clock cycle, only one complex multiplication is performed. For example, for a 512-point input sample set, it would take 512 clock cycles per DFT bin. However, since the DFT mode is typically only used for FFT peak interpolation (very few bins), it is acceptable. The starting frequency for the DFT computation is specified in the TWIDINCR register (similar to the frequency shifter mode). The increment value by which the frequency increments every iteration is obtained from FFTSIZE register – Note that the DFT mode cannot be used simultaneously with FFT enabled, hence the FFTSIZE register has been over-loaded for providing the increment value in this mode. The increment value is calculated as 2^(14 – FFTSIZE) and hence the DFT resolution is 16384/2^(14 – FFTSIZE) = 2^FFTSIZE. As an example, if FFTSIZE = 1011b, then the DFT resolution is 2048. This is equivalent to computing DFT points corresponding to 2K size FFT grid. The highest resolution for the DFT would be obtained when FFTSIZE = 1110b (max allowed value), in which case the DFT resolution is 16384 (corresponding to 16K size FFT grid).

  In effect, for the k^{th} iteration (with k starting from 0), the input samples x(n) for n = 0 to SRCACNT-1 are multiplied by the sequence, exp(j*2*pi*(TWIDINCR+2^(14-FFTSIZE)*k)*(0:SRCACNT-1)/16384).
- FFT Stitching mode: If the register value is CMULT_MODE = 011b, then the complex multiplier functions in FFT stitching mode. This mode is useful when large size FFTs (2K and 4K) are required. Since the FFT block natively supports only up to 1024 size, for 2048 and 4096 point FFT, an FFT Stitching procedure using two steps (two parameter-sets) can be used. As an example, when a 4K size FFT is needed, it is achieved in two steps as follows. In the first step, every 4th input sample is passed through a 1K size FFT (four 1K point FFTs are performed on decimated input samples). Then, in the next step, the resulting 4x1024 FFT outputs are sent through four-point "stitching" FFTs (1024 four-point FFTs), with an additional pre-multiplication by the complex multiplier block to achieve FFT stitching. This pre-multiplication uses the twiddle factor LUT in a specific pattern, for which additional configuration information is available in TWIDINCR register (2 LSB bits). If the LSB two bits of TWIDINCR register are 00b, then the twiddle factor pattern will correspond to what is required for 2K (2x1024) size FFT stitching. If the LSB two bits are 01b, then the twiddle factor pattern will correspond to what is required for 4K (4x1024) size FFT stitching. Values of 10b and 11b are reserved and should not be used. Also, the unused 12 MSB bits of TWIDINCR register must be zero in this mode of operation. The last section includes a more detailed explanation and configuration information for the FFT stitching example for 2K and 4K FFT, including the use of WINDOW_INTERP_FRACTION register for extending the window RAM using linear interpolation to more than 1024 coefficients.
- Magnitude squared mode: If the register value is CULT_MODE = 100b, then the complex multiplier functions in magnitude squared mode. In this case, the complex multiplier takes every complex input and produce the magnitude squared as the output. This can be used together with the statistics block (explained in Section 2.3) to compute the mean squared sum of the input samples.
- Scalar multiplication mode: If the register value is CMULT_MODE = 101b, then the complex multiplier functions in scalar multiplication mode. This feature is useful if the input samples need to be scaled by some constant factor. In this case, the complex multiplier will multiply each input sample with a 21-bit scalar complex number that is programmed in ICMULTSCALE and QCMULTSCALE registers (for I and Q value, each having 21 bits). The ICMULTSCALE and QCMULTSCALE registers are common registers and not part of parameter-set. Note that this feature cannot be used to multiply the input samples for different iterations (channels) with different complex scalars.

- Vector multiplication mode 1: If the register value is CMULT_MODE = 110b, then the complex multiplier functions in vector multiplication mode 1. The purpose of this mode is to enable element-wise multiplication of two complex vectors, as well as dot-product capability (using statistics block to sum the element-wise multiplication output). The samples from the Input Formatter block constitute one of the two vectors, whereas the other vector is taken from a pre-loaded internal RAM inside the core computational unit. (This internal RAM is a RAM that is normally used by the FFT block when performing 1024-point FFT computation). This internal RAM can store 512-complex samples and hence the vector multiplication can support a maximum of 512 elements of multiplication. The Vector multiplication is not a highly parallelized operation, in the sense that only one complex multiplication is done per 200MHz clock cycle.

  It is important to note one important limitation: since the internal RAM is shared with the FFT, performing a 1024-point FFT destroys the pre-loaded contents of the RAM. Therefore, performing vector multiplication and 1024-point FFT back-to-back many times requires re-loading of the internal RAM each time and will be inefficient. However, note that this limitation of having to re-load the internal RAM does not apply when performing FFT of size 512 or less, which is often the case for second and third dimension FFTs.

  The operation of the vector multiplication mode 1 is as follows. The streaming set of samples from the Input Formatter block coming at 200 MHz is element-wise multiplied with successive samples from the internal RAM. The statistics block (described in a later section) can be used to compute the sum for every iteration, which enables a dot-product implementation if desired. At the end of every iteration, the addressing from the internal RAM is reset, so that for the next iteration, the samples are picked up from the start of the internal RAM.
- Vector multiplication mode 2: If the register value is CMULT_MODE = 111b, then the complex multiplier functions in vector multiplication mode 2, which is slightly different from the earlier mode. The only difference in this case is that at the end of every iteration, the addressing of the internal RAM is not reset, so that for the next iteration, the samples from the internal RAM are picked up with an address that continues from where it left off at the end of the previous iteration. This mode can be used when a given set of input samples needs to be element-wise multiplied with multiple vectors. In this case, the input formatter block can be configured to repeat the same set of samples for multiple iterations, and the internal RAM can be loaded with all the vectors, such that for successive iterations, the input samples are multiplied with successive vectors.

For loading the internal RAM used for the vector multiplication modes, the register bit STG1LUTSELWR is used. The internal RAM for vector multiplication mode is mapped to the same address space as the Window RAM. Therefore, this register bit is required to specify which of these two (Window RAM or internal RAM) need to be selected, when loading the co-efficients via DMA or main processor. If the register bit is 0, then the Window RAM is selected, else, the internal RAM for vector multiplication mode is selected. Note that the other registers such as WINDOW_START, which pertains to windowing, are always applicable only for the Window RAM. The STG1LUTSELWR register bit should in general be kept as 0 (Window RAM selected). This allows the main processor or DMA to have access to the Window RAM by default. Only when it is desired to load the internal RAM with coefficients for vector multiplication mode, this register bit should be temporarily set to 1. After loading the coefficients, the register bit should be made 0 again.

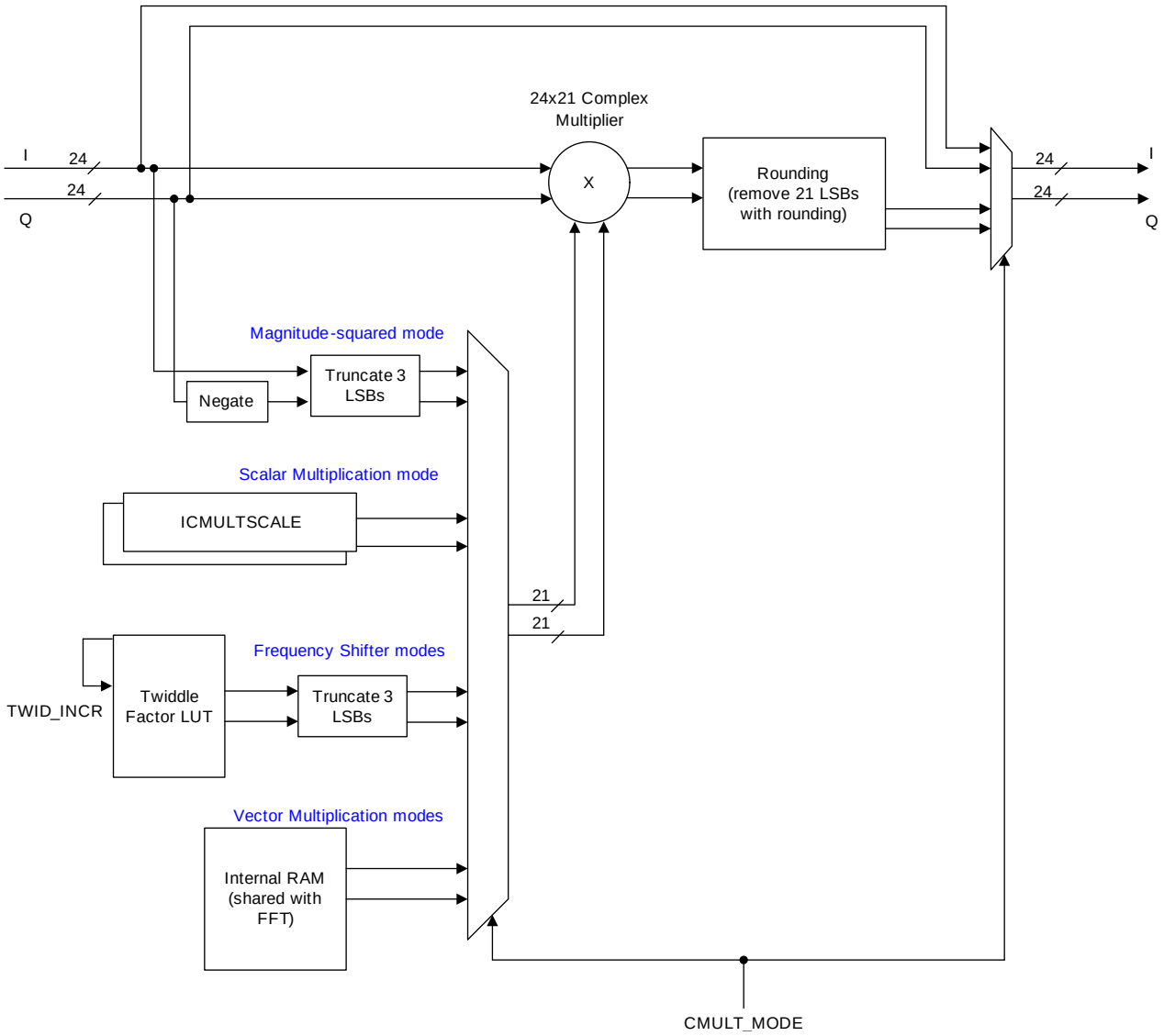


Figure 25. Complex Multiplication Capability in Pre-Processing Block

Note that in all the above seven modes of the complex multiplier, only one complex multiplication is performed every 200 MHz clock. So, the effective speed achieved for the multiply or multiply-accumulate operation is 200 MHz.

2.1.1.3 BPM Removal

Although not explicitly shown in Figure 24, it is possible to multiply the input samples going from the input formatter into the core computational unit with a +1/-1 programmable binary sequence (of length up to 64). This feature is enabled by setting the register bit BPM_EN in the parameter-set.

This feature may be useful when Binary Phase Modulation (BPM) is used during transmission of chirps. The BPM pattern is generally a pseudo-random sequence (chipping sequence) of 1's and -1's, which have already been applied to the radar transmit signal. Therefore, the radar signal processing of the resultant analog-to-digital converter (ADC) samples prior to FFT needs to undo the modulation. For instance, if each chirp is transmitted with a +1 or -1 polarity, then it is necessary to undo this sequence prior to the second dimension FFT processing across chirps. The BPM removal feature can be used to achieve this.

NOTE: An alternate way to achieve this is to pre-multiply the window coefficients, which are signed numbers, in the window RAM, so that the process of windowing prior to FFT takes care of undoing the BPM sequence.

When BPM removal is enabled, each input sample is multiplied by a +1 or -1, based on the sequence present in the 64-bit BPMPATTERNLSB and BPMPATTERNMSB register. The register BPMRATE is used to control for how many consecutive samples the same BPM bit is applied. For example, if BPMRATE = 4, then the same BPM bit is applied for 4 consecutive samples. Similarly, if BPMRATE = 1, then the BPM bit is changed for every sample.

There is another register BPMPHASE that specifies the number of consecutive samples for which the first BPM bit is applied. Note that this is applicable only for the first BPM bit. If BPMPHASE = 0, then the first BPM bit is applied for BPMRATE number of samples. Otherwise, the first BPM bit is applied for BPMRATE – BPMPHASE number of samples. For example, if BPMPHASE = 1 and BPMRATE = 4, then the first BPM bit is applied for 4-1 = 3 samples, and then subsequent BPM bits are applied with periodicity of 4 samples for each bit. This is shown in Figure 26.

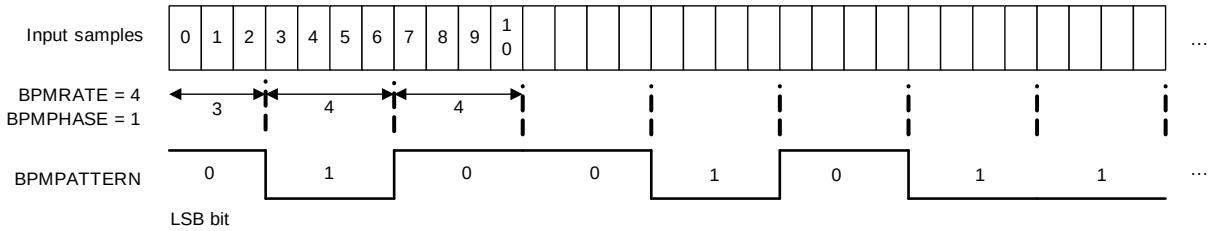


Figure 26. BPM Removal Capability

If multiple iterations (for example, four back-to-back FFTs in a single parameter-set using REG_BCNT=3) are done, then the same BPM pattern gets applied to the input samples in each iteration.

Note the limitation that the BPM pattern register is 64 bits long, hence, the maximum BPM sequence length that is supported is 64. For higher BPM sequence length, the alternate approach of pre-multiplying the window coefficients stored in the window RAM may be considered.

2.1.1.4 Pre-Processing Block – Register Descriptions

Table 12 lists all the registers of the pre-processing block. As explained in part 1 of this user's guide, some of the registers are common (common for all parameter-sets) registers, whereas, some others are “part of each parameter-set”. For each register, this distinction is captured as part of the register description in Table 12.

Table 12. Pre-Processing Block Registers

Register	Width	Parameter-Set? (Y/N)	Description
INTERFTHRESH .	24	N	Interference threshold: This register is used to specify the threshold for zero-ing out samples affected by interference. Any sample whose magnitude exceeds this threshold is zero-ed out, if the feature is enabled using INTERFTHRESH_EN register bit
INTERFTHRESH_EN	1	Y	Interference zero-out Enable/Disable: This register bit controls the enable/disable for the interference zero-out feature. The feature is enabled if this register bit is set in any given parameter-set.
CMULT_MODE	3	Y	Complex multiplication mode: This register is used to configure the mode of the complex multiplication sub-block. A value of 000b disables/bypasses the complex multiplication. Any other value chooses one of 7 available modes of operation . Detailed description of the seven modes in the main description section.

Table 12. Pre-Processing Block Registers (continued)

Register	Width	Parameter-Set? (Y/N)	Description
TWIDINCR	14	Y	Twiddle factor configuration: When the complex multiplication sub-block is programmed in one of the frequency shifter modes (CMULT_MODE = 001b or 010b), this register is used to indicate the amount of frequency shift. Specifically, the input samples x(n) for n = 0 to SRCACNT-1 are multiplied by the sequence: exp(j*2*pi*TWIDINCR*(0:SRCACNT-1)/16384). When the complex multiplication sub-block is programmed in FFT stitching mode (CMULT_MODE = 011b), the last two bits of this register specify whether it is 2K or 4K FFT stitching. Specifically, if the last two bits are 00b, then it is 2K (2*1024-point) FFT stitching and if the last two bits are 01b, then it is 4K (4*1024-point) FFT stitching. Values of 10b and 11b are reserved. Also, the 12 MSB bits of this register must be zero. In all other modes of the complex multiplication sub-block, this register must be kept as 0.
FFTSIZE	4	Y	FFT size (DFT mode): For the register description during normal FFT operation, see part 1 of this user's guide. This register also has a second purpose in DFT mode as described below. When FFT is disabled, and complex multiplication mode is set to be frequency shifter with auto-increment (slow DFT mode), then this register is used to specify the DFT bin resolution. The DFT bin resolution is given by 2^FFTSIZE. For example, if FFTSIZE = 1011b, then the DFT resolution is 2048 (2K size).
STG1LUTSELWR	1	N	Select Window RAM or internal RAM: The internal RAM for vector multiplication mode is mapped to the same address space as the Window RAM. Hence, this register bit is required to specify which of these two needs to be selected when loading the co-efficients via DMA or R4F. If this register bit is 0, then the Window RAM is selected, else, the internal RAM for vector multiplication mode is selected. Keep this register bit as 0 always, except during the period when internal RAM needs to be loaded.
BPM_EN	1	Y	Enable/Disable BPM removal: This register bit specifies whether the BPM removal needs to be enabled or not. If this register is set, then BPM removal is enabled prior to feeding samples from the input formatter into the core computational unit.
BPMPATTERNLSB and BPMPATTERNMSB	64	N	BPM pattern: Specifies the BPM pattern to be used to multiply the input samples if BPM removal is enabled.
BPMRATE	10	N	BPM rate: Specifies the number of input samples corresponding to each BPM bit. Minimum valid value for this register is 1.
BPMPHASE	4	Y	BPM starting phase: Specifies the starting phase of the BPM pattern periodicity. For more information, see the detailed description in Section 2.1.1.3.

2.2 Core Computational Unit - CFAR Engine

This section describes the CFAR engine block present in the core computational unit.

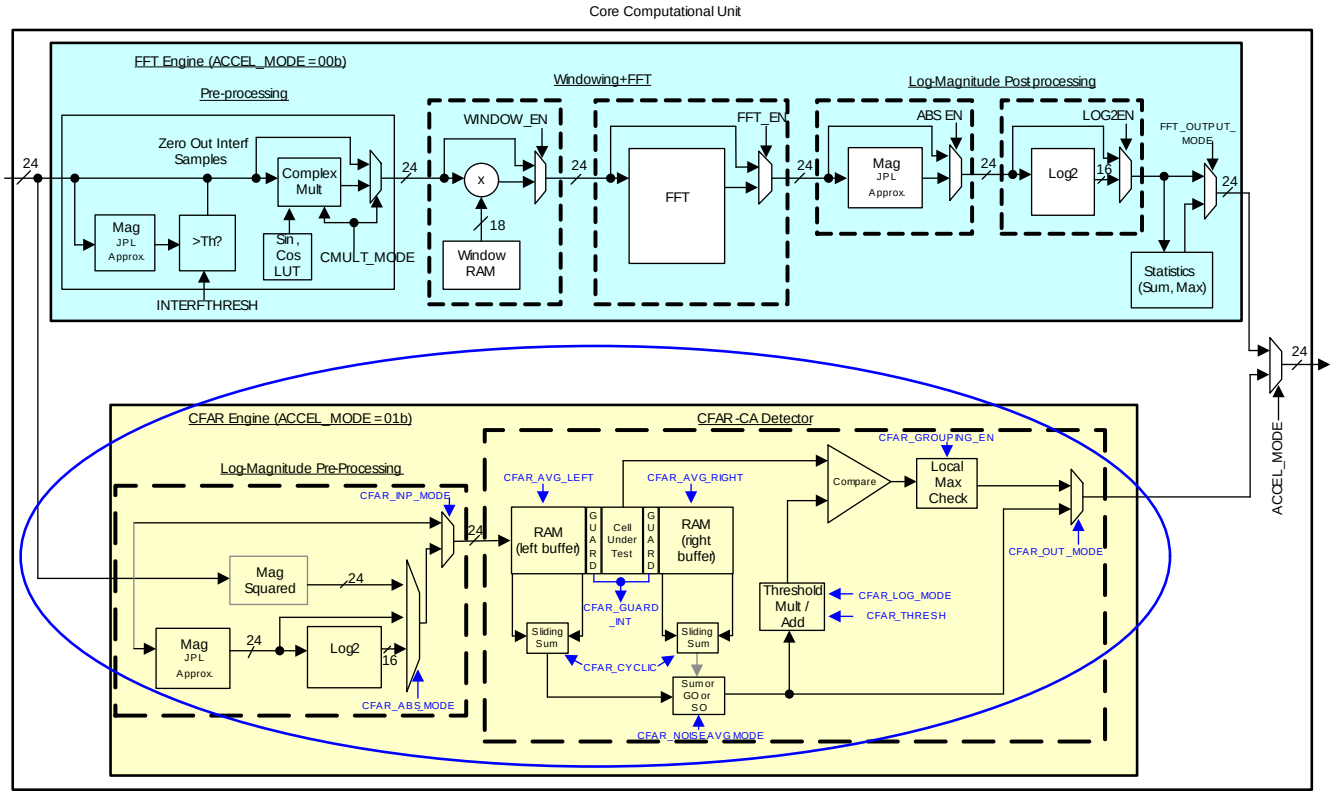


Figure 27. CFAR Engine

2.2.1 CFAR Engine

The CFAR engine (see Figure 27) is a module that enables detection of objects, by identifying peaks in the FFT output. Although there are several detection algorithms, the accelerator supports only the CFAR-CA algorithm and a few variants of it. CFAR-CA stands for constant false alarm rate – Cell Averaging. The other popular technique known as ordered-statistic CFAR, a.k.a CFAR-OS is not supported.

As shown in Figure 27, the CFAR engine path is selected by setting the accelerator mode ACCEL_MODE = 01b. In this mode, the FFT path is not usable simultaneously and the input 24-bit samples from the input formatter block will be routed into the CFAR engine. The CFAR engine has capability to perform CFAR-CA detection processing (both linear and logarithmic CFAR modes are available) and generate a peak list.

In cell-averaging CFAR, the processing steps involve computing a threshold for each sample under test (cell under test) and deciding whether a peak is detected or not based on whether the cell under test crosses that threshold. Additionally, peak grouping may be done, where a peak is declared only if the cell under test is greater than its most immediate neighboring cells to its left and right. One thing to note here is that for peak grouping, the left and right neighboring cells themselves are not required to be CFAR qualified.

For each cell under test, the computation of threshold is done by averaging the magnitude (or magnitude-squared or log-magnitude) of a specified number of noise samples to the left and right of the cell under test to determine a 'surrounding noise average' and then applying a scale factor (or addition factor in case log-magnitude is used) on that surrounding noise average to determine the threshold. Thus, the CFAR-CA detector takes one cell at a time, computes the threshold and decides whether a valid peak is present at that cell.

2.2.1.1 CFAR Engine – Operation

The CFAR engine receives 24-bit input samples from the Input Formatter block. Typically, these are real samples, representing the magnitude or magnitude-squared or log-magnitude of the FFT output. However, the input to CFAR engine can instead be complex samples, in which case, either magnitude or magnitude-squared or log-magnitude of the complex samples can be computed inside the CFAR engine itself. This is done by the log-magnitude pre-processing sub-block inside the CFAR engine (see Figure 28). The real unsigned result from this pre-processing operation is sent to CFAR-CA detection processing. The registers CFAR_INP_MODE and CFAR_ABS_MODE are used to configure real vs. complex input, as well as the nature of pre-processing required. The log-magnitude computation uses the same JPL approximation for magnitude calculation and the same look-up table (LUT) approximation for log2 computation as described in Part 1 of the user guide for FFT engine post-processing.

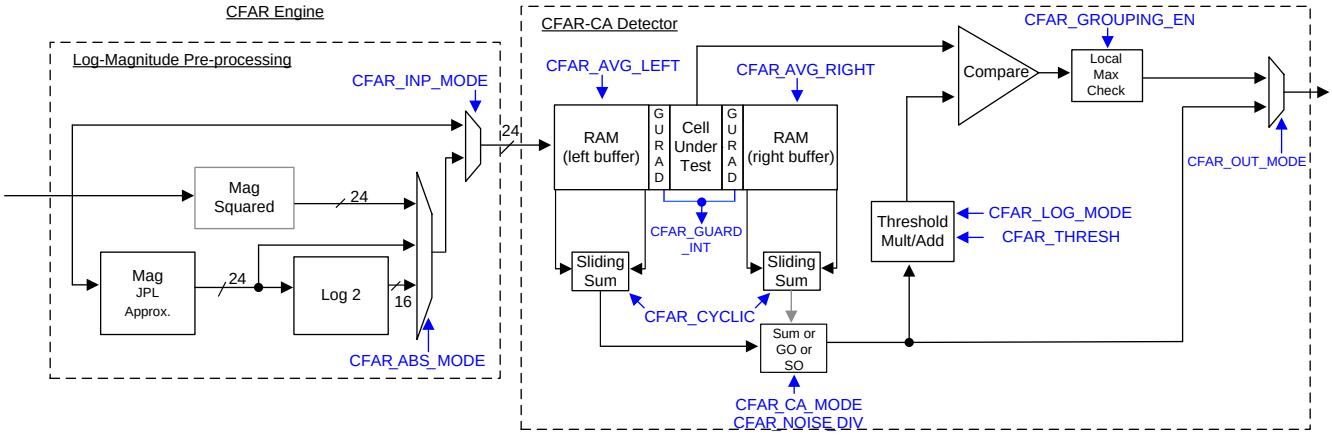


Figure 28. CFAR Engine Block Diagram

As described earlier, the CFAR-CA detection processing involves finding a “surrounding noise average” for each cell under test and then determining a threshold that is a function of the surrounding noise average. The cell under test is compared against this threshold to decide whether a peak is present or not in that cell. To calculate the threshold, the surrounding noise average is multiplied with (or added to) a threshold scaling factor specified in CFAR_THRESH register.

There are two modes in which the CFAR detector can be used – in non-logarithmic mode (a.k.a linear CFAR), the threshold scale factor is multiplied, and in logarithmic mode (a.k.a logarithmic CFAR), the threshold scale factor is added. This is decided based on CFAR_LOG_MODE register.

The final detection threshold that is so obtained is used to compare against the cell under test to determine whether a peak is detected in that cell.

Table 13 summarizes the register settings for the different CFAR modes of operation.

Table 13. CFAR Modes and Register Settings

Desired CFAR Mode	Input Real or Complex	Desired Pre-Processing	Register Values to Use		
			CFAR_INP_MODE	CFAR_ABS_MODE	CFAR_LOG_MODE
Linear CFAR	Real	N/A	1	00	0
	Complex	Magnitude	0	10	0
		Mag-squared	0	00	0
		Log2-Mag	0	11	0
Log CFAR	Real	N/A	1	00	1
	Complex	Log2-Mag	0	11	1

Desired CFAR-CA Algorithm	CFAR_CA_MODE Register Setting
CFAR-CA	00
CFAR-CAGO	01
CFAR-CASO	10

The surrounding noise average computation has multiple options – cell averaging (CFAR-CA), cell averaging with greater-of selection (CFAR-CAGO) and cell averaging with smaller-of selection (CFAR-CASO). The register CFAR_CA_MODE is used to select between CFAR-CA, CFAR-CAGO and CFAR-CASO modes. In CFAR-CA, the noise samples on the left side and right side of the cell under test (after ignoring some guard cells on either side) are simply averaged to determine the surrounding noise average value. In CFAR-CAGO, the noise samples on the left side and right side are averaged independently and the greater of the two is used to determine the threshold. In CFAR-CASO, the lesser of the two is used

The number of samples on the left side and right side used for computing the noise average is configured using CFAR_AVG_LEFT and CFAR_AVG_RIGHT registers and the number of guard cells is configured using CFAR_GUARD_INT register. The number of samples used for left side noise averaging is given by 2*CFAR_AVG_LEFT. The number of samples used for right side noise averaging is given by 2*CFAR_AVG_RIGHT. The number of guard cells that are ignored on each side of the cell under test is given by CFAR_GUARD_INT. For example, if CFAR_AVG_LEFT = CFAR_AVG_RIGHT = 16, and CFAR_GUARD_INT = 3, then it means that the most immediate three samples each to the left and right of the cell under test are skipped and then, 32 samples on the left and 32 samples on the right side are used for noise averaging. Note that even though the term noise averaging is used here, the actual implementation simply adds the noise samples first and the “averaging” is done as a divide by a power-of-2 as specified in a separate register, CFAR_NOISE_DIV. These registers are described in Table 16.

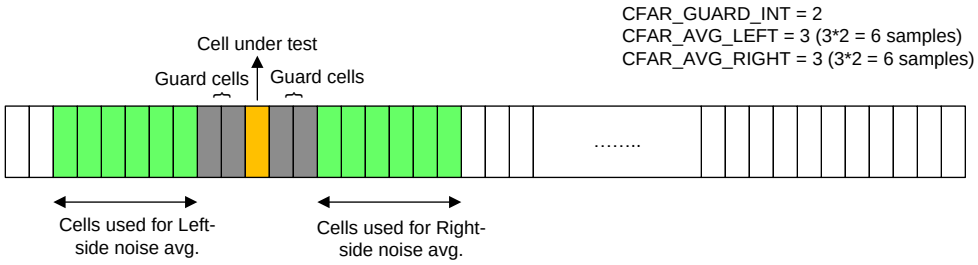


Figure 29. CFAR-CA: Cells Used for Surrounding Noise Average

As mentioned earlier, the CFAR_THRESH register specifies the threshold scaling factor. This is an 18-bit register whose value is used to either multiply or add to the ‘surrounding noise average’ to determine the threshold used for detection of the present cell under test. If logarithmic mode is disabled (in magnitude or magnitude-squared mode), then the register value is multiplied with the surrounding noise average to determine the threshold, else it is added to the surrounding noise average. In the former case, this 18-bit register is interpreted as a 14.4 value and supports a range of values from 1/16 to 2^14-1. In the latter case (logarithmic mode), the 18-bit register is interpreted as a 7.11 value.

The CFAR engine supports a few output formats that are described next.

2.2.1.2 CFAR Engine – Output Formats

The cells that exceed the threshold are noted and this ‘Detected Peaks list’ is sent to the destination memory. Since the output format of the core computational unit is 24-bits I and 24-bits Q, the detected peaks list is formatted into ‘I’ and ‘Q’ channels as shown in Figure 30. The 24-bit I channel contains the index at which the peak is detected, with the MSB 12 bits containing the iteration number (corresponding to REG_BCNT counter value) and the LSB 12 bits containing the sample index number (corresponding to SRCACNT counter value). The 24-bit Q channel contains the surrounding noise value or the cell under test value of that detected peak. This is chosen based on CFAR_OUT_MODE register setting. Instead of ‘Detected Peaks list’, it is also possible for the CFAR engine to send out the raw ‘surrounding noise average’ value for each cell. This is called ‘Raw output mode’.

Figure 30 and Table 14 show the different output formats available.

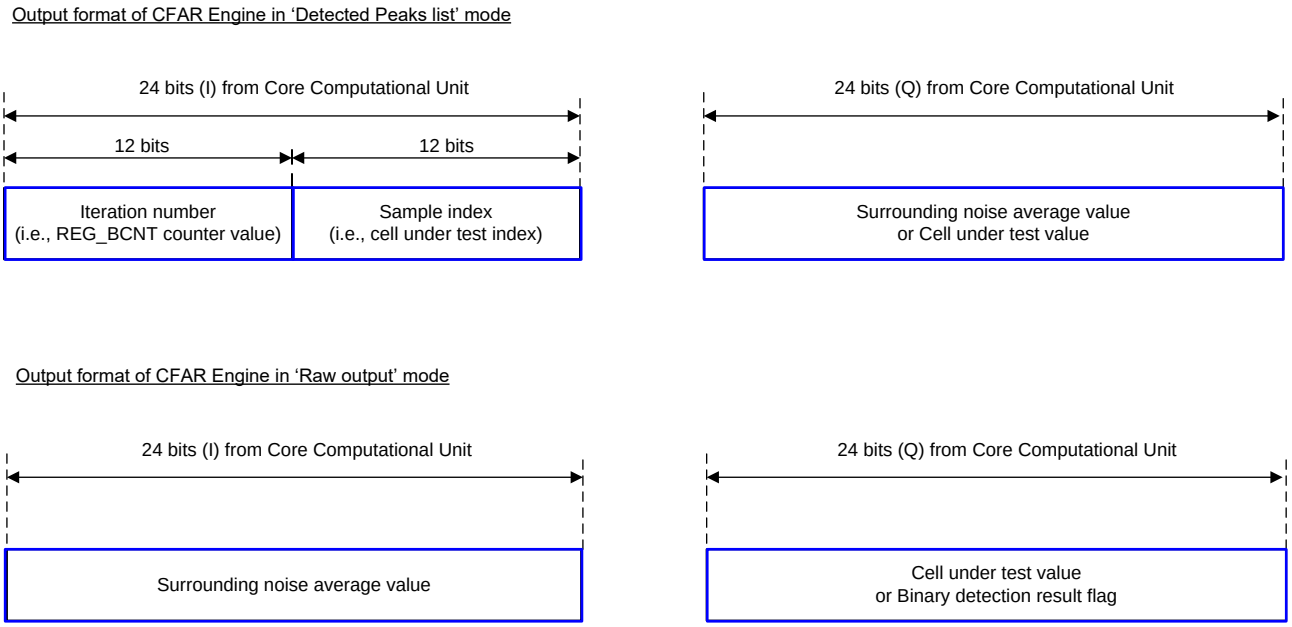


Figure 30. CFAR Engine Output Format

Table 14. CFAR Output Modes and Register Settings

CFAR Output Mode	I Channel Output	Q Channel Output	CFAR_OUT_MODE Register Setting
Detected peaks list mode (only detected peaks are output)	Peak index	Surrounding noise average value	10
	Peak index	Cell under test value	11
Raw output mode (all cells are output)	Surrounding noise average value	Cell under test value	00
	Surrounding noise average value	Binary detection result flag (0 or 1)	01

In detected peaks list mode, only the detected peaks are output to the destination memory. In this case, the read-only register FFTPEAKCNT indicates how many peaks have been totally detected, so that the main processor can read that many locations from the destination memory.

While detecting peaks, if 'peak grouping' is required, then it can be enabled using CFAR_GROUPING_EN register. In this case, a peak is declared as detected only if it the cell under test exceeds the threshold, as well as, if the cell under test exceeds the two neighboring cells to its immediate left and right (the peak is a local maximum).

2.2.1.3 CFAR Engine – Cyclic vs. Non-Cyclic

The register CFAR_CYCLIC specifies whether the CFAR-CA detector needs to work in cyclic mode or in non-cyclic mode. These two modes are different in the way the samples at the edges are handled. In non-cyclic mode, the left side noise average is unavailable for the first several cells under test, therefore, only the right side noise average is used for those initial cells. Similarly, the right side noise average is unavailable for the last several cells under test, and only the left side noise average is used for those cells. For all other cells in the middle, both left and right side noise averaging is used as per the number of samples programmed in CFAR_AVG_LEFT and CFAR_AVG_RIGHT registers.

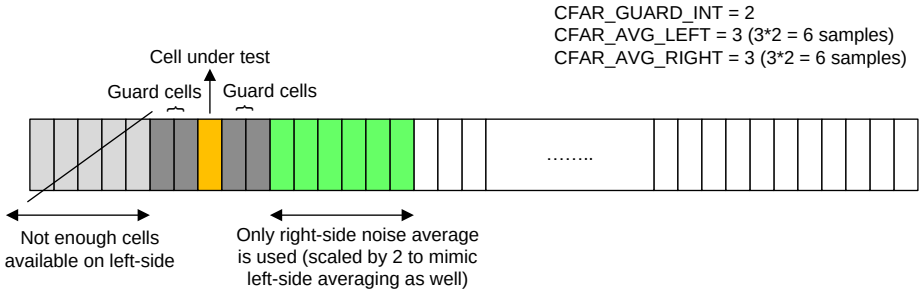


Figure 31. Handling of Samples Near the Edge in Non-Cyclic Mode

On the other hand, in cyclic mode, the CFAR-CA detector needs to wrap around the edges in a circular manner. In other words, for a cell under test that is near the left extreme, the left side noise average computation needs to use some samples from the right edge (circular wrap around the edge). Similarly, for a cell under test that is near the right extreme, the right side noise average computation needs to use some samples from the left edge (again, circular wrap around.

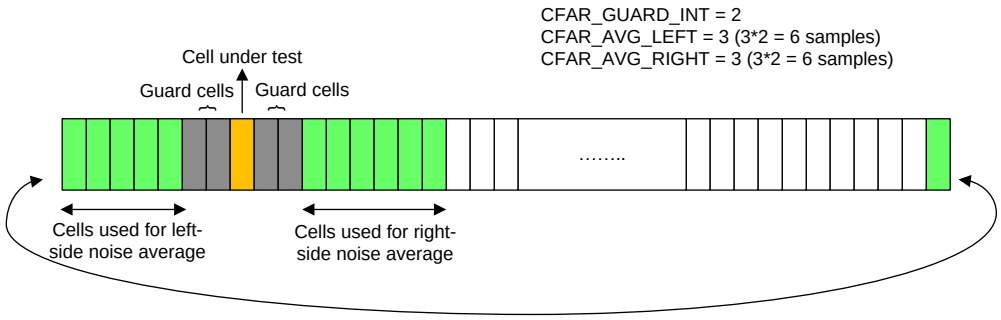


Figure 32. Handling of Samples Near the Edge in Cyclic Mode

This cyclic CFAR implementation is accomplished through a combination of a few register settings within the CFAR engine, as well as in the input and output formatter blocks. Specifically, the input formatter is configured to send additional samples (repeat samples) in a circular manner wrapping around the left and right edges. This is achieved by using the circular shift (CIRCIRSHIFT) and circular wrap-around (CIRCSHIFTWRAP) registers in the input formatter, such that the required number of extra samples at both edges are streamed into the CFAR engine. The cyclic CFAR mode only works when the number of cells under test is a power of 2.

For example, if the number of cells under test is 256, the average number of left and right noise samples is 32 each and the number of guard cells is 3 on either side. Then, the registers need to be programmed as shown in Table 15

Table 15. Configuration Example for CFAR Cyclic Mode

Module	Register Setting	Comments
CFAR Engine	CFAR_GUARD_INT = 3	3 guard cells on either side
	CFAR_AVG_LEFT = 16 CFAR_AVG_RIGHT = 16	32 samples on left side and 32 samples on right side for noise averaging
Input Formatter	SRCACNT = 325	255 + (32+3) + (32+3), where 255 is the usually configured value of SRCACNT for a 256 sample vector, plus 32+3 additional samples for circular repeat at either end
	CIRCIRSHIFT = 221	256 – (32+3), which is the starting offset for the circular shift, so that samples are streamed into CFAR engine start from this point
	CIRCSHIFTWRAP = 8	The circular wrap-around happens when SRCACNT counter value reaches 2^CIRCSHIFTWRAPP = 256
Output Formatter	REG_DST_SKIP_INIT = 0	No need to skip any samples at Output Formatter even though extra samples are fed into CFAR engine, because CFAR engine automatically strips out the extra samples
	DSTACNT = 255	256 outputs corresponding to 256 cells

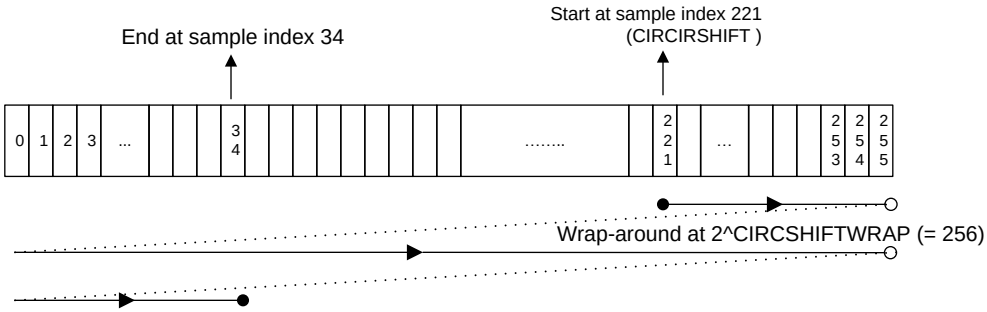


Figure 33. Input Formatter Sample Streaming for the Cyclic CFAR Example

2.2.1.4 CFAR Engine – Register Descriptions

Table 16 lists all the registers of the CFAR engine block. A few registers belonging to input formatter block that are related to cyclic CFAR mode of operation are also listed here.

Table 16. CFAR Engine Registers

Register	Width	Parameter-Set? (Y/N)	Description
CFAR_AVG_LEFT	6	Y	Number of samples for left-side noise averaging: This register is used to specify the number of samples used for noise averaging to the left of the cell under test. The number of samples used for noise averaging is equal to the value of this register multiplied by 2. For example, if this register value is 15, then the number of left-side samples used for averaging is 30. The maximum averaging that is possible is 126. A value of zero in this register means that the noise samples on the left side are not used for averaging. A value of 1 is not supported (valid values are 0, 2, 3, … 63).
CFAR_AVG_RIGHT	6	Y	Number of samples for right-side noise averaging: This register is very similar to the above, except that this register specifies the averaging to the right of the cell under test. In most cases, it is expected that CFAR_AVG_RIGHT has the same value as CFAR_AVG_LEFT.
CFAR_GUARD_INT	3	Y	Number of guard cells: This register specifies the number of guard cells to ignore on either side of the cell under test. If this register value is 3, then three guard cells on the left side and three guard cells on the right side are ignored. Only the noise samples beyond this guard region are used for calculating the surrounding noise average.

Table 16. CFAR Engine Registers (continued)

Register	Width	Parameter-Set? (Y/N)	Description
CFAR_THRESH	18	N	Threshold scale factor: This register is used to specify the threshold scale factor. This value is used to either multiply or add to the 'surrounding noise average' to determine the threshold used for detection of the present cell under test. If logarithmic CFAR mode is disabled (in magnitude or magnitude-squared mode), then the register value is multiplied with the surrounding noise average to determine the threshold, else it is added to the surrounding noise average. In the former case, this 18-bit register is interpreted as a 14.4 value. In the latter case (logarithmic mode), the 18-bit register is interpreted as a 7.11 value.
CFAR_LOG_MODE	1	Y	CFAR linear or logarithmic mode: This register is one of the registers used to specify whether the CFAR detector operates in linear or logarithmic mode. If this register bit is set, then the CFAR detector operates in logarithmic mode, which means that the threshold scale factor is added to (instead of multiplied with) the surrounding noise average value to determine the threshold. Note that this mode is meaningful only when the input samples to the CFAR detector are log-magnitude samples (see CFAR_INP_MODE as well). If this register bit is 0, then the logarithmic mode is disabled, in which case, the threshold scale factor is multiplied with (instead of added to) the surrounding noise average to determine the threshold. This mode is meaningful when magnitude or magnitude-squared samples are fed to the CFAR detector.
CFAR_INP_MODE	1	Y	CFAR engine input mode: This register bit specifies whether the inputs to the CFAR engine are complex samples or real values (the real values are already magnitude, magnitude-squared or log-magnitude numbers that can be directly sent to CFAR detection process). If this register bit is 1, then the input samples are real values and are directly sent to CFAR detection. If this register bit is 0, then the inputs are complex samples and hence either magnitude or magnitude-squared or log-magnitude computation is required prior to CFAR detection. Which of the three, viz., magnitude or magnitude-squared or log-magnitude is done, is selected by CFAR_ABS_MODE register described below.
CFAR_ABS_MODE	2	Y	CFAR magnitude, mag-squared or log-mag mode: This register is used to specify which of the three computations, namely Magnitude, Mag-squared or Log-Magnitude, is enabled inside the CFAR engine prior to CFAR detection. This register is only relevant when CFAR_INP_MODE is 0 (complex samples are fed to CFAR engine). 00b – Magnitude-squared 01b – Not valid 10b – Magnitude (using JPL approx.) 11b – Log2-Magnitude (using LUT approx.)
CFAR_OUT_MODE	2	Y	CFAR engine output mode: This register is used to select the output mode of the CFAR engine. The MSB bit of this register selects whether the CFAR Engine outputs all the noise average values for all the cells ('Raw output' mode), or whether the CFAR Engine outputs only the detected peaks ('Detected Peaks List' mode). The LSB bit specifies the content of the 24-bit 'I' and 'Q' channel outputs logged in destination memory. Refer main description section for details.
CFAR_GROUPING_EN	1	Y	CFAR peak grouping enable: This register bit specifies whether peak grouping should be enabled. When this register bit is 0, peak grouping is disabled, which means that a peak is declared as detected as long as the cell under test exceeds the threshold. On the other hand, if this register bit is 1, then a peak is declared as detected only if it the cell under test exceeds the threshold, as well as, if the cell under test exceeds the two neighboring cells to its immediate left and right (local maximum).
CFAR_NOISE_DIV	4	Y	CFAR noise average division factor: This register specifies the division factor with which the noise sum calculated from the left and right noise windows are divided, in order to get the final surrounding noise average value. The division factor is equal to 2^ CFAR_NOISE_DIV. Therefore, only powers-of-2 division are possible, even though the number of samples specified in CFAR_AVG_LEFT and CFAR_AVG_RIGHT are not restricted to powers of 2. The surrounding noise average value obtained after the division is multiplied or added with CFAR_THRESH to determine the final threshold used to compare the cell under test for detection. The maximum allowed value for this register is 8, which gives a division factor of 256.
CFAR_CA_MODE	2	Y	CFAR noise averaging mode:

Table 16. CFAR Engine Registers (continued)

Register	Width	Parameter-Set? (Y/N)	Description
CFAR_CYCLIC	1	Y	This register configures the noise averaging mode in the CFAR detector from one of three options – CFAR-CA, CFAR-CAGO, CFAR-CASO. 00b – CFAR-CA 01b – CFAR-CAGO 10b – CFAR-CASO 11b – Not valid CFAR cyclic vs. non-cyclic mode: This register bit specifies whether the CFAR-CA detector needs to work in cyclic mode or in non-cyclic mode. When this register bit is 0, the CFAR detector works in non-cyclic mode and when it is 1, it works in cyclic mode. Refer main description section for details on how to configure and use cyclic mode.
FFTPEAKCNT (read-only)	12	N	CFAR detected peak count: This is a read-only register that contains the number of detected peaks that are logged in the destination memory, when CFAR Engine is configured in 'Detected Peaks List' mode. In the Detected Peaks List mode, since only the detected peaks are logged in the destination memory, this read-only register provides the number of detected peaks that are logged to the main processor, so that the main processor can determine how many entries to read from the destination memory.
CIRCIRSHIFT	12	Y	This register is part of input formatter block. Source Circular Shift: This register specifies the circular shift (offset in samples) that should be applied on the sequence of input samples before feeding them to the core computational unit. This register, together with CIRCSHIFTWRAP register, is useful when CFAR detection needs to be done in cyclic mode. Refer main description section for details.
CIRCSHIFTWRAP	4	Y	This register is part of Input Formatter block Source Circular Shift Wraparound: This register indicates at what number (power-of-2) the sample counter value should wraparound, when circular shift is needed. The counter wraps around at 2^ CIRCSHIFTWRAP. The sample counter starts counting from the programmed circular shift value (CIRCIRSHIFT) and when the counter crosses (2^CIRCSHIFTWRAP-1), it wraps back to zero to continue the count.

2.3 Core Computational Unit – Statistics

This section describes the statistics block present in the core computational unit.

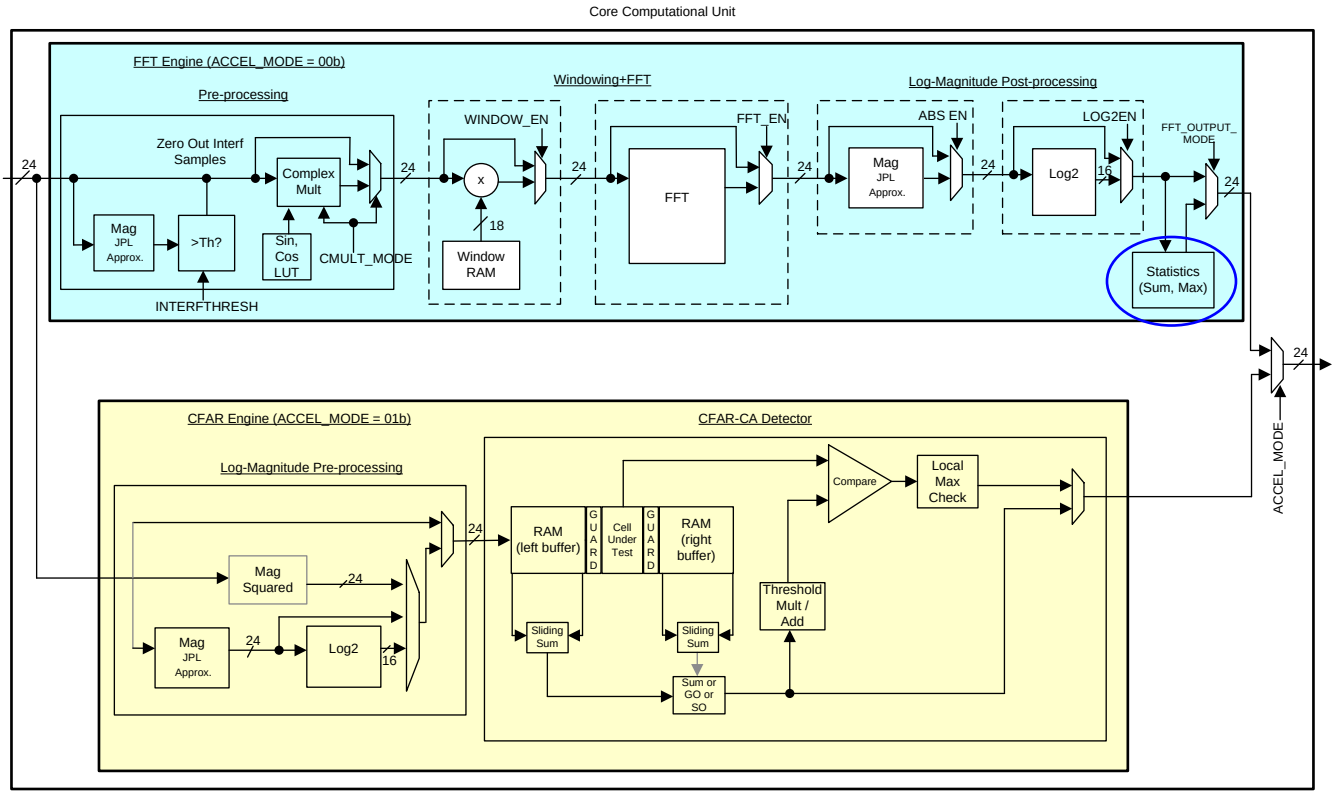


Figure 34. Statistics Block

2.3.1 Statistics Block

The core computational unit has a statistics computation block at the end of the FFT Engine path as shown in Figure 34. This block can be used to compute a few statistics, such as sum and max of the samples output by the core computational unit.

2.3.1.1 Statistics Block – Operation

The 24-bit I and 24-bit Q output of the core computational unit goes to a statistics computation block. The purpose of this block is to find the maximum and sum (average) of the output samples

The sum and max statistics are computed on a 'per-iteration' basis (the sum and max values are logged at the end of each iteration) and the computation is reset for the next iteration. The sum and max values are logged in register-sets (see MAXn_VALUE, MAXn_INDEX, ISUMn, QSUMn register-sets), which can be read by the main processor. However, only four such registers are provided for each statistic and therefore, the sum and max values can be logged in these registers only for up to a maximum of four iterations.

The max statistics register-set comprises four read-only registers of 24 bits each, named MAXn_VALUE, for recording max values, and four read-only registers each 12 bits unsigned, named MAXn_INDEX, for recording the max indices. The sum statistics register-set contains four registers of 36 bits each, named ISUMn, for I-sum statistics, and 4 registers of 36 bits each, named QSUMn, for Q-sum statistics.

For larger number (>4) of iterations, either the sum or the max value can be sent to the destination memory for each iteration, which allows the statistic to be available even for cases with more than four iterations. The logging of the statistic into the destination memory is enabled using FFT_OUTPUT_MODE register described below.

The MSB bit of the FFT_OUTPUT_MODE register selects whether the default (main) output of the core computational unit goes to the destination memory, or the statistics block output. If the MSB of this 2-bit register is 0, then it selects the default mode of operation, where the main output (FFT or Log-Mag result) is sent to the destination memory. If the MSB is 1, then it selects the statistics output mode, where either the sum or max statistic is sent to the destination memory (one value per iteration). Whether the sum or max is sent to memory is dependent on the LSB bit. If the LSB bit is 0, then the statistic value that is sent is the max value (useful in conjunction with Log-Mag enabled to find the biggest peak and peak index per iteration). Here, the I output is the maximum value itself and the Q output is the index (location) of the maximum value. If the LSB bit is 1, then the statistic value that is sent is the sum value (useful for DFT mode, as well as for mean squared or mean of absolute values computation).

Table 17. Statistics Output Modes

FFT_OUTPUT_MODE Register	I Channel Output	Q Channel Output
00b – Default output mode	Main output of core computational unit	
10b – Max statistics output (One output per iteration)	Max Value	Max Index
11b – Sum statistics output (One output per iteration)	Sum of I values	Sum of Q values

The max statistic records the maximum value (and its index) of the magnitude or log-magnitude samples corresponding to every iteration. The sum statistic records the sum of the magnitude or log-magnitude or the complex output samples corresponding to each iteration. If the main output of the core computational unit is the complex FFT output (ABS EN=0 and LOG2EN=0), then the sum statistics is the complex sum.

The complex sum statistics mode is useful when used in conjunction with the complex multiplier block in DFT mode or vector multiplication mode. For example, the sum statistic computed here, together with the DFT mode of the complex multiplier block, enable DFT computation for the desired number of bins (iterations). When the desired number of bins is more than 4, the sum statistic can be sent to destination memory (instead of the main data output that is normally sent to the destination memory).

Note that when the sum statistics is logged into the destination memory, it goes through the Output Formatter block as only 24-bits each for I and Q (same bit-width as the primary FFT outputs). Hence, the computed sum statistics value of 36-bits width, needs to be scaled down by right-shifting the appropriate number of LSBs (using FFTSUMDIV register) before sending to output formatter. Thus, when logging the statistics in destination memory, the sum statistics is to be used as an “average” value, rather than a “sum” value itself.

The FFTSUMDIV register specifies the number of bits to right-shift the sum statistic before it is written to destination memory. The internal sum statistic register is 36-bits wide (allowing 12 bits of MSB growth of the 24-bit data path), but this statistics value needs to be scaled down to 24 bits to match the data path width going to the Output Formatter. This register specifies how many LSBs to drop to convert the sum statistics to 24-bit value. Note that only signed saturation is implemented (irrespective of whether magnitude values are being summed or complex FFT output values are being summed). Therefore, it is recommended that this register is configured to drop an appropriate number of LSBs such that incorrect saturation in case of magnitude sum is avoided.

Note that in statistics output mode, the registers DSTACNT, DSTAINDX, DSTBINDX, DST16b32b and DSTREAL are not meant to be used, since it is known that there is only one value to be written to destination memory for every iteration in a specific format. It is recommended that in this mode, DSTACNT be programmed to its maximum value of 4095, DSTAINDX and DSTBINDX are both programmed to a value of 8 bytes, DST16b32b is set to 1 and DSTREAL is reset to 0. The statistics is then always logged in the destination memory as consecutive 32-bit I and Q samples, irrespective of whether sum statistic or max statistic is being logged.

2.3.1.2 Statistics block – Register Descriptions

Table 18 lists all the registers of the statistics block.

Table 18. Statistics Block Registers

Register	Width	Parameter-Set? (Y/N)	Description
MAX1VALUE MAX2VALUE MAX3VALUE MAX4VALUE	24	N	Max value: These registers contain the max value on a per-iteration basis. These registers are meaningful only when Magnitude or Log-Magnitude is enabled. Only the max values for up to four iterations are recorded in these registers. For larger number of iterations, use statistics output mode (FFT_OUTPUT_MODE below).
MAX1INDEX MAX2INDEX MAX3INDEX MAX4INDEX	12	N	Max index: These registers contain the max index on a per-iteration basis, corresponding to each max value in the MAXn_VALUE registers.
ISUM1LSB and ISUM1MSB ISUM2LSB, ISUM2MSB ISUM3LSB, ISUM3MSB ISUM4LSB, ISUM4MSB			Sum statistics: These registers contain the sum of the I outputs and Q outputs on a per-iteration basis. Only the statistics for up to four iterations are recorded in these registers. For larger number of iterations, use statistics output mode (FFT_OUTPUT_MODE below).
FFT_OUTPUT_MODE	2	Y	FFT Path output mode: This register specifies the output mode of the FFT path. Instead of the default mode where the main output of the core computational unit is sent to the destination memory, this register can be configured such that either the max or sum statistics can be sent to the destination memory. 00b – Default mode (main output) 10b – Max statistics output mode 11b – Sum statistics output mode
FFTSUMDIV	5	N	Right-shifting for Sum statistic: This register specifies the number of bits to right-shift the sum statistic before it is written to destination memory. The internal sum statistic register is 36-bits wide (allowing 12 bits of MSB growth of the 24-bit data path), but this statistics value needs to be scaled down to 24 bits to match the data path width going to the Output Formatter. This register specifies how many LSBs to drop to convert the sum statistics to 24-bit value.

2.4 FFT Stitching Use-Case Example

This section presents examples that illustrate how to configure and use the radar hardware accelerator for the special use-case of FFT stitching.

2.4.1 FFT Stitching Use-Case

As described earlier, the radar hardware accelerator natively supports FFT sizes of up to 1024. However, FFT of size 2048 and 4096 can also be accomplished using a two-step FFT stitching process. This involves the use of two parameter-sets as shown in the example below.

Consider an industrial level-sensing use-case with 1 TX, 1 RX and 4096 complex samples per chirp. During active chirp transmission, the Digital Front-end (DFE) writes ADC samples to the ADC buffer in ping-pong manner. This example assumes that the ADC data is complex (the RF/analog is configured as complex baseband (instead of real-only) chain). Since 4096 samples requires 4096*4 = 16384 bytes, the DFE fills up the entire ping or pong ADC buffers (ACCEL_MEM0 or ACCEL_MEM1) for every chirp. The DFE configuration details are outside the scope of this user's guide.

The FFT processing using the radar hardware accelerator can be done inline (as and when the ADC data is available) by setting FFT1DEN = 1, such that the ADC buffer is shared with the accelerator input memories and the DFE output is directly available to the accelerator for processing at the end of every chirp (ping-pong switch). Alternately, it is possible to not use the radar hardware accelerator during ADC data collection, and instead, simply transferring the ADC data into L3 memory using DMA at the end of every ping-pong switch. In such a case, after all the chirps are collected, the radar hardware accelerator can be used to perform FFT during inter-frame time. This is accomplished by setting FFT1DEN register bit to 0, such that the ADC buffer is not shared with the accelerator input memories and therefore, the accelerator input memories are directly accessible for DMA transfer to provide input data for the accelerator. The use of inline FFT processing and inter-frame FFT processing is covered in the *Radar Hardware Accelerator - Use Case Example* Section 1.6.

In Table 19 and Table 20, the parameter-set configurations for performing one 4096-point FFT using FFT stitching is shown. The input data is assumed to be in ACCEL_MEM0 as shown in the layout below. The FFT stitching is achieved in two steps as follows.

1. The first step involves computing four 1024-point FFTs of input samples. For these four 1024-FFT computation, input samples are sent in following order:
   a. Iteration #0: x[0], x[4], x[8], x[12], …, x[4092]
   b. Iteration #1: x[1], x[5], x[9], x[13], …, x[4093]
   c. Iteration #2: x[2], x[6], x[10], x[14], …, x[4094]
   d. Iteration #3: x[3], x[7], x[11], x[15], …, x[4095]

   Where x[0], x[1], x[2], …., x[4095] are the original 4096-point input samples which are stored in ACCEL_MEM0 in consecutive locations.

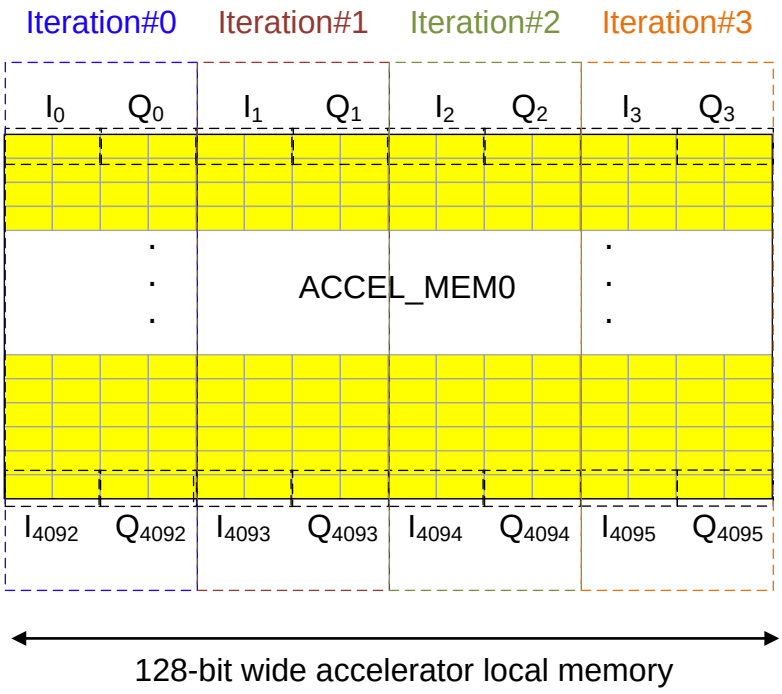


Figure 35. Layout of Samples in Source Memory for 1TX, 1RX, 4K Complex FFT

Further, before computing the 1024-point FFT in each iteration, apply windowing to the incoming input samples. Note that the window RAM can hold a maximum of only 1024 window coefficients. When larger FFT (2K and 4K) is needed via stitching of multiple smaller-size FFTs, a down-sampled set of window coefficients is stored in window RAM (1024 coefficients of original 4096 point window are stored by picking every 4th coefficient) and then, the accelerator is configured to use a linearly interpolation between these 1024 window samples to get intermediate window coefficients. The WINDOW_INTERP_FRACTION register, which is part of the parameter set, is used to configure interpolation mode for the window coefficients. When this register is 01b, then the window coefficients are applied as is from the window RAM for the first iteration, and then linearly interpolated between successive coefficients with an interpolation fraction of 0.25, 0.5, 0.75 for the second, third and fourth iterations respectively. This corresponds to the linear interpolation of window coefficients as needed for a 4K size FFT. (When performing 2K size FFT, the WINDOW_INTERP_FRACTION register should be programmed to 10b, in which case, the first iteration uses the window RAM coefficients as is, and the second iteration linearly interpolates between consecutive coefficients with interpolation fraction of 0.5). Note that when linear interpolation for the window coefficients is used, the symmetric window mode (WINSYMM = 1) cannot be used.

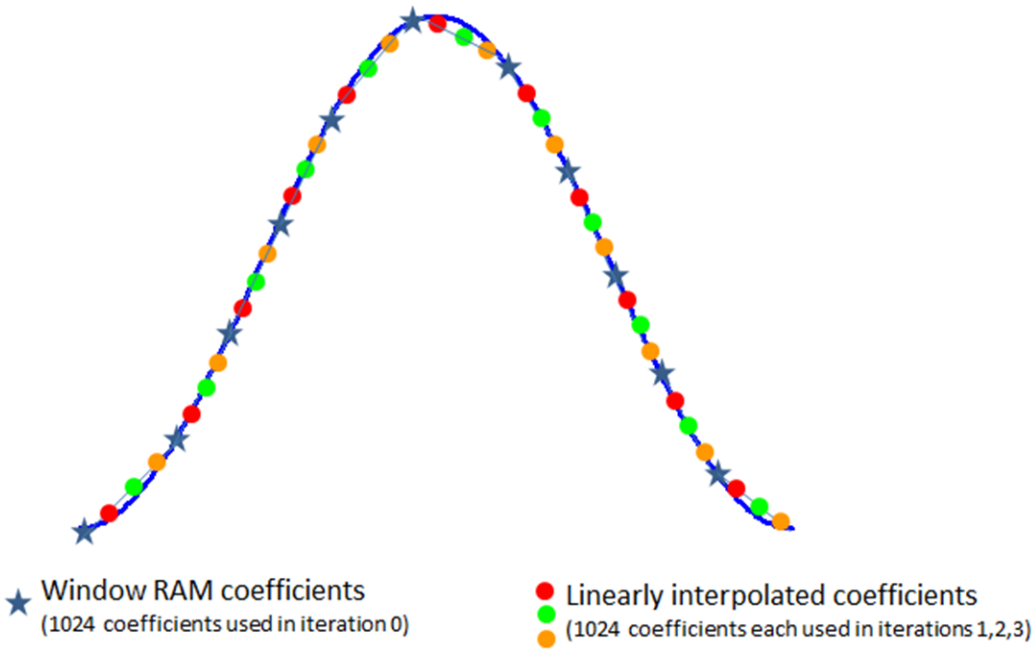


Figure 36. Linear Interpolation of Window RAM Coefficients for 4K FFT

Windowing output is fed to FFT engine for 1024-point FFT computation (see Step 1). At the end of this step, the complex FFT output is stored in ACCEL_MEM2 in the same fashion as they are picked from source memory as illustrated in Figure 37. Note that $I_{i,j}$ represent real part of i^{th} bin FFT output for j^{th} iteration. Similarly, $Q_{i,j}$ represent imaginary part of i^{th} bin FFT output for j^{th} iteration.

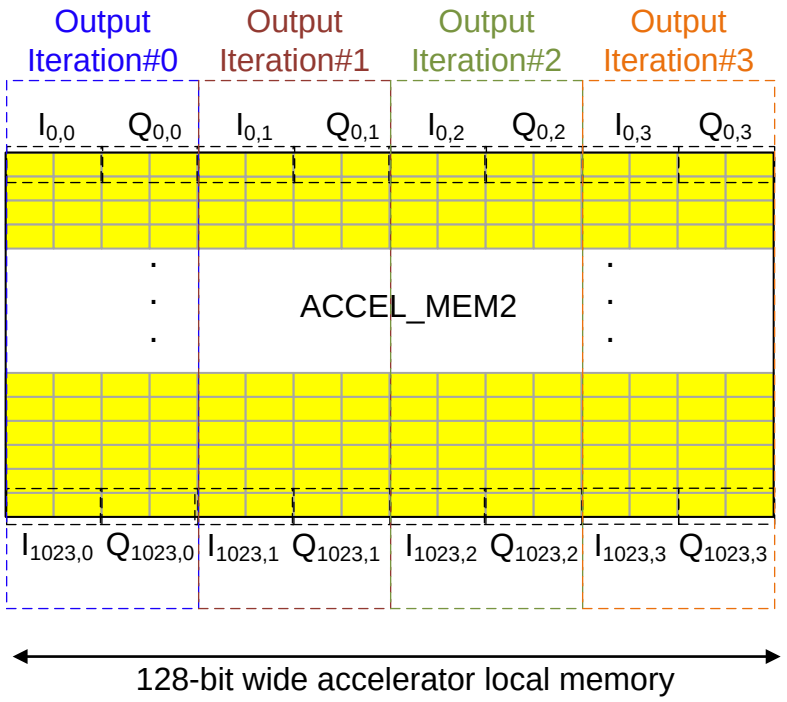


Figure 37. Layout of Samples in Destination Memory (after Step 1)

2. In the second step, 1024 4-point FFTs in stitching mode are computed and the results written back to ACCEL_MEM0, thus, over-writing the original input data. For 4-point FFT computation, the input samples are sent through the FFT engine in a transpose manner as illustrated in Figure 38. In this case, the complex multiplier in the pre-processing block needs to be configured to enable 4K FFT stitching. After each 4-point FFT, the output samples correspond to FFT bins spaced apart by 1024. For example, the first iteration produces outputs corresponding to bins 0, 1024, 2048 and 3072. Similarly, the second iteration produces outputs corresponding to bins 1, 1025, 2049 and 3073. By using appropriate DSTAINDX and DSTBINDX settings, the output samples can be arranged in the correct bin order as desired.

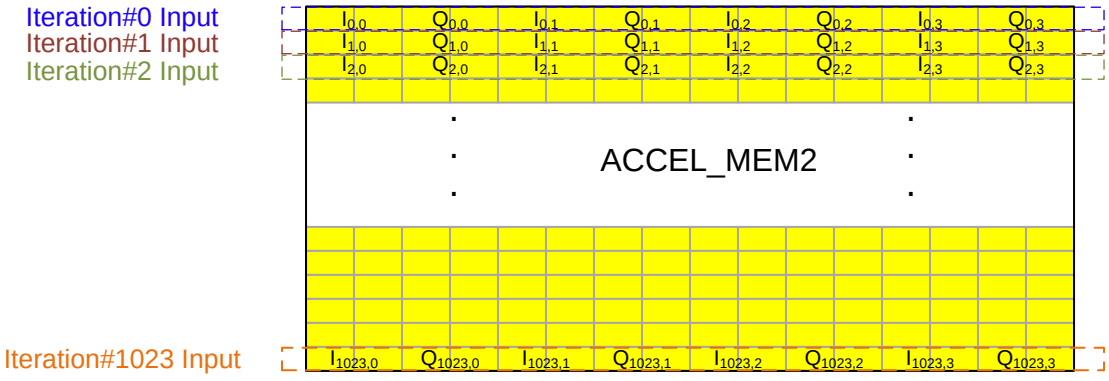


Figure 38. 1024 4-Point FFTs in Step 2 (FFT Stitching)

The key register configurations to perform 4K size complex FFT using FFT stitching are tabulated in Table 19.

Table 19. Parameter-Set #0: Used for Step #1

Register	Value	Comments
FFT_EN	1	Enable FFT computation
FFTSIZE	10	FFT size = 2^10 = 1024
WINDOW_EN	1	Enable windowing
WINDOW_INTERP_ FRACTION	1	Enable windowing interpolation for 4096-point FFT stitching
SRCACNT	1023	1024 valid samples
SRCAINDX	16	Adjacent samples spaced 16 bytes apart
REG_BCNT	3	Four back-to-back 1024-point FFT
SRCBINDX	4	Samples are 4 bytes apart for successive iterations
SRCADDR	0	Start at beginning of ACCEL_MEM0
DSTADDR	32KB	Destination memory is ACCEL_MEM2
DSTAINDX	16	
DSTBINDX	4	
DSTACNT	1023	
SRC16b32b	0	FFT input samples are 16-bit word aligned
DST16b32b	0	FFT output samples are 16-bit word aligned

Table 20. Parameter-Set #1: Used for Step #2

Register	Value	Comments
FFT_EN	1	Enable FFT computation
FFTSIZE	2	FFT size = 2^2 = 4
CMULT_MODE	3	Enables FFT Stitching Mode of complex multiplier
TWIDINCR	1	4096-Point FFT Stitching
SRCACNT	3	Four valid samples (zero-based count)
SRCAINDX	4	Adjacent samples spaced 4 bytes apart
REG_BCNT	1023	Need to compute 4-point FFT 1024 times
SRCBINDX	16	Each set for 4–point FFT are spaced 16 bytes apart
SRCADDR	32KB	Input samples for this step are stored in ACCEL_MEM2
DSTADDR	0KB	Output of this step stored back in ACCEL_MEM0
DSTAINDX	1024*4	Adjacent output samples of each iteration are 4KB apart, so that at the end, 4096-point FFT is output in linear bin order
DSTBINDX	4	First output sample of each iteration is 4 bytes apart to ensure final 4096-point FFT output is in linear bin order
DSTACNT	3	4-point FFT output
SRC16b32b	0	FFT input samples are 16-bit word aligned
DST16b32b	0	FFT output samples are 16-bit word aligned

At the end of two parameter sets, the result of the 4096-point FFT is available in ACCEL_MEM0 in correct bin order and can be transferred back to L3 memory via DMA. Alternately, log-magnitude and/or CFAR detection processing can be done in the radar hardware accelerator itself.

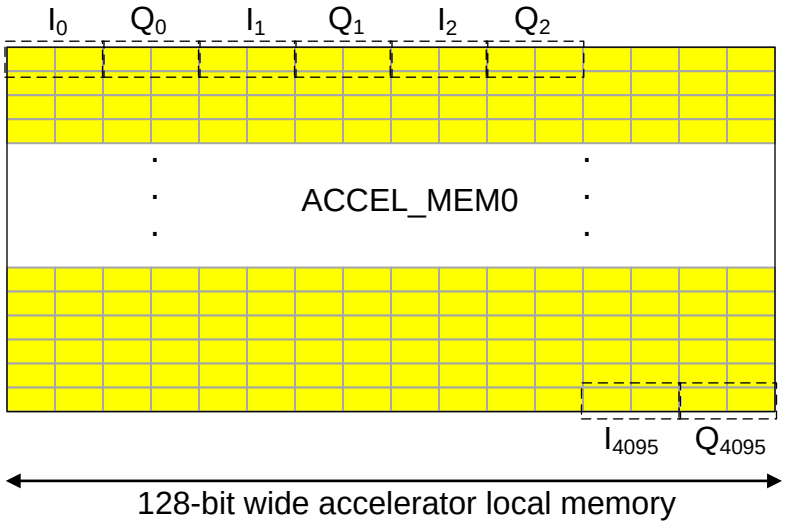


Figure 39. Layout of Final FFT Output in Correct Bin Order

3 References

- *Industrial Radar Family Technical Reference Manual*
- AWR16xx/14xx Technical Reference Manual
- *mmWave Software Development Kit*

Hardware Accelerator Errata

A.1 Advisory to HWA Variant/Revision Map

Table 21. Advisory to HWA Variant / Revision Map

Advisory Number	Advisory Title	HWA Version	
		V1.0	V1.05
MSS#03	Incorrect Handling of “Saturation” in FFT Hardware Accelerator’s Input / Output Formatter and Statistics Block	√	√
MSS#04	Number of Samples (SRCACNT) Should be >3 for Correct Operation of FFT Hardware Accelerator	√	
MSS#05	Incorrect FFT Intermediate Stage Clip Status Indication	√	
MSS#13	Incorrect Read from FFT Hardware Accelerator After Complex Multiplication Operation	√	√
MSS#21	Issue with Input formatter 16 bit real signed format	√	√

MSS#03 *Incorrect Handling of “Saturation” in FFT Hardware Accelerator’s Input/Output Formatter and Statistics Block*

Revision(s) Affected: HWA v1.0.

For HWA v1.05, only staticstics block "saturation" issue is applicable. See below descritption for details.

Description:

- Input formatter block performs saturation based on signed or unsigned samples. However, the compute engine module always saturates the input as a 24 bit signed number. (1)
- Output formatter block saturates the 16 bits unsigned number as signed number. This causes magnitude outputs to have a smaller max range. (2)
- Statistics block always assumes that input is signed when checking for saturation, but the input can be unsigned in some cases. (3)

Workaround(s): None. Silicon update will be provided by TI.

MSS#04 *Number of Samples (SRCACNT) Should be >3 for Correct Operation of FFT Hardware Accelerator*

Revision(s) Affected: HWA v1.0

Description: Logic which subtracts the compute engine pipelined delay from FFT counter wraps around incorrectly when the number of samples is less (specifically, when it is 3).

Workaround(s): None. Silicon update will be provided by TI.

(1) Only applicable to HWA v1.0
(2) Only applicable to HWA v1.0
(3) Applicable to both HWA v1.0 and HWA v1.05

MSS#05 *Incorrect FFT Intermediate Stage Clip Status Indication*

Revision(s) Affected: HWA v1.0

Description: FFT clip status register incorrectly uses all butterfly stages, even if only a few of the stages are enabled.

Workaround(s): None. Silicon update will be provided by TI.

MSS#13 *Incorrect Read from FFT Hardware Accelerator After Complex Multiplication Operation*

Revision(s) Affected: HWA v1.0 and HWA v1.05

Description: Read-back from FFT hardware accelerator slave, static configuration registers, Window RAM, Param RAM, and First stage RAM gives incorrect data if the last operation performed by the accelerator was a complex multiplication.

Workaround(s) None. Silicon update will be provided by TI.

MSS#21 *Issue with Input formatter 16-bit real signed format Operation*

Revision(s) Affected HWA v1.0 and HWA v1.05

Description: Wrong sign extension is implemented for 16 bit signed format in real only mode operation. Hence, signed 16-bit real format cannot be supported for input formatter.

Workaround(s) None. Silicon update will be provided by TI.

B.1 Register Map

The registers used to control and configure the radar accelerator fall into two categories: parameter-set registers programmed in the parameter-set configuration memory for each parameter-set that is used (up to 16 parameter-sets are possible, and they can be chained and looped), and common registers that are not part of the parameter-set and thus are common for all the 16 parameter sets. The registers listed in this section is for both HWA v1.0 and v1.05.

B.1.1 Parameter-Set Register Map

Figure 40 shows the layout of the parameter-set registers. The layout of the registers is shown separately for the FFT Engine and CFAR Engine paths. For each parameter-set, there are eight 32-bit registers to be configured in the parameter-set configuration memory.

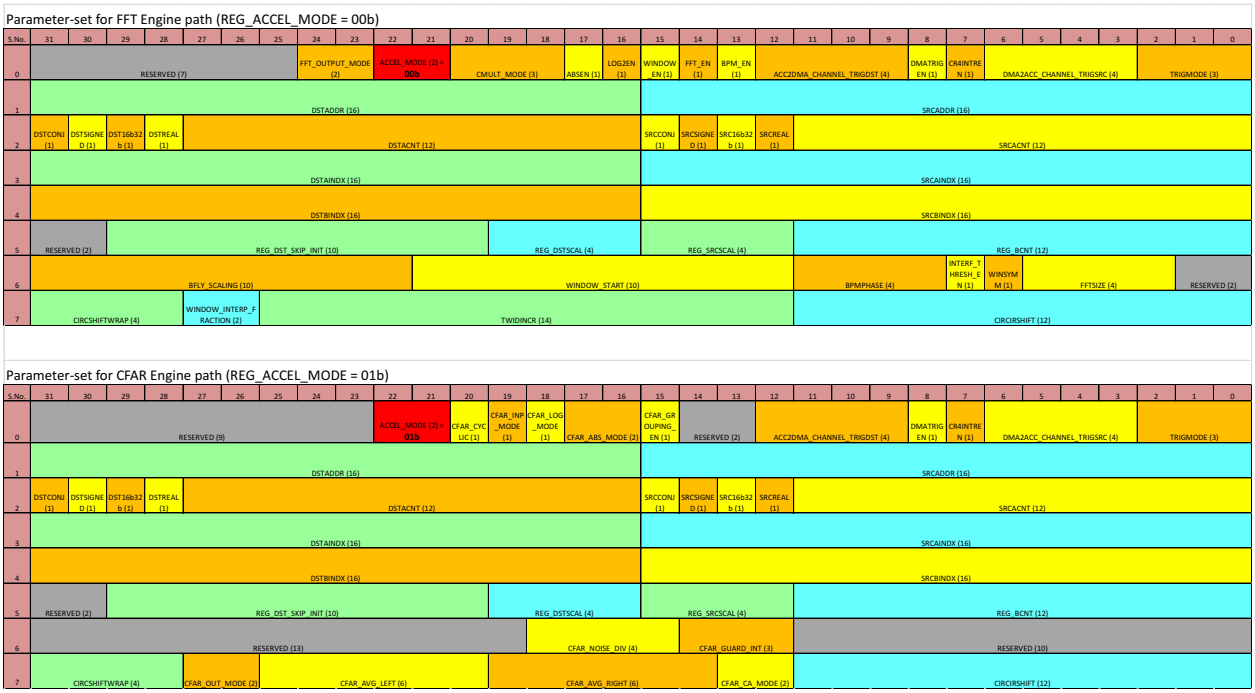


Figure 40. Register Layout of Parameter-Set Registers

B.1.2 DSS_HW_ACC_PARAM Registers

Table 22 lists the memory-mapped registers for the DSS_HW_ACC_PARAM. All register offset addresses not listed in Table 22 should be considered as reserved locations. TI recommends clearing reserved registers to "0" before programming. The description of the parameter registers are for fft path only. For the CFAR path register description, please refer to Section 2.2 for more details.

Table 22. DSS_HW_ACC_PARAM Registers

Offset	Acronym	Register Name	Section
0h	PARAM1_0	Parameter-set register	Section B.1.2.1
4h	PARAM1_1	Parameter-set register	Section B.1.2.2
8h	PARAM1_2	Parameter-set register	Section B.1.2.3
Ch	PARAM1_3	Parameter-set register	Section B.1.2.4
10h	PARAM1_4	Parameter-set register	Section B.1.2.5
14h	PARAM1_5	Parameter-set register	Section B.1.2.6
18h	PARAM1_6	Parameter-set register	Section B.1.2.7
1Ch	PARAM1_7	Parameter-set register	Section B.1.2.8
20h	PARAM2_0	Parameter-set register	Section B.1.2.9
24h	PARAM2_1	Parameter-set register	Section B.1.2.10
28h	PARAM2_2	Parameter-set register	Section B.1.2.11
2Ch	PARAM2_3	Parameter-set register	Section B.1.2.12
30h	PARAM2_4	Parameter-set register	Section B.1.2.13
34h	PARAM2_5	Parameter-set register	Section B.1.2.14
38h	PARAM2_6	Parameter-set register	Section B.1.2.15
3Ch	PARAM2_7	Parameter-set register	Section B.1.2.16
40h	PARAM3_0	Parameter-set register	Section B.1.2.17
44h	PARAM3_1	Parameter-set register	Section B.1.2.18
48h	PARAM3_2	Parameter-set register	Section B.1.2.19
4Ch	PARAM3_3	Parameter-set register	Section B.1.2.20
50h	PARAM3_4	Parameter-set register	Section B.1.2.21
54h	PARAM3_5	Parameter-set register	Section B.1.2.22
58h	PARAM3_6	Parameter-set register	Section B.1.2.23
5Ch	PARAM3_7	Parameter-set register	Section B.1.2.24
60h	PARAM4_0	Parameter-set register	Section B.1.2.25
64h	PARAM4_1	Parameter-set register	Section B.1.2.26
68h	PARAM4_2	Parameter-set register	Section B.1.2.27
6Ch	PARAM4_3	Parameter-set register	Section B.1.2.28
70h	PARAM4_4	Parameter-set register	Section B.1.2.29
74h	PARAM4_5	Parameter-set register	Section B.1.2.30
78h	PARAM4_6	Parameter-set register	Section B.1.2.31
7Ch	PARAM4_7	Parameter-set register	Section B.1.2.32
80h	PARAM5_0	Parameter-set register	Section B.1.2.33
84h	PARAM5_1	Parameter-set register	Section B.1.2.34
88h	PARAM5_2	Parameter-set register	Section B.1.2.35
8Ch	PARAM5_3	Parameter-set register	Section B.1.2.36
90h	PARAM5_4	Parameter-set register	Section B.1.2.37
94h	PARAM5_5	Parameter-set register	Section B.1.2.38
98h	PARAM5_6	Parameter-set register	Section B.1.2.39
9Ch	PARAM5_7	Parameter-set register	Section B.1.2.40
A0h	PARAM6_0	Parameter-set register	Section B.1.2.41
A4h	PARAM6_1	Parameter-set register	Section B.1.2.42

Table 22. DSS_HW_ACC_PARAM Registers (continued)

Offset	Acronym	Register Name	Section
A8h	PARAM6_2	Parameter-set register	Section B.1.2.43
ACh	PARAM6_3	Parameter-set register	Section B.1.2.44
B0h	PARAM6_4	Parameter-set register	Section B.1.2.45
B4h	PARAM6_5	Parameter-set register	Section B.1.2.46
B8h	PARAM6_6	Parameter-set register	Section B.1.2.47
BCh	PARAM6_7	Parameter-set register	Section B.1.2.48
C0h	PARAM7_0	Parameter-set register	Section B.1.2.49
C4h	PARAM7_1	Parameter-set register	Section B.1.2.50
C8h	PARAM7_2	Parameter-set register	Section B.1.2.51
CCh	PARAM7_3	Parameter-set register	Section B.1.2.52
D0h	PARAM7_4	Parameter-set register	Section B.1.2.53
D4h	PARAM7_5	Parameter-set register	Section B.1.2.54
D8h	PARAM7_6	Parameter-set register	Section B.1.2.55
DCh	PARAM7_7	Parameter-set register	Section B.1.2.56
E0h	PARAM8_0	Parameter-set register	Section B.1.2.57
E4h	PARAM8_1	Parameter-set register	Section B.1.2.58
E8h	PARAM8_2	Parameter-set register	Section B.1.2.59
ECh	PARAM8_3	Parameter-set register	Section B.1.2.60
F0h	PARAM8_4	Parameter-set register	Section B.1.2.61
F4h	PARAM8_5	Parameter-set register	Section B.1.2.62
F8h	PARAM8_6	Parameter-set register	Section B.1.2.63
FCh	PARAM8_7	Parameter-set register	Section B.1.2.64
100h	PARAM9_0	Parameter-set register	Section B.1.2.65
104h	PARAM9_1	Parameter-set register	Section B.1.2.66
108h	PARAM9_2	Parameter-set register	Section B.1.2.67
10Ch	PARAM9_3	Parameter-set register	Section B.1.2.68
110h	PARAM9_4	Parameter-set register	Section B.1.2.69
114h	PARAM9_5	Parameter-set register	Section B.1.2.70
118h	PARAM9_6	Parameter-set register	Section B.1.2.71
11Ch	PARAM9_7	Parameter-set register	Section B.1.2.72
120h	PARAM10_0	Parameter-set register	Section B.1.2.73
124h	PARAM10_1	Parameter-set register	Section B.1.2.74
128h	PARAM10_2	Parameter-set register	Section B.1.2.75
12Ch	PARAM10_3	Parameter-set register	Section B.1.2.76
130h	PARAM10_4	Parameter-set register	Section B.1.2.77
134h	PARAM10_5	Parameter-set register	Section B.1.2.78
138h	PARAM10_6	Parameter-set register	Section B.1.2.79
13Ch	PARAM10_7	Parameter-set register	Section B.1.2.80
140h	PARAM11_0	Parameter-set register	Section B.1.2.81
144h	PARAM11_1	Parameter-set register	Section B.1.2.82
148h	PARAM11_2	Parameter-set register	Section B.1.2.83
14Ch	PARAM11_3	Parameter-set register	Section B.1.2.84
150h	PARAM11_4	Parameter-set register	Section B.1.2.85
154h	PARAM11_5	Parameter-set register	Section B.1.2.86
158h	PARAM11_6	Parameter-set register	Section B.1.2.87
15Ch	PARAM11_7	Parameter-set register	Section B.1.2.88
160h	PARAM12_0	Parameter-set register	Section B.1.2.89

Table 22. DSS_HW_ACC_PARAM Registers (continued)

Offset	Acronym	Register Name	Section
164h	PARAM12_1	Parameter-set register	Section B.1.2.90
168h	PARAM12_2	Parameter-set register	Section B.1.2.91
16Ch	PARAM12_3	Parameter-set register	Section B.1.2.92
170h	PARAM12_4	Parameter-set register	Section B.1.2.93
174h	PARAM12_5	Parameter-set register	Section B.1.2.94
178h	PARAM12_6	Parameter-set register	Section B.1.2.95
17Ch	PARAM12_7	Parameter-set register	Section B.1.2.96
180h	PARAM13_0	Parameter-set register	Section B.1.2.97
184h	PARAM13_1	Parameter-set register	Section B.1.2.98
188h	PARAM13_2	Parameter-set register	Section B.1.2.99
18Ch	PARAM13_3	Parameter-set register	Section B.1.2.100
190h	PARAM13_4	Parameter-set register	Section B.1.2.101
194h	PARAM13_5	Parameter-set register	Section B.1.2.102
198h	PARAM13_6	Parameter-set register	Section B.1.2.103
19Ch	PARAM13_7	Parameter-set register	Section B.1.2.104
1A0h	PARAM14_0	Parameter-set register	Section B.1.2.105
1A4h	PARAM14_1	Parameter-set register	Section B.1.2.106
1A8h	PARAM14_2	Parameter-set register	Section B.1.2.107
1ACh	PARAM14_3	Parameter-set register	Section B.1.2.108
1B0h	PARAM14_4	Parameter-set register	Section B.1.2.109
1B4h	PARAM14_5	Parameter-set register	Section B.1.2.110
1B8h	PARAM14_6	Parameter-set register	Section B.1.2.111
1BCh	PARAM14_7	Parameter-set register	Section B.1.2.112
1C0h	PARAM15_0	Parameter-set register	Section B.1.2.113
1C4h	PARAM15_1	Parameter-set register	Section B.1.2.114
1C8h	PARAM15_2	Parameter-set register	Section B.1.2.115
1CCh	PARAM15_3	Parameter-set register	Section B.1.2.116
1D0h	PARAM15_4	Parameter-set register	Section B.1.2.117
1D4h	PARAM15_5	Parameter-set register	Section B.1.2.118
1D8h	PARAM15_6	Parameter-set register	Section B.1.2.119
1DCh	PARAM15_7	Parameter-set register	Section B.1.2.120
1E0h	PARAM16_0	Parameter-set register	Section B.1.2.121
1E4h	PARAM16_1	Parameter-set register	Section B.1.2.122
1E8h	PARAM16_2	Parameter-set register	Section B.1.2.123
1ECh	PARAM16_3	Parameter-set register	Section B.1.2.124
1F0h	PARAM16_4	Parameter-set register	Section B.1.2.125
1F4h	PARAM16_5	Parameter-set register	Section B.1.2.126
1F8h	PARAM16_6	Parameter-set register	Section B.1.2.127
1FCh	PARAM16_7	Parameter-set register	Section B.1.2.128

Complex bit access types are encoded to fit into small table cells. Table 23 shows the codes that are used for access types in this section.

Table 23. DSS_HW_ACC_PARAM Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read

Table 23. DSS_HW_ACC_PARAM Access Type Codes (continued)

Access Type	Code	Description
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

B.1.2.1 PARAM1_0 Register (Offset = 0h) [reset = 0h]

PARAM1_0 is shown in Figure 41 and described in Table 24.

Return to Summary Table.

Parameter-set 0 (1st parameter-set out of 16), Register 0

Figure 41. PARAM1_0 Register

31	30	29	28	27	26	25	24
NU							FFT_OUTPUT_MODE
R/W-0h							R/W-0h
23	22	21	20	19	18	17	16
FFT_OUTPUT_MODE	ACCEL_MODE		CMULT_MODE			ABS EN	LOG2EN
R/W-0h	R/W-0h		R/W-0h			R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
WINDOW_EN	FFT_EN	BPM_EN	ACC2DMA_CHANNEL_TRIGDST				DMATRIGEN
R/W-0h	R/W-0h	R/W-0h	R/W-0h				R/W-0h
7	6	5	4	3	2	1	0
CR4INTREN	DMA2ACC_CHANNEL_TRIGSRC					TRIGMODE	
R/W-0h	R/W-0h					R/W-0h	

Table 24. PARAM1_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	NU	R/W	0h	Reserved for future use
24-23	FFT_OUTPUT_MODE	R/W	0h	Configures the output mode of the FFT Engine path. Refer REG_FFTOUT_MODE in the system spec document. 0xb - Default output (i.e., main data path output is sent out - eg. FFT or Log-Mag output) 10b - Statistics output - Max statistics is sent out (Max valiue on the I arm, Max index on the Q arm) 11b - Statistics output - Sum statistics is sent out
22-21	ACCEL_MODE	R/W	0h	Configures the mode of operation of the accelerator for the current parameter-set. Refer REG_ACCEL_MODE in the system spec document. 00b - FFT Engine path 01b - CFAR Engine path 10b - Reserved for future use 11b - NULL mode
20-18	CMULT_MODE	R/W	0h	Configuration for the Complex Multiplier block. Refer REG_CMULT_MODE in the system spec document. 000b - Disabled 001b to 111b - Enabled in one of 7 different modes (as explained in the system spec document)
17	ABS EN	R/W	0h	Enable/Disable for Magnitude computation. Refer REG_ABS_EN in the system spec document. When enabled, the magnitude of the input samples coming in to the Log-Magnitude block is calculated and sent out on the I-arm of the output. The Q-arm is made zeros.
16	LOG2EN	R/W	0h	Enable/Disable for Log2 computation. Refer REG_LOG2_EN in the system spec document. When enabled, the Log2 of the magnitude of the input samples coming in to the Log-Magnitude block is calculated and sent out on the I-arm of the output. The Q-arm is made zeros.
15	WINDOW_EN	R/W	0h	Enable/Disable for Windowing operation. Refer REG_WINDOW_EN in the system spec document.
14	FFT_EN	R/W	0h	Enable/Disable for FFT computation. Refer REG_FFT_EN in the system spec document.
13	BPM_EN	R/W	0h	Enable/Disable for BPM removal operation. Refer REG_BPM_EN in the system spec document. If set (i.e., enabled), then BPM removal is performed prior to feeding samples to the core computational unit.

Table 24. PARAM1_0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
12-9	ACC2DMA_CHANNEL_TRIGDST	R/W	0h	Selects which of the 16 DMA channels (allocated to the Radar Accelerator) should be triggered upon completion of the accelerator operation for the current parameter-set. Refer REG_ACC2DMA_TRIGDST in the system spec document.
8	DMATRIGEN	R/W	0h	Enable/disable trigger to DMA upon completion of the computational operations for the current parameter-set. Refer REG_DMATRIGEN in the system spec document. If enabled, the accelerator will trigger a specified DMA channel, so that the output samples can be shipped to R4F memory or Shared memory. There is a separate 4-bit register (see below), which is used to select which of the 16 DMA channels should be triggered upon completion of the accelerator operation for the current parameter-set.
7	CR4INTREN	R/W	0h	Enable/disable interrupt to the R4F upon completion of the computational operations for the current parameter-set. Refer REG_INTREN in the system spec document. If enabled, the R4F will receive an interrupt from the Accelerator at the completion of the current parameter-set.
6-3	DMA2ACC_CHANNEL_TRIGSRC	R/W	0h	This register is only relevant when TRIGMODE = 011b (DMA based trigger). This register selects the bit number in REG_DMA2ACCTRIG for the state machine to monitor in order to trigger the operation for the current parameter-set.
2-0	TRIGMODE	R/W	0h	Configures the trigger mode to start the computational operations for this parameter-set. Refer REG_TRIGMODE in the system spec document. 000b - Immediate trigger 001b - R4F based software trigger (In this mode, R4F writes a single register bit REG_CR42ACCTRIG to trigger the operation for the current parameter-set. The triggering is based on the state machine monitoring this register bit to be set) 010b - Ping-pong switch based trigger (Whenever the DFE switches from ping-to-pong buffer, or pong-to-ping buffer) 011b - DMA based trigger (In this mode, DMA writes a specific one-hot signature in the 16-bit register REG_DMA2ACCTRIG to trigger the operation for the current parameter-set. The triggering is based on the state machine monitoring bit number DMA2ACC_CHANNEL_TRIGSRC to be set)

B.1.2.2 PARAM1_1 Register (Offset = 4h) [reset = 0h]

PARAM1_1 is shown in Figure 42 and described in Table 25.

Return to Summary Table.

Parameter-set 0 (1st parameter-set out of 16), Register 1

Figure 42. PARAM1_1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSTADDR																SRCADDR															
R/W-0h																R/W-0h															

Table 25. PARAM1_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	DSTADDR	R/W	0h	Destination memory start address from which output samples should be written to. Refer REG_DSTADDR in the system spec document.
15-0	SRCADDR	R/W	0h	Source memory start address from which to pick input samples from. Refer REG_SRCADDR in the system spec document.

B.1.2.3 PARAM1_2 Register (Offset = 8h) [reset = 0h]

PARAM1_2 is shown in Figure 43 and described in Table 26.

Return to Summary Table.

Parameter-set 0 (1st parameter-set out of 16), Register 2

Figure 43. PARAM1_2 Register

31	30	29	28	27	26	25	24
DSTCONJ	DSTSIGNED	DST16b32b	DSTREAL	DSTACNT			
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h			
23	22	21	20	19	18	17	16
DSTACNT							
R/W-0h							
15	14	13	12	11	10	9	8
SRCCONJ	SRCSIGNED	SRC16b32b	SRCREAL	SRCACNT			
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h			
7	6	5	4	3	2	1	0
SRCACNT							
R/W-0h							

Table 26. PARAM1_2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DSTCONJ	R/W	0h	Enable/Disable Conjugation of the output samples. Refer REG_DSTCONJ in the system spec document. 0 = No conjugation, 1 = Enable conjugation This register only makes sense for complex output case (i.e., Log-Mag is not enabled). It is useful in conjunction with SRCCONJ to derive an IFFT mode (by conjugating at both input and output)
30	DSTSIGNED	R/W	0h	Signed or Unsigned data in the destination memory. Refer REG_DSTSIGNED in the system spec document. This register enables the output formatter to perform the correct sign-extension and/or saturation operation when writing the 24-bit output as 16-bit aligned or 32-bit aligned data in the destination memory.
29	DST16b32b	R/W	0h	16-bit or 32-bit alignment of data in destination memory. Refer REG_DST16b32b in the system spec document. 0 = 16-bit aligned, 1 = 32-bit aligned
28	DSTREAL	R/W	0h	Real or Complex data in the destination memory. Refer REG_DSTREAL in the system spec document. 0 = Complex output, 1 = Real output Real output is generally useful when Magnitude or Log-Magnitude computation is enabled.
27-16	DSTACNT	R/W	0h	Destination sample count. Refer REG_DSTACNT in the system spec document. Zero-based count of number of samples till which to write to the destination memory for each iteration. Note that this must be equal to or less than REG_SRCANT+any zero pads (applicable in case of FFT). Note that the actual number of samples written to destination memory would be REG_DSTACNT+1-REG_DST_SKIP_INIT.
15	SRCCONJ	R/W	0h	Enable/Disable Conjugation of the input samples. Refer REG_SRCCONJ in the system spec document. 0 = No conjugation, 1 = Enable conjugation This register only makes sense for complex input case. It is useful in conjunction with DSTCONJ to derive an IFFT mode (by conjugating at both input and output)
14	SRCSIGNED	R/W	0h	Signed or Unsigned data in the source memory. Refer REG_SRCSIGNED in the system spec document. This register is only relevant for 16-bit aligned data in source memory, because that is when sign-extension may be required to convert to internal 24-bit width. 0 = Unsigned, 1 = Signed.

Table 26. PARAM1_2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
13	SRC16b32b	R/W	0h	16-bit or 32-bit alignment of data in source memory. Refer REG_SRC16b32b in the system spec document. 0 = 16-bit aligned, 1 = 32-bit aligned
12	SRCREAL	R/W	0h	Real or Complex data in the source memory. Refer REG_SRCREAL in the system spec document. 0 = Complex input, 1 = Real input
11-0	SRCACNT	R/W	0h	Source sample count. Refer REG_SRCACNT in the system spec document. Zero-based count of number of samples to read from the source memory for each iteration.

B.1.2.4 PARAM1_3 Register (Offset = Ch) [reset = 0h]

PARAM1_3 is shown in Figure 44 and described in Table 27.

Return to Summary Table.

Parameter-set 0 (1st parameter-set out of 16), Register 3

Figure 44. PARAM1_3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSTAINDX																SRCAINDX															
R/W-0h																R/W-0h															

Table 27. PARAM1_3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	DSTAINDX	R/W	0h	Destination sample index increment. Refer REG_DSTAINDX in the system spec document. Specifies the number of bytes separating adjacent samples in the destination memory.
15-0	SRCAINDX	R/W	0h	Source sample index increment. Refer REG_SRCAINDX in the system spec document. Specifies the number of bytes separating adjacent samples in the source memory.

B.1.2.5 PARAM1_4 Register (Offset = 10h) [reset = 0h]

PARAM1_4 is shown in Figure 45 and described in Table 28.

Return to Summary Table.

Parameter-set 0 (1st parameter-set out of 16), Register 4

Figure 45. PARAM1_4 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSTBINDX																SRCBINDX															
R/W-0h																R/W-0h															

Table 28. PARAM1_4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	DSTBINDX	R/W	0h	Destination Offset across successive iterations. Refer REG_DSTBINDX in the system spec document. Specifies the number of bytes separating the start address of samples for successive iterations in the destination memory.
15-0	SRCBINDX	R/W	0h	Source Offset across successive iterations. Refer REG_SRCBINDX in the system spec document. Specifies the number of bytes separating the start address of samples for successive iterations in the source memory.

B.1.2.6 PARAM1_5 Register (Offset = 14h) [reset = 0h]

PARAM1_5 is shown in Figure 46 and described in Table 29.

Return to Summary Table.

Parameter-set 0 (1st parameter-set out of 16), Register 5

Figure 46. PARAM1_5 Register

31	30	29	28	27	26	25	24
NU		REG_DST_SKIP_INIT					
R/W-0h		R/W-0h					
23	22	21	20	19	18	17	16
REG_DST_SKIP_INIT				REG_DSTSCAL			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
REG_SRCSCAL				REG_BCNT			
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
REG_BCNT							
R/W-0h							

Table 29. PARAM1_5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	NU	R/W	0h	Reserved for future use
29-20	REG_DST_SKIP_INIT	R/W	0h	Destination skip sample count. Refer REG_DST_SKIP_INIT in the system spec document. Number of samples to skip in the beginning (for each iteration) before writing samples to the destination memory.
19-16	REG_DSTSCAL	R/W	0h	Output scaling. Refer REG_DSTSCAL in the system spec document. Specifies a programmable scaling, via right bit-shift, from 0 bits to 8 bits, for the output samples.
15-12	REG_SRCSCAL	R/W	0h	Input scaling. Refer REG_SRCSCAL in the system spec document. Specifies a programmable scaling, via right bit-shift, from 0 bits to 8 bits, for the input samples.
11-0	REG_BCNT	R/W	0h	Number of iterations. Refer REG_BCNT in the system spec document.

B.1.2.7 PARAM1_6 Register (Offset = 18h) [reset = 0h]

PARAM1_6 is shown in Figure 47 and described in Table 30.

Return to Summary Table.

Parameter-set 0 (1st parameter-set out of 16), Register 6

Figure 47. PARAM1_6 Register

31	30	29	28	27	26	25	24
BFLY_SCALING							
R/W-0h							
23	22	21	20	19	18	17	16
BFLY_SCALING		WINDOW_START					
R/W-0h		R/W-0h					
15	14	13	12	11	10	9	8
WINDOW_START				BPMPHASE			
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
INTERF_THRESH_EN	WINSYMM	FFTSIZE				DSTWIDTH	
R/W-0h	R/W-0h	R/W-0h				R/W-0h	

Table 30. PARAM1_6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-22	BFLY_SCALING	R/W	0h	Specifies the butterfly scaling for each of the 10 butterfly stages. Refer REG_BFLY_SCALING in the system spec document.
21-12	WINDOW_START	R/W	0h	Specifies the starting address of the window function in the Window RAM. Refer REG_WINDOW_START in the system spec document.
11-8	BPMPHASE	R/W	0h	Specifies the starting phase of the BPM pattern to be applied. Refer REG_BPMPHASE in the system spec document.
7	INTERF_THRESH_EN	R/W	0h	Enable/Disable for interference zeroing out. Refer REG_INTERF_THRESH_EN in the system spec document. If set, then input samples with large magnitude are zeroed out. Also see REG_INTERF_THRESH register which is a common register, not part of the parameter-set.
6	WINSYMM	R/W	0h	Indicates whether the window function is symmetric or not. Refer REG_WIMSYMM in the system spec document. If this register bit is set, only the first half the window coefficients are expected to be present in the window RAM. The same coefficients are read in reverse direction for the second half.
5-2	FFTSIZE	R/W	0h	Specifies FFT size. Actual FFT size is equal to 2 raised to the register value. Refer REG_FFTSIZE in the system spec document. For example, FFTSIZE = 0110b corresponds to actual FFT size of 2^6 = 64. Supported values are [1..10], which correspond to FFT sizes [2..1024].
1-0	DSTWIDTH	R/W	0h	Must be kept as 00b. Reserved for future use. (This was originally meant for bit-packing , such as bitwidths of 18-bits, 21-bits, etc. which is no longer supported).

B.1.2.8 PARAM1_7 Register (Offset = 1Ch) [reset = 0h]

PARAM1_7 is shown in Figure 48 and described in Table 31.

Return to Summary Table.

Parameter-set 0 (1st parameter-set out of 16), Register 7

Figure 48. PARAM1_7 Register

31	30	29	28	27	26	25	24
CIRCSHIFTWRAP				WINDOW_INTERP_FRACTION		TWIDINCR	
R/W-0h				R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
TWIDINCR							
R/W-0h							
15	14	13	12	11	10	9	8
TWIDINCR				CIRCIRSHIFT			
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
CIRCIRSHIFT							
R/W-0h							

Table 31. PARAM1_7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	CIRCSHIFTWRAP	R/W	0h	Indicates at what number (power-of-2) the sample counter value should wraparound, when using circular shift. Refer REG_CIRCSHIFTWRAP in the system spec document.
27-26	WINDOW_INTERP_FRACTION	R/W	0h	Configures linear interpolation for the window coefficients, which is relevant for large-size FFT computation (i.e., FFT sizes of 2K, 4K) that is obtained via stitching multiple smaller-size FFTs. Refer REG_WINDOW_INTERP_FRACTION in the system spec document.
25-12	TWIDINCR	R/W	0h	Configures the nature of frequency de-rotation done in the Complex Multiplier block. This meaning of this register depends on the Complex Multipler mode of operation. Refer REG_TWIDINCR in the system spec document.
11-0	CIRCIRSHIFT	R/W	0h	Specifies the circular shift (offset in samples) that should be applied to the sequence of input samples before feeding them to the core computational unit. Refer REG_SRCSHIFT in the system spec document.

B.1.2.9 PARAM2_0 Register (Offset = 20h)

PARAM2_0 is shown in Figure 49 and described in Table 32.

Return to Summary Table.

Parameter-set 1 (2nd parameter-set out of 16), Register 0

Figure 49. PARAM2_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 32. PARAM2_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.10 PARAM2_1 Register (Offset = 24h)

PARAM2_1 is shown in Figure 50 and described in Table 33.

Return to Summary Table.

Parameter-set 1 (2nd parameter-set out of 16), Register 1

Figure 50. PARAM2_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 33. PARAM2_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.11 PARAM2_2 Register (Offset = 28h)

PARAM2_2 is shown in Figure 51 and described in Table 34.

Return to Summary Table.

Parameter-set 1 (2nd parameter-set out of 16), Register 2

Figure 51. PARAM2_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 34. PARAM2_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.12 PARAM2_3 Register (Offset = 2Ch)

PARAM2_3 is shown in Figure 52 and described in Table 35.

Return to Summary Table.

Parameter-set 1 (2nd parameter-set out of 16), Register 3

Figure 52. PARAM2_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 35. PARAM2_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.13 PARAM2_4 Register (Offset = 30h)

PARAM2_4 is shown in Figure 53 and described in Table 36.

Return to Summary Table.

Parameter-set 1 (2nd parameter-set out of 16), Register 4

Figure 53. PARAM2_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 36. PARAM2_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.14 PARAM2_5 Register (Offset = 34h)

PARAM2_5 is shown in Figure 54 and described in Table 37.

Return to Summary Table.

Parameter-set 1 (2nd parameter-set out of 16), Register 5

Figure 54. PARAM2_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 37. PARAM2_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.15 PARAM2_6 Register (Offset = 38h)

PARAM2_6 is shown in Figure 55 and described in Table 38.

Return to Summary Table.

Parameter-set 1 (2nd parameter-set out of 16), Register 6

Figure 55. PARAM2_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 38. PARAM2_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.16 PARAM2_7 Register (Offset = 3Ch)

PARAM2_7 is shown in Figure 56 and described in Table 39.

Return to Summary Table.

Parameter-set 1 (2nd parameter-set out of 16), Register 7

Figure 56. PARAM2_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 39. PARAM2_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.17 PARAM3_0 Register (Offset = 40h)

PARAM3_0 is shown in Figure 57 and described in Table 40.

Return to Summary Table.

Parameter-set 2 (3rd parameter-set out of 16), Register 0

Figure 57. PARAM3_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 40. PARAM3_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.18 PARAM3_1 Register (Offset = 44h)

PARAM3_1 is shown in Figure 58 and described in Table 41.

Return to Summary Table.

Parameter-set 2 (3rd parameter-set out of 16), Register 1

Figure 58. PARAM3_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 41. PARAM3_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.19 PARAM3_2 Register (Offset = 48h)

PARAM3_2 is shown in Figure 59 and described in Table 42.

Return to Summary Table.

Parameter-set 2 (3rd parameter-set out of 16), Register 2

Figure 59. PARAM3_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 42. PARAM3_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.20 PARAM3_3 Register (Offset = 4Ch)

PARAM3_3 is shown in Figure 60 and described in Table 43.

Return to Summary Table.

Parameter-set 2 (3rd parameter-set out of 16), Register 3

Figure 60. PARAM3_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 43. PARAM3_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.21 PARAM3_4 Register (Offset = 50h)

PARAM3_4 is shown in Figure 61 and described in Table 44.

Return to Summary Table.

Parameter-set 2 (3rd parameter-set out of 16), Register 4

Figure 61. PARAM3_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 44. PARAM3_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.22 PARAM3_5 Register (Offset = 54h)

PARAM3_5 is shown in Figure 62 and described in Table 45.

Return to Summary Table.

Parameter-set 2 (3rd parameter-set out of 16), Register 5

Figure 62. PARAM3_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 45. PARAM3_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.23 PARAM3_6 Register (Offset = 58h)

PARAM3_6 is shown in Figure 63 and described in Table 46.

Return to Summary Table.

Parameter-set 2 (3rd parameter-set out of 16), Register 6

Figure 63. PARAM3_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 46. PARAM3_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.24 PARAM3_7 Register (Offset = 5Ch)

PARAM3_7 is shown in Figure 64 and described in Table 47.

Return to Summary Table.

Parameter-set 2 (3rd parameter-set out of 16), Register 7

Figure 64. PARAM3_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 47. PARAM3_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.25 PARAM4_0 Register (Offset = 60h)

PARAM4_0 is shown in Figure 65 and described in Table 48.

Return to Summary Table.

Parameter-set 3 (4th parameter-set out of 16), Register 0

Figure 65. PARAM4_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 48. PARAM4_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.26 PARAM4_1 Register (Offset = 64h)

PARAM4_1 is shown in Figure 66 and described in Table 49.

Return to Summary Table.

Parameter-set 3 (4th parameter-set out of 16), Register 1

Figure 66. PARAM4_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 49. PARAM4_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.27 PARAM4_2 Register (Offset = 68h)

PARAM4_2 is shown in Figure 67 and described in Table 50.

Return to Summary Table.

Parameter-set 3 (4th parameter-set out of 16), Register 2

Figure 67. PARAM4_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 50. PARAM4_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.28 PARAM4_3 Register (Offset = 6Ch)

PARAM4_3 is shown in Figure 68 and described in Table 51.

Return to Summary Table.

Parameter-set 3 (4th parameter-set out of 16), Register 3

Figure 68. PARAM4_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 51. PARAM4_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.29 PARAM4_4 Register (Offset = 70h)

PARAM4_4 is shown in Figure 69 and described in Table 52.

Return to Summary Table.

Parameter-set 3 (4th parameter-set out of 16), Register 4

Figure 69. PARAM4_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 52. PARAM4_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.30 PARAM4_5 Register (Offset = 74h)

PARAM4_5 is shown in Figure 70 and described in Table 53.

Return to Summary Table.

Parameter-set 3 (4th parameter-set out of 16), Register 5

Figure 70. PARAM4_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 53. PARAM4_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.31 PARAM4_6 Register (Offset = 78h)

PARAM4_6 is shown in Figure 71 and described in Table 54.

Return to Summary Table.

Parameter-set 3 (4th parameter-set out of 16), Register 6

Figure 71. PARAM4_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 54. PARAM4_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.32 PARAM4_7 Register (Offset = 7Ch)

PARAM4_7 is shown in Figure 72 and described in Table 55.

Return to Summary Table.

Parameter-set 3 (4th parameter-set out of 16), Register 7

Figure 72. PARAM4_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 55. PARAM4_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.33 PARAM5_0 Register (Offset = 80h)

PARAM5_0 is shown in Figure 73 and described in Table 56.

Return to Summary Table.

Parameter-set 4 (5th parameter-set out of 16), Register 0

Figure 73. PARAM5_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 56. PARAM5_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.34 PARAM5_1 Register (Offset = 84h)

PARAM5_1 is shown in Figure 74 and described in Table 57.

Return to Summary Table.

Parameter-set 4 (5th parameter-set out of 16), Register 1

Figure 74. PARAM5_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 57. PARAM5_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.35 PARAM5_2 Register (Offset = 88h)

PARAM5_2 is shown in Figure 75 and described in Table 58.

Return to Summary Table.

Parameter-set 4 (5th parameter-set out of 16), Register 2

Figure 75. PARAM5_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 58. PARAM5_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.36 PARAM5_3 Register (Offset = 8Ch)

PARAM5_3 is shown in Figure 76 and described in Table 59.

Return to Summary Table.

Parameter-set 4 (5th parameter-set out of 16), Register 3

Figure 76. PARAM5_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 59. PARAM5_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.37 PARAM5_4 Register (Offset = 90h)

PARAM5_4 is shown in Figure 77 and described in Table 60.

Return to Summary Table.

Parameter-set 4 (5th parameter-set out of 16), Register 4

Figure 77. PARAM5_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 60. PARAM5_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.38 PARAM5_5 Register (Offset = 94h)

PARAM5_5 is shown in Figure 78 and described in Table 61.

Return to Summary Table.

Parameter-set 4 (5th parameter-set out of 16), Register 5

Figure 78. PARAM5_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 61. PARAM5_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.39 PARAM5_6 Register (Offset = 98h)

PARAM5_6 is shown in Figure 79 and described in Table 62.

Return to Summary Table.

Parameter-set 4 (5th parameter-set out of 16), Register 6

Figure 79. PARAM5_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 62. PARAM5_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.40 PARAM5_7 Register (Offset = 9Ch)

PARAM5_7 is shown in Figure 80 and described in Table 63.

Return to Summary Table.

Parameter-set 4 (5th parameter-set out of 16), Register 7

Figure 80. PARAM5_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 63. PARAM5_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.41 PARAM6_0 Register (Offset = A0h)

PARAM6_0 is shown in Figure 81 and described in Table 64.

Return to Summary Table.

Parameter-set 5 (6th parameter-set out of 16), Register 0

Figure 81. PARAM6_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 64. PARAM6_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.42 PARAM6_1 Register (Offset = A4h)

PARAM6_1 is shown in Figure 82 and described in Table 65.

Return to Summary Table.

Parameter-set 5 (6th parameter-set out of 16), Register 1

Figure 82. PARAM6_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 65. PARAM6_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.43 PARAM6_2 Register (Offset = A8h)

PARAM6_2 is shown in Figure 83 and described in Table 66.

Return to Summary Table.

Parameter-set 5 (6th parameter-set out of 16), Register 2

Figure 83. PARAM6_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 66. PARAM6_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.44 PARAM6_3 Register (Offset = ACh)

PARAM6_3 is shown in Figure 84 and described in Table 67.

Return to Summary Table.

Parameter-set 5 (6th parameter-set out of 16), Register 3

Figure 84. PARAM6_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 67. PARAM6_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.45 PARAM6_4 Register (Offset = B0h)

PARAM6_4 is shown in Figure 85 and described in Table 68.

Return to Summary Table.

Parameter-set 5 (6th parameter-set out of 16), Register 4

Figure 85. PARAM6_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 68. PARAM6_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.46 PARAM6_5 Register (Offset = B4h)

PARAM6_5 is shown in Figure 86 and described in Table 69.

Return to Summary Table.

Parameter-set 5 (6th parameter-set out of 16), Register 5

Figure 86. PARAM6_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 69. PARAM6_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.47 PARAM6_6 Register (Offset = B8h)

PARAM6_6 is shown in Figure 87 and described in Table 70.

Return to Summary Table.

Parameter-set 5 (6th parameter-set out of 16), Register 6

Figure 87. PARAM6_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 70. PARAM6_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.48 PARAM6_7 Register (Offset = BCh)

PARAM6_7 is shown in Figure 88 and described in Table 71.

Return to Summary Table.

Parameter-set 5 (6th parameter-set out of 16), Register 7

Figure 88. PARAM6_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 71. PARAM6_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.49 PARAM7_0 Register (Offset = C0h)

PARAM7_0 is shown in Figure 89 and described in Table 72.

Return to Summary Table.

Parameter-set 6 (7th parameter-set out of 16), Register 0

Figure 89. PARAM7_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 72. PARAM7_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.50 PARAM7_1 Register (Offset = C4h)

PARAM7_1 is shown in Figure 90 and described in Table 73.

Return to Summary Table.

Parameter-set 6 (7th parameter-set out of 16), Register 1

Figure 90. PARAM7_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 73. PARAM7_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.51 PARAM7_2 Register (Offset = C8h)

PARAM7_2 is shown in Figure 91 and described in Table 74.

Return to Summary Table.

Parameter-set 6 (7th parameter-set out of 16), Register 2

Figure 91. PARAM7_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 74. PARAM7_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.52 PARAM7_3 Register (Offset = CCh)

PARAM7_3 is shown in Figure 92 and described in Table 75.

Return to Summary Table.

Parameter-set 6 (7th parameter-set out of 16), Register 3

Figure 92. PARAM7_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 75. PARAM7_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.53 PARAM7_4 Register (Offset = D0h)

PARAM7_4 is shown in Figure 93 and described in Table 76.

Return to Summary Table.

Parameter-set 6 (7th parameter-set out of 16), Register 4

Figure 93. PARAM7_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 76. PARAM7_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.54 PARAM7_5 Register (Offset = D4h)

PARAM7_5 is shown in Figure 94 and described in Table 77.

Return to Summary Table.

Parameter-set 6 (7th parameter-set out of 16), Register 5

Figure 94. PARAM7_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 77. PARAM7_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.55 PARAM7_6 Register (Offset = D8h)

PARAM7_6 is shown in Figure 95 and described in Table 78.

Return to Summary Table.

Parameter-set 6 (7th parameter-set out of 16), Register 6

Figure 95. PARAM7_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 78. PARAM7_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.56 PARAM7_7 Register (Offset = DCh)

PARAM7_7 is shown in Figure 96 and described in Table 79.

Return to Summary Table.

Parameter-set 6 (7th parameter-set out of 16), Register 7

Figure 96. PARAM7_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 79. PARAM7_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.57 PARAM8_0 Register (Offset = E0h)

PARAM8_0 is shown in Figure 97 and described in Table 80.

Return to Summary Table.

Parameter-set 7 (8th parameter-set out of 16), Register 0

Figure 97. PARAM8_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 80. PARAM8_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.58 PARAM8_1 Register (Offset = E4h)

PARAM8_1 is shown in Figure 98 and described in Table 81.

Return to Summary Table.

Parameter-set 7 (8th parameter-set out of 16), Register 1

Figure 98. PARAM8_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 81. PARAM8_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.59 PARAM8_2 Register (Offset = E8h)

PARAM8_2 is shown in Figure 99 and described in Table 82.

Return to Summary Table.

Parameter-set 7 (8th parameter-set out of 16), Register 2

Figure 99. PARAM8_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 82. PARAM8_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.60 PARAM8_3 Register (Offset = ECh)

PARAM8_3 is shown in Figure 100 and described in Table 83.

Return to Summary Table.

Parameter-set 7 (8th parameter-set out of 16), Register 3

Figure 100. PARAM8_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 83. PARAM8_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.61 PARAM8_4 Register (Offset = F0h)

PARAM8_4 is shown in Figure 101 and described in Table 84.

Return to Summary Table.

Parameter-set 7 (8th parameter-set out of 16), Register 4

Figure 101. PARAM8_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 84. PARAM8_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.62 PARAM8_5 Register (Offset = F4h)

PARAM8_5 is shown in Figure 102 and described in Table 85.

Return to Summary Table.

Parameter-set 7 (8th parameter-set out of 16), Register 5

Figure 102. PARAM8_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 85. PARAM8_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.63 PARAM8_6 Register (Offset = F8h)

PARAM8_6 is shown in Figure 103 and described in Table 86.

Return to Summary Table.

Parameter-set 7 (8th parameter-set out of 16), Register 6

Figure 103. PARAM8_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 86. PARAM8_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.64 PARAM8_7 Register (Offset = FCh)

PARAM8_7 is shown in Figure 104 and described in Table 87.

Return to Summary Table.

Parameter-set 7 (8th parameter-set out of 16), Register 7

Figure 104. PARAM8_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 87. PARAM8_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.65 PARAM9_0 Register (Offset = 100h)

PARAM9_0 is shown in Figure 105 and described in Table 88.

Return to Summary Table.

Parameter-set 8 (9th parameter-set out of 16), Register 0

Figure 105. PARAM9_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 88. PARAM9_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.66 PARAM9_1 Register (Offset = 104h)

PARAM9_1 is shown in Figure 106 and described in Table 89.

Return to Summary Table.

Parameter-set 8 (9th parameter-set out of 16), Register 1

Figure 106. PARAM9_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 89. PARAM9_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.67 PARAM9_2 Register (Offset = 108h)

PARAM9_2 is shown in Figure 107 and described in Table 90.

Return to Summary Table.

Parameter-set 8 (9th parameter-set out of 16), Register 2

Figure 107. PARAM9_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 90. PARAM9_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.68 PARAM9_3 Register (Offset = 10Ch)

PARAM9_3 is shown in Figure 108 and described in Table 91.

Return to Summary Table.

Parameter-set 8 (9th parameter-set out of 16), Register 3

Figure 108. PARAM9_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 91. PARAM9_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.69 PARAM9_4 Register (Offset = 110h)

PARAM9_4 is shown in Figure 109 and described in Table 92.

Return to Summary Table.

Parameter-set 8 (9th parameter-set out of 16), Register 4

Figure 109. PARAM9_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 92. PARAM9_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.70 PARAM9_5 Register (Offset = 114h)

PARAM9_5 is shown in Figure 110 and described in Table 93.

Return to Summary Table.

Parameter-set 8 (9th parameter-set out of 16), Register 5

Figure 110. PARAM9_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 93. PARAM9_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.71 PARAM9_6 Register (Offset = 118h)

PARAM9_6 is shown in Figure 111 and described in Table 94.

Return to Summary Table.

Parameter-set 8 (9th parameter-set out of 16), Register 6

Figure 111. PARAM9_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 94. PARAM9_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.72 PARAM9_7 Register (Offset = 11Ch)

PARAM9_7 is shown in Figure 112 and described in Table 95.

Return to Summary Table.

Parameter-set 8 (9th parameter-set out of 16), Register 7

Figure 112. PARAM9_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 95. PARAM9_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.73 PARAM10_0 Register (Offset = 120h)

PARAM10_0 is shown in Figure 113 and described in Table 96.

Return to Summary Table.

Parameter-set 9 (10th parameter-set out of 16), Register 0

Figure 113. PARAM10_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 96. PARAM10_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.74 PARAM10_1 Register (Offset = 124h)

PARAM10_1 is shown in Figure 114 and described in Table 97.

Return to Summary Table.

Parameter-set 9 (10th parameter-set out of 16), Register 1

Figure 114. PARAM10_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 97. PARAM10_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.75 PARAM10_2 Register (Offset = 128h)

PARAM10_2 is shown in Figure 115 and described in Table 98.

Return to Summary Table.

Parameter-set 9 (10th parameter-set out of 16), Register 2

Figure 115. PARAM10_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 98. PARAM10_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.76 PARAM10_3 Register (Offset = 12Ch)

PARAM10_3 is shown in Figure 116 and described in Table 99.

Return to Summary Table.

Parameter-set 9 (10th parameter-set out of 16), Register 3

Figure 116. PARAM10_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 99. PARAM10_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.77 PARAM10_4 Register (Offset = 130h)

PARAM10_4 is shown in Figure 117 and described in Table 100.

Return to Summary Table.

Parameter-set 9 (10th parameter-set out of 16), Register 4

Figure 117. PARAM10_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 100. PARAM10_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.78 PARAM10_5 Register (Offset = 134h)

PARAM10_5 is shown in Figure 118 and described in Table 101.

Return to Summary Table.

Parameter-set 9 (10th parameter-set out of 16), Register 5

Figure 118. PARAM10_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 101. PARAM10_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.79 PARAM10_6 Register (Offset = 138h)

PARAM10_6 is shown in Figure 119 and described in Table 102.

Return to Summary Table.

Parameter-set 9 (10th parameter-set out of 16), Register 6

Figure 119. PARAM10_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 102. PARAM10_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.80 PARAM10_7 Register (Offset = 13Ch)

PARAM10_7 is shown in Figure 120 and described in Table 103.

Return to Summary Table.

Parameter-set 9 (10th parameter-set out of 16), Register 7

Figure 120. PARAM10_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 103. PARAM10_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.81 PARAM11_0 Register (Offset = 140h)

PARAM11_0 is shown in Figure 121 and described in Table 104.

Return to Summary Table.

Parameter-set 10 (11th parameter-set out of 16), Register 0

Figure 121. PARAM11_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 104. PARAM11_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.82 PARAM11_1 Register (Offset = 144h)

PARAM11_1 is shown in Figure 122 and described in Table 105.

Return to Summary Table.

Parameter-set 10 (11th parameter-set out of 16), Register 1

Figure 122. PARAM11_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 105. PARAM11_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.83 PARAM11_2 Register (Offset = 148h)

PARAM11_2 is shown in Figure 123 and described in Table 106.

Return to Summary Table.

Parameter-set 10 (11th parameter-set out of 16), Register 2

Figure 123. PARAM11_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 106. PARAM11_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.84 PARAM11_3 Register (Offset = 14Ch)

PARAM11_3 is shown in Figure 124 and described in Table 107.

Return to Summary Table.

Parameter-set 10 (11th parameter-set out of 16), Register 3

Figure 124. PARAM11_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 107. PARAM11_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.85 PARAM11_4 Register (Offset = 150h)

PARAM11_4 is shown in Figure 125 and described in Table 108.

Return to Summary Table.

Parameter-set 10 (11th parameter-set out of 16), Register 4

Figure 125. PARAM11_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 108. PARAM11_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.86 PARAM11_5 Register (Offset = 154h)

PARAM11_5 is shown in Figure 126 and described in Table 109.

Return to Summary Table.

Parameter-set 10 (11th parameter-set out of 16), Register 5

Figure 126. PARAM11_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 109. PARAM11_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.87 PARAM11_6 Register (Offset = 158h)

PARAM11_6 is shown in Figure 127 and described in Table 110.

Return to Summary Table.

Parameter-set 10 (11th parameter-set out of 16), Register 6

Figure 127. PARAM11_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 110. PARAM11_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.88 PARAM11_7 Register (Offset = 15Ch)

PARAM11_7 is shown in Figure 128 and described in Table 111.

Return to Summary Table.

Parameter-set 10 (11th parameter-set out of 16), Register 7

Figure 128. PARAM11_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 111. PARAM11_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.89 PARAM12_0 Register (Offset = 160h)

PARAM12_0 is shown in Figure 129 and described in Table 112.

Return to Summary Table.

Parameter-set 11 (12th parameter-set out of 16), Register 0

Figure 129. PARAM12_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 112. PARAM12_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.90 PARAM12_1 Register (Offset = 164h)

PARAM12_1 is shown in Figure 130 and described in Table 113.

Return to Summary Table.

Parameter-set 11 (12th parameter-set out of 16), Register 1

Figure 130. PARAM12_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 113. PARAM12_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.91 PARAM12_2 Register (Offset = 168h)

PARAM12_2 is shown in Figure 131 and described in Table 114.

Return to Summary Table.

Parameter-set 11 (12th parameter-set out of 16), Register 2

Figure 131. PARAM12_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 114. PARAM12_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.92 PARAM12_3 Register (Offset = 16Ch)

PARAM12_3 is shown in Figure 132 and described in Table 115.

Return to Summary Table.

Parameter-set 11 (12th parameter-set out of 16), Register 3

Figure 132. PARAM12_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 115. PARAM12_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.93 PARAM12_4 Register (Offset = 170h)

PARAM12_4 is shown in Figure 133 and described in Table 116.

Return to Summary Table.

Parameter-set 11 (12th parameter-set out of 16), Register 4

Figure 133. PARAM12_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 116. PARAM12_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.94 PARAM12_5 Register (Offset = 174h)

PARAM12_5 is shown in Figure 134 and described in Table 117.

Return to Summary Table.

Parameter-set 11 (12th parameter-set out of 16), Register 5

Figure 134. PARAM12_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 117. PARAM12_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.95 PARAM12_6 Register (Offset = 178h)

PARAM12_6 is shown in Figure 135 and described in Table 118.

Return to Summary Table.

Parameter-set 11 (12th parameter-set out of 16), Register 6

Figure 135. PARAM12_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 118. PARAM12_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.96 PARAM12_7 Register (Offset = 17Ch)

PARAM12_7 is shown in Figure 136 and described in Table 119.

Return to Summary Table.

Parameter-set 11 (12th parameter-set out of 16), Register 7

Figure 136. PARAM12_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 119. PARAM12_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.97 PARAM13_0 Register (Offset = 180h)

PARAM13_0 is shown in Figure 137 and described in Table 120.

Return to Summary Table.

Parameter-set 12 (13th parameter-set out of 16), Register 0

Figure 137. PARAM13_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 120. PARAM13_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.98 PARAM13_1 Register (Offset = 184h)

PARAM13_1 is shown in Figure 138 and described in Table 121.

Return to Summary Table.

Parameter-set 12 (13th parameter-set out of 16), Register 1

Figure 138. PARAM13_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 121. PARAM13_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.99 PARAM13_2 Register (Offset = 188h)

PARAM13_2 is shown in Figure 139 and described in Table 122.

Return to Summary Table.

Parameter-set 12 (13th parameter-set out of 16), Register 2

Figure 139. PARAM13_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 122. PARAM13_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.100 PARAM13_3 Register (Offset = 18Ch)

PARAM13_3 is shown in Figure 140 and described in Table 123.

Return to Summary Table.

Parameter-set 12 (13th parameter-set out of 16), Register 3

Figure 140. PARAM13_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 123. PARAM13_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.101 PARAM13_4 Register (Offset = 190h)

PARAM13_4 is shown in Figure 141 and described in Table 124.

Return to Summary Table.

Parameter-set 12 (13th parameter-set out of 16), Register 4

Figure 141. PARAM13_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 124. PARAM13_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.102 PARAM13_5 Register (Offset = 194h)

PARAM13_5 is shown in Figure 142 and described in Table 125.

Return to Summary Table.

Parameter-set 12 (13th parameter-set out of 16), Register 5

Figure 142. PARAM13_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 125. PARAM13_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.103 PARAM13_6 Register (Offset = 198h)

PARAM13_6 is shown in Figure 143 and described in Table 126.

Return to Summary Table.

Parameter-set 12 (13th parameter-set out of 16), Register 6

Figure 143. PARAM13_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 126. PARAM13_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.104 PARAM13_7 Register (Offset = 19Ch)

PARAM13_7 is shown in Figure 144 and described in Table 127.

Return to Summary Table.

Parameter-set 12 (13th parameter-set out of 16), Register 7

Figure 144. PARAM13_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 127. PARAM13_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.105 PARAM14_0 Register (Offset = 1A0h)

PARAM14_0 is shown in Figure 145 and described in Table 128.

Return to Summary Table.

Parameter-set 13 (14th parameter-set out of 16), Register 0

Figure 145. PARAM14_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 128. PARAM14_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.106 PARAM14_1 Register (Offset = 1A4h)

PARAM14_1 is shown in Figure 146 and described in Table 129.

Return to Summary Table.

Parameter-set 13 (14th parameter-set out of 16), Register 1

Figure 146. PARAM14_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 129. PARAM14_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.107 PARAM14_2 Register (Offset = 1A8h)

PARAM14_2 is shown in Figure 147 and described in Table 130.

Return to Summary Table.

Parameter-set 13 (14th parameter-set out of 16), Register 2

Figure 147. PARAM14_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 130. PARAM14_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.108 PARAM14_3 Register (Offset = 1ACh)

PARAM14_3 is shown in Figure 148 and described in Table 131.

Return to Summary Table.

Parameter-set 13 (14th parameter-set out of 16), Register 3

Figure 148. PARAM14_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 131. PARAM14_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.109 PARAM14_4 Register (Offset = 1B0h)

PARAM14_4 is shown in Figure 149 and described in Table 132.

Return to Summary Table.

Parameter-set 13 (14th parameter-set out of 16), Register 4

Figure 149. PARAM14_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 132. PARAM14_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.110 PARAM14_5 Register (Offset = 1B4h)

PARAM14_5 is shown in Figure 150 and described in Table 133.

Return to Summary Table.

Parameter-set 13 (14th parameter-set out of 16), Register 5

Figure 150. PARAM14_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 133. PARAM14_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.111 PARAM14_6 Register (Offset = 1B8h)

PARAM14_6 is shown in Figure 151 and described in Table 134.

Return to Summary Table.

Parameter-set 13 (14th parameter-set out of 16), Register 6

Figure 151. PARAM14_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 134. PARAM14_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.112 PARAM14_7 Register (Offset = 1BCh)

PARAM14_7 is shown in Figure 152 and described in Table 135.

Return to Summary Table.

Parameter-set 13 (14th parameter-set out of 16), Register 7

Figure 152. PARAM14_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 135. PARAM14_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.113 PARAM15_0 Register (Offset = 1C0h)

PARAM15_0 is shown in Figure 153 and described in Table 136.

Return to Summary Table.

Parameter-set 14 (15th parameter-set out of 16), Register 0

Figure 153. PARAM15_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 136. PARAM15_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.114 PARAM15_1 Register (Offset = 1C4h)

PARAM15_1 is shown in Figure 154 and described in Table 137.

Return to Summary Table.

Parameter-set 14 (15th parameter-set out of 16), Register 1

Figure 154. PARAM15_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 137. PARAM15_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.115 PARAM15_2 Register (Offset = 1C8h)

PARAM15_2 is shown in Figure 155 and described in Table 138.

Return to Summary Table.

Parameter-set 14 (15th parameter-set out of 16), Register 2

Figure 155. PARAM15_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 138. PARAM15_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.116 PARAM15_3 Register (Offset = 1CCh)

PARAM15_3 is shown in Figure 156 and described in Table 139.

Return to Summary Table.

Parameter-set 14 (15th parameter-set out of 16), Register 3

Figure 156. PARAM15_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 139. PARAM15_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.117 PARAM15_4 Register (Offset = 1D0h)

PARAM15_4 is shown in Figure 157 and described in Table 140.

Return to Summary Table.

Parameter-set 14 (15th parameter-set out of 16), Register 4

Figure 157. PARAM15_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 140. PARAM15_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.118 PARAM15_5 Register (Offset = 1D4h)

PARAM15_5 is shown in Figure 158 and described in Table 141.

Return to Summary Table.

Parameter-set 14 (15th parameter-set out of 16), Register 5

Figure 158. PARAM15_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 141. PARAM15_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.119 PARAM15_6 Register (Offset = 1D8h)

PARAM15_6 is shown in Figure 159 and described in Table 142.

Return to Summary Table.

Parameter-set 14 (15th parameter-set out of 16), Register 6

Figure 159. PARAM15_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 142. PARAM15_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.120 PARAM15_7 Register (Offset = 1DCh)

PARAM15_7 is shown in Figure 160 and described in Table 143.

Return to Summary Table.

Parameter-set 14 (15th parameter-set out of 16), Register 7

Figure 160. PARAM15_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 143. PARAM15_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.121 PARAM16_0 Register (Offset = 1E0h)

PARAM16_0 is shown in Figure 161 and described in Table 144.

Return to Summary Table.

Parameter-set 15 (16th parameter-set out of 16), Register 0

Figure 161. PARAM16_0 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 144. PARAM16_0 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.122 PARAM16_1 Register (Offset = 1E4h)

PARAM16_1 is shown in Figure 162 and described in Table 145.

Return to Summary Table.

Parameter-set 15 (16th parameter-set out of 16), Register 1

Figure 162. PARAM16_1 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 145. PARAM16_1 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.123 PARAM16_2 Register (Offset = 1E8h)

PARAM16_2 is shown in Figure 163 and described in Table 146.

Return to Summary Table.

Parameter-set 15 (16th parameter-set out of 16), Register 2

Figure 163. PARAM16_2 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 146. PARAM16_2 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.124 PARAM16_3 Register (Offset = 1ECh)

PARAM16_3 is shown in Figure 164 and described in Table 147.

Return to Summary Table.

Parameter-set 15 (16th parameter-set out of 16), Register 3

Figure 164. PARAM16_3 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 147. PARAM16_3 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.125 PARAM16_4 Register (Offset = 1F0h)

PARAM16_4 is shown in Figure 165 and described in Table 148.

Return to Summary Table.

Parameter-set 15 (16th parameter-set out of 16), Register 4

Figure 165. PARAM16_4 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 148. PARAM16_4 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.126 PARAM16_5 Register (Offset = 1F4h)

PARAM16_5 is shown in Figure 166 and described in Table 149.

Return to Summary Table.

Parameter-set 15 (16th parameter-set out of 16), Register 5

Figure 166. PARAM16_5 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 149. PARAM16_5 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.127 PARAM16_6 Register (Offset = 1F8h)

PARAM16_6 is shown in Figure 167 and described in Table 150.

Return to Summary Table.

Parameter-set 15 (16th parameter-set out of 16), Register 6

Figure 167. PARAM16_6 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 150. PARAM16_6 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.2.128 PARAM16_7 Register (Offset = 1FCh)

PARAM16_7 is shown in Figure 168 and described in Table 151.

Return to Summary Table.

Parameter-set 15 (16th parameter-set out of 16), Register 7

Figure 168. PARAM16_7 Register

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0

Table 151. PARAM16_7 Register Field Descriptions

Bit	Field	Type	Reset	Description

B.1.3 Common Registers

The list of common registers, which are not part of the parameter-set and thus are common to all the 16 parameter-sets, is tabulated below.

B.1.4 Debug Registers

This section describes registers useful for debugging. These are not part of the parameter-set register map.

For debug: **Parameter-set Current Index (REG_PARAMCURR) – 4 bits:**

(Also, REG_LOOPCOU – 12 bits)

This common register is a read-only register, which indicates the index of the current parameter-set under execution. This is useful for debug, where parameter-sets can be executed one-by-one using the software trigger mode for each of them. In this debug, this register indicates which parameter-set is currently waiting for the software trigger. There is another read-only register, REG_LOOPCOU, which indicates the loop count that is presently running.

For debug: **Accelerator Trigger Status (REG_ACC_TRIG_IN_STAT) – 19 bits:**

(Also, ACC_TRIG_IN_CLR – 1 bit)

This common register is a read-only register that indicates the trigger status of the accelerator. The 16 MSB bits indicate whether a trigger was received through the DMA trigger method (see REG_TRIG_MODE). The next two bits (bit indices 2 and 1) indicate the status of the DFE ping-pong trigger and software trigger. The LSB bit is always 1, and can be ignored. There is a separate single-bit register, ACC_TRIG_IN_CLR, which can be set to clear the contents of the trigger status register.

B.1.5 DSS_HW_ACC_STATIC Registers

Table 152 lists the memory-mapped registers for the DSS_HW_ACC_STATIC. All register offset addresses not listed in Table 152 should be considered as reserved locations and the register contents should not be modified.

Table 152. DSS_HW_ACC_STATIC Registers

Offset	Acronym	Register Name	Section
0h	HWACCREG1		Section B.1.5.1
4h	HWACCREG2		Section B.1.5.2
8h	HWACCREG3		Section B.1.5.3
Ch	HWACCREG4		Section B.1.5.4
10h	HWACCREG5		Section B.1.5.5
14h	HWACCREG6		Section B.1.5.6
18h	HWACCREG7		Section B.1.5.7
1Ch	HWACCREG8		Section B.1.5.8
20h	HWACCREG9		Section B.1.5.9
24h	HWACCREG10		Section B.1.5.10
28h	HWACCREG11		Section B.1.5.11
2Ch	HWACCREG12		Section B.1.5.12
30h	HWACCREG13		Section B.1.5.13
34h	HWACCREG14		Section B.1.5.14
38h	HWACCREG15		Section B.1.5.15
3Ch	HWACCREG16		Section B.1.5.16
40h	MAX1VALUE		Section B.1.5.17
44h	MAX1INDEX		Section B.1.5.18
48h	ISUM1LSB		Section B.1.5.19
4Ch	ISUM1MSB		Section B.1.5.20
50h	QSUM1LSB		Section B.1.5.21
54h	QSUM1MSB		Section B.1.5.22
58h	MAX2VALUE		Section B.1.5.23
5Ch	MAX2INDEX		Section B.1.5.24
60h	ISUM2LSB		Section B.1.5.25
64h	ISUM2MSB		Section B.1.5.26
68h	QSUM2LSB		Section B.1.5.27
6Ch	QSUM2MSB		Section B.1.5.28
70h	MAX3VALUE		Section B.1.5.29
74h	MAX3INDEX		Section B.1.5.30
78h	ISUM3LSB		Section B.1.5.31
7Ch	ISUM3MSB		Section B.1.5.32
80h	QSUM3LSB		Section B.1.5.33
84h	QSUM3MSB		Section B.1.5.34
88h	MAX4VALUE		Section B.1.5.35
8Ch	MAX4INDEX		Section B.1.5.36
90h	ISUM4LSB		Section B.1.5.37
94h	ISUM4MSB		Section B.1.5.38
98h	QSUM4LSB		Section B.1.5.39
9Ch	QSUM4MSB		Section B.1.5.40
A0h	DCOFFSETI		Section B.1.5.41
A4h	DCOFFSETQ		Section B.1.5.42
A8h	CFARTEST		Section B.1.5.43

Table 152. DSS_HW_ACC_STATIC Registers (continued)

Offset	Acronym	Register Name	Section
ACh	RDSTATUS		Section B.1.5.44
B0h	SIGDMACH1DONE		Section B.1.5.45
B4h	SIGDMACH2DONE		Section B.1.5.46
B8h	SIGDMACH3DONE		Section B.1.5.47
BCh	SIGDMACH4DONE		Section B.1.5.48
C0h	SIGDMACH5DONE		Section B.1.5.49
C4h	SIGDMACH6DONE		Section B.1.5.50
C8h	SIGDMACH7DONE		Section B.1.5.51
CCh	SIGDMACH8DONE		Section B.1.5.52
D0h	SIGDMACH9DONE		Section B.1.5.53
D4h	SIGDMACH10DONE		Section B.1.5.54
D8h	SIGDMACH11DONE		Section B.1.5.55
DCh	SIGDMACH12DONE		Section B.1.5.56
E0h	SIGDMACH13DONE		Section B.1.5.57
E4h	SIGDMACH14DONE		Section B.1.5.58
E8h	SIGDMACH15DONE		Section B.1.5.59
ECh	SIGDMACH16DONE		Section B.1.5.60
F0h	MEMACCESSERR		Section B.1.5.61
F4h	FFTCLIP		Section B.1.5.62
F8h	FFTPEAKCNT		Section B.1.5.63
FCh	HWACCREG1RD		Section B.1.5.64
100h	HWACCREG2RD		Section B.1.5.65
104h	HWACCREG3RD		Section B.1.5.66

Complex bit access types are encoded to fit into small table cells. Table 153 shows the codes that are used for access types in this section.

Table 153. DSS_HW_ACC Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

B.1.5.1 HWACCREG1 Register (Offset = 0h) [reset = 0h]

HWACCREG1 is shown in Figure 169 and described in Table 154.

Return to Summary Table.

Figure 169. HWACCREG1 Register

31	30	29	28	27	26	25	24
NU			FFT1DEN	PARAMSTOP			
R-			R/W-0h	R/W-0h			
23	22	21	20	19	18	17	16
PARAMSTART				NLOOPS			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
NLOOPS							
R/W-0h							
7	6	5	4	3	2	1	0
NU2	ACCRESET			ACCCLKEN	ACCENABLE		
R-	R/W-0h			R/W-0h	R/W-0h		

Table 154. HWACCREG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	NU	R		
28	FFT1DEN	R/W	0h	
27-24	PARAMSTOP	R/W	0h	
23-20	PARAMSTART	R/W	0h	
19-8	NLOOPS	R/W	0h	
7	NU2	R		
6-4	ACCRESET	R/W	0h	0x7:Reset
3	ACCCLKEN	R/W	0h	1:Enable clock
2-0	ACCENABLE	R/W	0h	0x7 : Enable ;0x0 : Disable

B.1.5.2 HWACCREG2 Register (Offset = 4h) [reset = 0h]

HWACCREG2 is shown in Figure 170 and described in Table 155.

Return to Summary Table.

Figure 170. HWACCREG2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU																DMA2ACCTRIG															
R-																0h															

Table 155. HWACCREG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	NU	R		
15-0	DMA2ACCTRIG		0h	DMA completion indication corresponding to each DMA channel

B.1.5.3 HWACCREG3 Register (Offset = 8h) [reset = 0h]

HWACCREG3 is shown in Figure 171 and described in Table 156.

Return to Summary Table.

Figure 171. HWACCREG3 Register

31	30	29	28	27	26	25	24
CR42DMATRIG							
0h							
23	22	21	20	19	18	17	16
CR42DMATRIG							
0h							
15	14	13	12	11	10	9	8
NU							
R-							
7	6	5	4	3	2	1	0
NU							CR42ACCTRIG
R-							0h

Table 156. HWACCREG3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	CR42DMATRIG		0h	Trigger from CR4 to DMA.Can be used for trigerring the first and second DMA transfer
15-1	NU	R		
0	CR42ACCTRIG		0h	Trigger from CR4 to accelerator

B.1.5.4 HWACCREG4 Register (Offset = Ch) [reset = 0h]

HWACCREG4 is shown in Figure 172 and described in Table 157.

Return to Summary Table.

Figure 172. HWACCREG4 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PARAMDONECLR																PARAMDONESTAT															
0h																R-															

Table 157. HWACCREG4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	PARAMDONECLR		0h	Clear from CR4
15-0	PARAMDONESTAT	R		Status to CR4

B.1.5.5 HWACCREG5 Register (Offset = 10h) [reset = 0h]

HWACCREG5 is shown in Figure 173 and described in Table 158.

Return to Summary Table.

Figure 173. HWACCREG5 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BPMPATTERNMSB																															
R/W-0h																															

Table 158. HWACCREG5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	BPMPATTERNMSB	R/W	0h	

B.1.5.6 HWACCREG6 Register (Offset = 14h) [reset = 0h]

HWACCREG6 is shown in Figure 174 and described in Table 159.

Return to Summary Table.

Figure 174. HWACCREG6 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BPMPATTERNLSB																															
R/W-0h																															

Table 159. HWACCREG6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	BPMPATTERNLSB	R/W	0h	

B.1.5.7 HWACCREG7 Register (Offset = 18h) [reset = 0h]

HWACCREG7 is shown in Figure 175 and described in Table 160.

Return to Summary Table.

Figure 175. HWACCREG7 Register

31	30	29	28	27	26	25	24
NU3							STG1LUTSELWR
R-							R/W-0h
23	22	21	20	19	18	17	16
NU2							DITHERTWIDEN
R-							R/W-0h
15	14	13	12	11	10	9	8
NU1						BPMRATE	
R-						R/W-0h	
7	6	5	4	3	2	1	0
BPMRATE							
R/W-0h							

Table 160. HWACCREG7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	NU3	R		
24	STG1LUTSELWR	R/W	0h	0' -> Window LUT is selected for writing 'through Bus Matrix 1' -> FFT 1st stage RAM is selected for writing through Bus matrix
23-17	NU2	R		
16	DITHERTWIDEN	R/W	0h	
15-10	NU1	R		
9-0	BPMRATE	R/W	0h	

B.1.5.8 HWACCREG8 Register (Offset = 1Ch) [reset = 0h]

HWACCREG8 is shown in Figure 176 and described in Table 161.

Return to Summary Table.

Figure 176. HWACCREG8 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NU			FFTSUMDIV					INTERFTHRESH							
R-			R/W-0h					R/W-0h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTERFTHRESH															
R/W-0h															

Table 161. HWACCREG8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	NU	R		
28-24	FFTSUMDIV	R/W	0h	
23-0	INTERFTHRESH	R/W	0h	

B.1.5.9 HWACCREG9 Register (Offset = 20h) [reset = 0h]

HWACCREG9 is shown in Figure 177 and described in Table 162.

Return to Summary Table.

Figure 177. HWACCREG9 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU											ICMULTSCALE																				
R-											R/W-0h																				

Table 162. HWACCREG9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-21	NU	R		
20-0	ICMULTSCALE	R/W	0h	

B.1.5.10 HWACCREG10 Register (Offset = 24h) [reset = 0h]

HWACCREG10 is shown in Figure 178 and described in Table 163.

Return to Summary Table.

Figure 178. HWACCREG10 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU											QCMULTSCALE																				
R-											R/W-0h																				

Table 163. HWACCREG10 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-21	NU	R		
20-0	QCMULTSCALE	R/W	0h	

B.1.5.11 HWACCREG11 Register (Offset = 28h) [reset = 0h]

HWACCREG11 is shown in Figure 179 and described in Table 164.

Return to Summary Table.

Figure 179. HWACCREG11 Register

31	30	29	28	27	26	25	24
LFSRLOAD	NU		LFSRSEED				
0h	R-		R/W-0h				
23	22	21	20	19	18	17	16
LFSRSEED							
R/W-0h							
15	14	13	12	11	10	9	8
LFSRSEED							
R/W-0h							
7	6	5	4	3	2	1	0
LFSRSEED							
R/W-0h							

Table 164. HWACCREG11 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	LFSRLOAD		0h	
30-29	NU	R		
28-0	LFSRSEED	R/W	0h	

B.1.5.12 HWACCREG12 Register (Offset = 2Ch) [reset = 0h]

HWACCREG12 is shown in Figure 180 and described in Table 165.

Return to Summary Table.

Figure 180. HWACCREG12 Register

31	30	29	28	27	26	25	24
NU2							ACC_TRIGGER_IN_CLR
R/W-							0h
23	22	21	20	19	18	17	16
NU1					ACC_TRIGGER_IN_STAT		
R-					R-		
15	14	13	12	11	10	9	8
ACC_TRIGGER_IN_STAT							
R-							
7	6	5	4	3	2	1	0
ACC_TRIGGER_IN_STAT							
R-							

Table 165. HWACCREG12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	NU2	R/W		
24	ACC_TRIGGER_IN_CLR		0h	
23-19	NU1	R		
18-0	ACC_TRIGGER_IN_STAT	R		{dma2acctrig[15:0],adc_buffer_done,cr42acctrig,1}

B.1.5.13 HWACCREG13 Register (Offset = 30h) [reset = 0h]

HWACCREG13 is shown in Figure 181 and described in Table 166.

Return to Summary Table.

Figure 181. HWACCREG13 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU														CFAR_THRESH																	
R-														R/W-0h																	

Table 166. HWACCREG13 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	NU	R		
17-0	CFAR_THRESH	R/W	0h	CFAR threshold

B.1.5.14 HWACCREG14 Register (Offset = 34h) [reset = 770h]

HWACCREG14 is shown in Figure 182 and described in Table 167.

Return to Summary Table.

Figure 182. HWACCREG14 Register

31	30	29	28	27	26	25	24
NU3							
R/W-							
23	22	21	20	19	18	17	16
NU3							
R/W-							
15	14	13	12	11	10	9	8
NU3					OUTRAMAONIN		
R/W-					R/W-7h		
7	6	5	4	3	2	1	0
NU2	OUTRAMAGOODIN			NU1	OUTRAMISO		
R-	R/W-7h			R-	R/W-0h		

Table 167. HWACCREG14 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	NU3	R/W		
10-8	OUTRAMAONIN	R/W	7h	Control for Array for the two Output RAMS
7	NU2	R		
6-4	OUTRAMAGOODIN	R/W	7h	Control for Array for the two Output RAMS
3	NU1	R		
2-0	OUTRAMISO	R/W	0h	Control for isolation for the two Output RAMS

B.1.5.15 HWACCREG15 Register (Offset = 38h) [reset = 0h]

HWACCREG15 is shown in Figure 183 and described in Table 168.

Return to Summary Table.

Figure 183. HWACCREG15 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPARE2																															
R/W-0h																															

Table 168. HWACCREG15 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SPARE2	R/W	0h	

B.1.5.16 HWACCREG16 Register (Offset = 3Ch) [reset = 0h]

HWACCREG16 is shown in Figure 184 and described in Table 169.

Return to Summary Table.

Figure 184. HWACCREG16 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPARE3																															
R/W-0h																															

Table 169. HWACCREG16 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SPARE3	R/W	0h	

B.1.5.17 MAX1VALUE Register (Offset = 40h) [reset = 0h]

MAX1VALUE is shown in Figure 185 and described in Table 170.

Return to Summary Table.

Figure 185. MAX1VALUE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU								MAX1VALUE																							
R-								R-																							

Table 170. MAX1VALUE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	NU	R		
23-0	MAX1VALUE	R		

B.1.5.18 MAX1INDEX Register (Offset = 44h) [reset = 0h]

MAX1INDEX is shown in Figure 186 and described in Table 171.

Return to Summary Table.

Figure 186. MAX1INDEX Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU																				MAX1INDEX											
R-																				R-											

Table 171. MAX1INDEX Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	NU	R		
11-0	MAX1INDEX	R		

B.1.5.19 ISUM1LSB Register (Offset = 48h) [reset = 0h]

ISUM1LSB is shown in Figure 187 and described in Table 172.

Return to Summary Table.

Figure 187. ISUM1LSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISUM1LSB																															
R-																															

Table 172. ISUM1LSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ISUM1LSB	R	0	

B.1.5.20 ISUM1MSB Register (Offset = 4Ch) [reset = 0h]

ISUM1MSB is shown in Figure 188 and described in Table 173.

Return to Summary Table.

Figure 188. ISUM1MSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NU															
R-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU												ISUM1MSB			
R-												R-			

Table 173. ISUM1MSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	NU	R		
3-0	ISUM1MSB	R		

B.1.5.21 QSUM1LSB Register (Offset = 50h) [reset = 0h]

QSUM1LSB is shown in Figure 189 and described in Table 174.

Return to Summary Table.

Figure 189. QSUM1LSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSUM1LSB																															
R-																															

Table 174. QSUM1LSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QSUM1LSB	R	0	

B.1.5.22 QSUM1MSB Register (Offset = 54h) [reset = 0h]

QSUM1MSB is shown in Figure 190 and described in Table 175.

Return to Summary Table.

Figure 190. QSUM1MSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NU															
R-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU												QSUM1MSB			
R-												R-			

Table 175. QSUM1MSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	NU	R		
3-0	QSUM1MSB	R		

B.1.5.23 MAX2VALUE Register (Offset = 58h) [reset = 0h]

MAX2VALUE is shown in Figure 191 and described in Table 176.

Return to Summary Table.

Figure 191. MAX2VALUE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU								MAX2VALUE																							
R-								R-																							

Table 176. MAX2VALUE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	NU	R		
23-0	MAX2VALUE	R		

B.1.5.24 MAX2INDEX Register (Offset = 5Ch) [reset = 0h]

MAX2INDEX is shown in Figure 192 and described in Table 177.

Return to Summary Table.

Figure 192. MAX2INDEX Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU																				MAX2INDEX											
R-																				R-											

Table 177. MAX2INDEX Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	NU	R		
11-0	MAX2INDEX	R		

B.1.5.25 ISUM2LSB Register (Offset = 60h) [reset = 0h]

ISUM2LSB is shown in Figure 193 and described in Table 178.

Return to Summary Table.

Figure 193. ISUM2LSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISUM2LSB																															
R-																															

Table 178. ISUM2LSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ISUM2LSB	R	0	

B.1.5.26 ISUM2MSB Register (Offset = 64h) [reset = 0h]

ISUM2MSB is shown in Figure 194 and described in Table 179.

Return to Summary Table.

Figure 194. ISUM2MSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NU															
R-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU												ISUM2MSB			
R-												R-			

Table 179. ISUM2MSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	NU	R		
3-0	ISUM2MSB	R		

B.1.5.27 QSUM2LSB Register (Offset = 68h) [reset = 0h]

QSUM2LSB is shown in Figure 195 and described in Table 180.

Return to Summary Table.

Figure 195. QSUM2LSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSUM2LSB																															
R-																															

Table 180. QSUM2LSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QSUM2LSB	R	0	

B.1.5.28 QSUM2MSB Register (Offset = 6Ch) [reset = 0h]

QSUM2MSB is shown in Figure 196 and described in Table 181.

Return to Summary Table.

Figure 196. QSUM2MSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NU															
R-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU												QSUM2MSB			
R-												R-			

Table 181. QSUM2MSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	NU	R		
3-0	QSUM2MSB	R		

B.1.5.29 MAX3VALUE Register (Offset = 70h) [reset = 0h]

MAX3VALUE is shown in Figure 197 and described in Table 182.

Return to Summary Table.

Figure 197. MAX3VALUE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU								MAX3VALUE																							
R-								R-																							

Table 182. MAX3VALUE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	NU	R		
23-0	MAX3VALUE	R		

B.1.5.30 MAX3INDEX Register (Offset = 74h) [reset = 0h]

MAX3INDEX is shown in Figure 198 and described in Table 183.

Return to Summary Table.

Figure 198. MAX3INDEX Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU																				MAX3INDEX											
R-																				R-											

Table 183. MAX3INDEX Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	NU	R		
11-0	MAX3INDEX	R		

B.1.5.31 ISUM3LSB Register (Offset = 78h) [reset = 0h]

ISUM3LSB is shown in Figure 199 and described in Table 184.

Return to Summary Table.

Figure 199. ISUM3LSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISUM3LSB																															
R-																															

Table 184. ISUM3LSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ISUM3LSB	R	0	

B.1.5.32 ISUM3MSB Register (Offset = 7Ch) [reset = 0h]

ISUM3MSB is shown in Figure 200 and described in Table 185.

Return to Summary Table.

Figure 200. ISUM3MSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NU															
R-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU												ISUM3MSB			
R-												R-			

Table 185. ISUM3MSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	NU	R		
3-0	ISUM3MSB	R		

B.1.5.33 QSUM3LSB Register (Offset = 80h) [reset = 0h]

QSUM3LSB is shown in Figure 201 and described in Table 186.

Return to Summary Table.

Figure 201. QSUM3LSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSUM3LSB																															
R-																															

Table 186. QSUM3LSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QSUM3LSB	R	0	

B.1.5.34 QSUM3MSB Register (Offset = 84h) [reset = 0h]

QSUM3MSB is shown in Figure 202 and described in Table 187.

Return to Summary Table.

Figure 202. QSUM3MSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NU															
R-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU												QSUM3MSB			
R-												R-			

Table 187. QSUM3MSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	NU	R		
3-0	QSUM3MSB	R		

B.1.5.35 MAX4VALUE Register (Offset = 88h) [reset = 0h]

MAX4VALUE is shown in Figure 203 and described in Table 188.

Return to Summary Table.

Figure 203. MAX4VALUE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU								MAX4VALUE																							
R-								R-																							

Table 188. MAX4VALUE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	NU	R		
23-0	MAX4VALUE	R		

B.1.5.36 MAX4INDEX Register (Offset = 8Ch) [reset = 0h]

MAX4INDEX is shown in Figure 204 and described in Table 189.

Return to Summary Table.

Figure 204. MAX4INDEX Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU																				MAX4INDEX											
R-																				R-											

Table 189. MAX4INDEX Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	NU	R		
11-0	MAX4INDEX	R		

B.1.5.37 ISUM4LSB Register (Offset = 90h) [reset = 0h]

ISUM4LSB is shown in Figure 205 and described in Table 190.

Return to Summary Table.

Figure 205. ISUM4LSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISUM4LSB																															
R-																															

Table 190. ISUM4LSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ISUM4LSB	R	0	

B.1.5.38 ISUM4MSB Register (Offset = 94h) [reset = 0h]

ISUM4MSB is shown in Figure 206 and described in Table 191.

Return to Summary Table.

Figure 206. ISUM4MSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NU															
R-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU												ISUM4MSB			
R-												R-			

Table 191. ISUM4MSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	NU	R		
3-0	ISUM4MSB	R		

B.1.5.39 QSUM4LSB Register (Offset = 98h) [reset = 0h]

QSUM4LSB is shown in Figure 207 and described in Table 192.

Return to Summary Table.

Figure 207. QSUM4LSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSUM4LSB																															
R-																															

Table 192. QSUM4LSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QSUM4LSB	R	0	

B.1.5.40 QSUM4MSB Register (Offset = 9Ch) [reset = 0h]

QSUM4MSB is shown in Figure 208 and described in Table 193.

Return to Summary Table.

Figure 208. QSUM4MSB Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NU															
R-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU												QSUM4MSB			
R-												R-			

Table 193. QSUM4MSB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	NU	R		
3-0	QSUM4MSB	R		

B.1.5.41 DCOFFSETI Register (Offset = A0h) [reset = 0h]

DCOFFSETI is shown in Figure 209 and described in Table 194.

Return to Summary Table.

Figure 209. DCOFFSETI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU								DCOFFSETI																							
R-								R/W-0h																							

Table 194. DCOFFSETI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	NU	R		
23-0	DCOFFSETI	R/W	0h	

B.1.5.42 DCOFFSETQ Register (Offset = A4h) [reset = 0h]

DCOFFSETQ is shown in Figure 210 and described in Table 195.

Return to Summary Table.

Figure 210. DCOFFSETQ Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU								DCOFFSETQ																							
R-								R/W-0h																							

Table 195. DCOFFSETQ Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	NU	R		
23-0	DCOFFSETQ	R/W	0h	

B.1.5.43 CFARTEST Register (Offset = A8h) [reset = 0h]

CFARTEST is shown in Figure 211 and described in Table 196.

Return to Summary Table.

Figure 211. CFARTEST Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU								CFARTEST																							
R-								R/W-0h																							

Table 196. CFARTEST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	NU	R		
23-0	CFARTEST	R/W	0h	

B.1.5.44 RDSTATUS Register (Offset = ACh) [reset = 0h]

RDSTATUS is shown in Figure 212 and described in Table 197.

Return to Summary Table.

Figure 212. RDSTATUS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NU															
R-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOOPCNT												PARAMADDR			
R-												R-			

Table 197. RDSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	NU	R		
15-4	LOOPCNT	R		
3-0	PARAMADDR	R		

B.1.5.45 SIGDMACH1DONE Register (Offset = B0h) [reset = 0h]

SIGDMACH1DONE is shown in Figure 213 and described in Table 198.

Return to Summary Table.

Figure 213. SIGDMACH1DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH1DONE																															
R-																															

Table 198. SIGDMACH1DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH1DONE	R	0	Signature for DMA channel 1 completion (tied to “0x0001”in HW)

B.1.5.46 SIGDMACH2DONE Register (Offset = B4h) [reset = 0h]

SIGDMACH2DONE is shown in Figure 214 and described in Table 199.

Return to Summary Table.

Figure 214. SIGDMACH2DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH2DONE																															
R-																															

Table 199. SIGDMACH2DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH2DONE	R	0	Signature for DMA channel 2 completion (tied to “0x0002” in HW)

B.1.5.47 SIGDMACH3DONE Register (Offset = B8h) [reset = 0h]

SIGDMACH3DONE is shown in Figure 215 and described in Table 200.

Return to Summary Table.

Figure 215. SIGDMACH3DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH3DONE																															
R-																															

Table 200. SIGDMACH3DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH3DONE	R	0	Signature for DMA channel 3 completion (tied to “0x0004”in HW)

B.1.5.48 SIGDMACH4DONE Register (Offset = BCh) [reset = 0h]

SIGDMACH4DONE is shown in Figure 216 and described in Table 201.

Return to Summary Table.

Figure 216. SIGDMACH4DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH4DONE																															
R-																															

Table 201. SIGDMACH4DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH4DONE	R	0	Signature for DMA channel 4 completion (tied to “0x0008” in HW)

B.1.5.49 SIGDMACH5DONE Register (Offset = C0h) [reset = 0h]

SIGDMACH5DONE is shown in Figure 217 and described in Table 202.

Return to Summary Table.

Figure 217. SIGDMACH5DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH5DONE																															
R-																															

Table 202. SIGDMACH5DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH5DONE	R	0	Signature for DMA channel 5 completion (tied to “0x0010” in HW)

B.1.5.50 SIGDMACH6DONE Register (Offset = C4h) [reset = 0h]

SIGDMACH6DONE is shown in Figure 218 and described in Table 203.

Return to Summary Table.

Figure 218. SIGDMACH6DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH6DONE																															
R-																															

Table 203. SIGDMACH6DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH6DONE	R	0	Signature for DMA channel 6 completion (tied to “0x0020” in HW)

B.1.5.51 SIGDMACH7DONE Register (Offset = C8h) [reset = 0h]

SIGDMACH7DONE is shown in Figure 219 and described in Table 204.

Return to Summary Table.

Figure 219. SIGDMACH7DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH7DONE																															
R-																															

Table 204. SIGDMACH7DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH7DONE	R	0	Signature for DMA channel 7 completion (tied to “0x0040” in HW)

B.1.5.52 SIGDMACH8DONE Register (Offset = CCh) [reset = 0h]

SIGDMACH8DONE is shown in Figure 220 and described in Table 205.

Return to Summary Table.

Figure 220. SIGDMACH8DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH8DONE																															
R-																															

Table 205. SIGDMACH8DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH8DONE	R	0	Signature for DMA channel 8 completion (tied to “0x0080” in HW)

B.1.5.53 SIGDMACH9DONE Register (Offset = D0h) [reset = 0h]

SIGDMACH9DONE is shown in Figure 221 and described in Table 206.

Return to Summary Table.

Figure 221. SIGDMACH9DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH9DONE																															
R-																															

Table 206. SIGDMACH9DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH9DONE	R	0	Signature for DMA channel 9 completion (tied to “0x0100” in HW)

B.1.5.54 SIGDMACH10DONE Register (Offset = D4h) [reset = 0h]

SIGDMACH10DONE is shown in Figure 222 and described in Table 207.

Return to Summary Table.

Figure 222. SIGDMACH10DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH10DONE																															
R-																															

Table 207. SIGDMACH10DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH10DONE	R	0	Signature for DMA channel 10 completion (tied to “0x0200” in HW)

B.1.5.55 SIGDMACH11DONE Register (Offset = D8h) [reset = 0h]

SIGDMACH11DONE is shown in Figure 223 and described in Table 208.

Return to Summary Table.

Figure 223. SIGDMACH11DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH11DONE																															
R-																															

Table 208. SIGDMACH11DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH11DONE	R	0	Signature for DMA channel 11 completion (tied to “0x0040” in HW)

B.1.5.56 SIGDMACH12DONE Register (Offset = DCh) [reset = 0h]

SIGDMACH12DONE is shown in Figure 224 and described in Table 209.

Return to Summary Table.

Figure 224. SIGDMACH12DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH12DONE																															
R-																															

Table 209. SIGDMACH12DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH12DONE	R	0	Signature for DMA channel 12 completion (tied to “0x0080” in HW)

B.1.5.57 SIGDMACH13DONE Register (Offset = E0h) [reset = 0h]

SIGDMACH13DONE is shown in Figure 225 and described in Table 210.

Return to Summary Table.

Figure 225. SIGDMACH13DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH13DONE																															
R-																															

Table 210. SIGDMACH13DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH13DONE	R	0	Signature for DMA channel 13 completion (tied to “0x1000” in HW)

B.1.5.58 SIGDMACH14DONE Register (Offset = E4h) [reset = 0h]

SIGDMACH14DONE is shown in Figure 226 and described in Table 211.

Return to Summary Table.

Figure 226. SIGDMACH14DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH14DONE																															
R-																															

Table 211. SIGDMACH14DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH14DONE	R	0	Signature for DMA channel 14 completion (tied to “0x2000” in HW)

B.1.5.59 SIGDMACH15DONE Register (Offset = E8h) [reset = 0h]

SIGDMACH15DONE is shown in Figure 227 and described in Table 212.

Return to Summary Table.

Figure 227. SIGDMACH15DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH15DONE																															
R-																															

Table 212. SIGDMACH15DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH15DONE	R	0	Signature for DMA channel 15 completion (tied to “0x4000” in HW)

B.1.5.60 SIGDMACH16DONE Register (Offset = ECh) [reset = 0h]

SIGDMACH16DONE is shown in Figure 228 and described in Table 213.

Return to Summary Table.

Figure 228. SIGDMACH16DONE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGDMACH16DONE																															
R-																															

Table 213. SIGDMACH16DONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGDMACH16DONE	R	0	Signature for DMA channel 16 completion (tied to “0x8000” in HW)

B.1.5.61 MEMACCESSERR Register (Offset = F0h) [reset = 0h]

MEMACCESSERR is shown in Figure 229 and described in Table 214.

Return to Summary Table.

Figure 229. MEMACCESSERR Register

31	30	29	28	27	26	25	24
NU3							
R-							
23	22	21	20	19	18	17	16
NU3				STATERRCODE			
R-				R-			
15	14	13	12	11	10	9	8
NU2				ERRCODEMASK			
R-				R/W-0h			
7	6	5	4	3	2	1	0
NU1				ERRCODECLR			
R-				0h			

Table 214. MEMACCESSERR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	NU3	R		
19-16	STATERRCODE	R		Error status : {iping_access_err,ipong_access_err,oping_access_err,opong_access_err}
15-12	NU2	R		
11-8	ERRCODEMASK	R/W	0h	Mask for STATERRCODE (Masked STATERRCODE will not generate interrupt)
7-4	NU1	R		
3-0	ERRCODECLR		0h	Clear for STATERRCODE.Self clearing

B.1.5.62 FFTCLIP Register (Offset = F4h) [reset = 0h]

FFTCLIP is shown in Figure 230 and described in Table 215.

Return to Summary Table.

Figure 230. FFTCLIP Register

31	30	29	28	27	26	25	24
NU2							
R-							
23	22	21	20	19	18	17	16
NU2							CLRFFTCLIPSTAT
R-							0h
15	14	13	12	11	10	9	8
NU1						FFTCLCIPSTAT	
R-						R-	
7	6	5	4	3	2	1	0
FFTCLCIPSTAT							
R-							

Table 215. FFTCLIP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	NU2	R		
16	CLRFFTCLIPSTAT		0h	
15-10	NU1	R		
9-0	FFTCLCIPSTAT	R		

B.1.5.63 FFTPEAKCNT Register (Offset = F8h) [reset = 0h]

FFTPEAKCNT is shown in Figure 231 and described in Table 216.

Return to Summary Table.

Figure 231. FFTPEAKCNT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU																				FFTPEAKCNT											
R-																				R-											

Table 216. FFTPEAKCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	NU	R		
11-0	FFTPEAKCNT	R		

B.1.5.64 HWACCREG1RD Register (Offset = FCh) [reset = 0h]

HWACCREG1RD is shown in Figure 232 and described in Table 217.

Return to Summary Table.

Figure 232. HWACCREG1RD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HWACCREG1RD																															
R-																															

Table 217. HWACCREG1RD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HWACCREG1RD	R	0	

B.1.5.65 HWACCREG2RD Register (Offset = 100h) [reset = 0h]

HWACCREG2RD is shown in Figure 233 and described in Table 218.

Return to Summary Table.

Figure 233. HWACCREG2RD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HWACCREG2RD																															
R-																															

Table 218. HWACCREG2RD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HWACCREG2RD	R	0	

B.1.5.66 HWACCREG3RD Register (Offset = 104h) [reset = 0h]

HWACCREG3RD is shown in Figure 234 and described in Table 219.

Return to Summary Table.

Figure 234. HWACCREG3RD Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HWACCREG3RD																															
R-																															

Table 219. HWACCREG3RD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HWACCREG3RD	R	0	

C.1 Safety Feature

This section describes how the safety features are implemented. Not all HWA versions has safety features, please refer to Table 1 for feature list of HWA verions. For detailed register definition, please check Appendix B.

C.1.1 State Machine Lockstep

The state machine of the Radar Hardware Accelerator has a lockstep diagnostic implemented for safety. This feature includes the addition of a diagnostic state machine that runs the same set of operations as the main state machine at the same time in parallel Both the main state machine and the diagnostic state machine are fed the same input signals and will generate the same output. A comparator module compares the outputs of the two state machines and flags any mis-compares to the DSPSS ESM. The state machine lockstep registers are listed in Table 220.

The lockstep checking is disabled after reset and has to be enabled by the CPU. Once enabled, the lockstep diagnostic is continually operating every cock cycle.

In case of a mis-compare the following ESM error is triggered inside DSPSS:

- esm_error_gp1[37] = HWA_FSM_LOCKSTEP_ERR

Table 220. Lockstep Register

Register[bit]	Register Field	Width	Description
ECCENABLE[16]	FSM_LOCKSTEP_EN	1	'1': Enable Lockstep for Accelerator FSM
ECCENABLE[17]	FSM_LOCKSTEP_ SELFTEST_EN	1	'1': Enable Selftest for Accelerator FSM. Setting this bit will force an error at the state machine output to ensure that the compare logic is working properly. The error forcing capability allows you to test the system level response to a lockstep compare error. Note : This filed must be set to '0' during the normal operation of the Hardware Accelerator

C.1.2 ECC for Accelerator Local Memories

All the four local RAMs of the Hardware Accelerator are protected by a 6-bit SECDED ECC computed over 16-bit-wide data. In case of an ECC error in any of the 4 local RAMs, the following ESM errors are triggered inside DSPSS to indicate a double bit error and a single bit error respectively:

- esm_error_gp1[43] = HWA_IO_RAM_FATAL_ERR
- esm_error_gp1[42] = HWA_IO_RAM_REPAIR_ERR

The ECC checking is disabled after reset and has to be enabled by the CPU using the registers listed in Table 221. Before the ECC is enabled, the memory has to be initialized with all ‘0’s to ensure that ECC codewords are also valid. The CPU is expected to write the “INIT” registers and wait for “INITDONE” status to be set to ‘1’ before using these RAMs for functional use.

Table 221. ECC Control Registers for HWA Local Memories

Register[bit]	Register Field	Width	Description
ECCENABLE[12]	IPING_ECC_EN	1	‘1’: Enable ECC for ACCEL_MEM0
ECCENABLE[13]	IPONG_ECC_EN	1	‘1’: Enable ECC for ACCEL_MEM1
ECCENABLE[14]	OPING_ECC_EN	1	‘1’: Enable ECC for ACCEL_MEM2
ECCENABLE[15]	OPONG_ECC_EN	1	‘1’: Enable ECC for ACCEL_MEM3
MEMINIT[12]	IPING_INIT	1	'1': Start initialing ACCEL_MEM0 with all '0's
MEMINIT[13]	IPONG_INIT	1	'1': Start initialing ACCEL_MEM1 with all '0's
MEMINIT[14]	OPING_INIT	1	'1': Start initialing ACCEL_MEM2 with all '0's
MEMINIT[15]	OPONG_INIT	1	'1': Start initialing ACCEL_MEM3 with all '0's
MEMINIT[12]	IPING_INITDONE	1	'1': Init done status for ACCEL_MEM0
MEMINIT[13]	IPONG_INITDONE	1	'1': Init done status for ACCEL_MEM1
MEMINIT[14]	OPING_INITDONE	1	'1': Init done status for ACCEL_MEM2
MEMINIT[15]	OPONG_INITDONE	1	'1': Init done status for ACCEL_MEM3

For further debug, the status registers isted in Table 222 are also available to identify the RAM, RAM address location and RAM bit error locations that caused the ECC error to ESM. All the status registers can be cleared by writing ‘1’ into “ECCCLR” registers.

Table 222. ECC Status Registers for HWA Local Memories

Register[bit]	Register Field	Width	Description
ECC_SBE_STATUS[12]	IPING_ECC_SBE	1	'1': Single bit error status for ACCEL_MEM0
ECC_SBE_STATUS[13]	IPONG_ECC_SBE	1	'1': Single bit error status for ACCEL_MEM1
ECC_SBE_STATUS[14]	OPING_ECC_SBE	1	'1': Single bit error status for ACCEL_MEM2
ECC_SBE_STATUS[15]	OPONG_ECC_SBE	1	'1': Single bit error status for ACCEL_MEM3
ECC_DBE_STATUS[12]	IPING_ECC_DBE	1	'1': Double bit error status for ACCEL_MEM0
ECC_DBE_STATUS[13]	IPONG_ECC_DBE	1	'1': Double bit error status for ACCEL_MEM1
ECC_DBE_STATUS[14]	OPING_ECC_DBE	1	'1': Double bit error status for ACCEL_MEM2
ECC_DBE_STATUS[15]	OPONG_ECC_DBE	1	'1': Double bit error status for ACCEL_MEM3
IPINGERRLOC[15:0]	IPING_ERR_ADDR	16	Address of error location – rows 0-1023

Table 222. ECC Status Registers for HWA Local Memories (continued)

Register[bit]	Register Field	Width	Description
IPINGERRLOC[31:16]	IPING_ERR_BIT_REGION	16	[16]->1 , SBE is located in region0 (column [15: 0]) [17]->1 , SBE is located in region1 (column [31:16]) [18]->1 , SBE is located in region2 (column [47:32]) [19]->1 , SBE is located in region3 (column [63:48]) [20]->1 , SBE is located in region4 (column [79:63]) [21]->1 , SBE is located in region5 (column [95:80]) [22]->1 , SBE is located in region6 (column [111:96]) [23]->1 , SBE is located in region7 (column [127:112]) (Note each region is 16 bits as ECC is implemented every 16 bits)
IPINGSBELOC[31:0]	IPING_ERR_BIT_LOC	32	[3:0]->SBE bit location in region0 [7:4]->SBE bit location in region1 [11:8]->SBE bit location in region2 [15:12]->SBE bit location in region3 [19:16]->SBE bit location in region4 [23:20]->SBE bit location in region5 [27:24]->SBE bit location in region6 [31:28]->SBE bit location in region7
IPONGERRLOC[15:0]	IPONG_ERR_ADDR	16	Refer IPING_ERR_BIT_REGION
IPONGERRLOC[31:16]	IPONG_ERR_BIT_REGION	16	Refer IPING_ERR_BIT_LOC
IPONGSBELOC[31:0]	IPONG_ERR_BIT_LOC	32	Refer IPING_ERR_BIT_REGION
OPINGERRLOC[15:0]	OPING_ERR_ADDR	16	Refer IPING_ERR_BIT_LOC
OPINGERRLOC[31:16]	OPING_ERR_BIT_REGION	16	Refer IPING_ERR_BIT_REGION
OPINGSBELOC[31:0]	OPING_ERR_BIT_LOC	32	Refer IPING_ERR_BIT_LOC
OPONGERRLOC[15:0]	OPONG_ERR_ADDR	16	Refer IPING_ERR_BIT_REGION
OPONGERRLOC[31:16]	OPONG_ERR_BIT_REGION	16	Refer IPING_ERR_BIT_LOC
OPONGSBELOC[31:0]	OPONG_ERR_BIT_LOC	32	Refer IPING_ERR_BIT_REGION
ECCERRCLR[12]	IPING_ECC_ERRCLR	1	'1': Clear Error registers(status,address and bit location) for ACCEL_MEM0
ECCERRCLR[13]	IPONG_ECC_ERRCLR	1	'1': Clear Error registers(status,address and bit location) for ACCEL_MEM1
ECCERRCLR[14]	OPING_ECC_ERRCLR	1	'1': Clear Error registers(status,address and bit location) for ACCEL_MEM2
ECCERRCLR[15]	OPONG_ECC_ERRCLR	1	'1': Clear Error registers(status,address and bit location) for ACCEL_MEM3

C.1.3 ECC for Window RAM

The memory for storing windowing function is protected by a 7-bit SECDED ECC over 18-bit-wide data. In case of an ECC error, the following ESM errors are triggered inside DSPSS to indicate a double bit error and a single bit error respectively:

- esm_error_gp1[45] = HWA_WIN_RAM_REPAIR_ERR
- esm_error_gp1[44] = HWA_WIN_RAM_FATAL_ERR

The ECC checking is disabled after reset and has to be enabled by the CPU using the registers listed in Table 223. Before the ECC is enabled, the memory has to be initialized with all ‘0’s to ensure that ECC codewords are also valid. The CPU is expected to write the “INIT” registers and wait for “INITDONE” status to be set to ‘1’ before using these RAMs for functional use . The related registers are described in Table 223.

Table 223. ECC Control Registers for Window RAM

Register	Register Field	Width	Description
ECCENABLE[0]	WIN_RAM_ECC_EN	1	‘1’: Enable ECC for Window RAM
MEMINIT[0]	WIN_RAM_INIT	1	'1': Start initialing Window RAM with all '0's

Table 223. ECC Control Registers for Window RAM (continued)

Register	Register Field	Width	Description
MEMINITDONE[0]	WIN_RAM_INITDONE	1	'1': Init done status for Window RAM

For further debug, status registers as shown in Table 224 are also available to identify the RAM, RAM address location and RAM bit error locations that caused the ECC error to ESM. All the status registers can be cleared by writing ‘1’ into “ECCCLR” registers

Table 224. ECC Status Registers for Window RAM

Register	Register Field	Width	Description
WINRAMERRLOC[15:0]	WIN_RAM_ERR_ADDR	16	Address of SBE/DBE location within Window RAM
WINRAMERRLOC[31:16]	WIN_RAM_ERR_BIT	16	Single bit error location within error address of Window RAM
ECCERRCLR[0]	WIN_RAM_ECC_ERRCLR	1	'1': Clear Error registers(address and bit location) for Window RAM

C.1.4 ECC for Parameter RAM

The memory for storing parameters is protected by a 8-bit SECDED ECC over 64-bit-wide data. In case of an ECC error, the following ESM errors are triggered inside DSPSS to indicate a double bit error and a single bit error, respectively:

- esm_error_gp1[47] = HWA_PARAM_RAM_FATAL_ERR
- esm_error_gp1[46] = HWA_PARAM_RAM_REPAIR_ERR

The ECC checking is disabled after reset and has to be enabled by the CPU using the registers listed in Table 225. Before the ECC is enabled, the memory has to be initialized with all ‘0’s to ensure that ECC codewords are also valid. The CPU is expected to write the “INIT” registers and wait for “INITDONE” status to be set to ‘1’ before using these RAMs for functional use. See Table 225.

Table 225. ECC Control Registers for Parameter RAM

Register	Register Field	Width	Description
ECCENABLE[1]	PARAM_ECC_EN	1	‘1’: Enable ECC for Parameter set RAM
MEMINITDONE[1]	PARAM_INIT	1	'1': Start initialing Parameter set RAM with all '0's
MEMINITDONE[1]	PARAM_INITDONE	1	'1': Init done status for Parameter set RAM

For further debug, status registers listed in Table 226 are also available to identify the RAM, RAM address location and RAM bit error locations that caused the ECC error to ESM. All the status registers can be cleared by writing ‘1’ into “ECCCLR” registers.

Table 226. ECC Status Registers for Parameter RAM

Register	Register Field	Width	Description
PARAMRAMERRLOC [15:0]	PARAM_ERR_ADDR	16	[Debug] Address of SBE/DBE location within Parameter set RAM
PARAMRAMERRLOC [31:16]	PARAM_ERR_BIT	16	[Debug]Single bit error location within error address of Parameter set RAM
ECCERRCLR[1]	PARAM_ECC_ERRCLR	1	'1': Clear Error registers(address and bit location) for Parameter set RAM

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from A Revision (August 2018) to B Revision	Page
• Updates were made in Section 1.1.2.	12
• Updates were made in Appendix A.	80
• Updates were made in Appendix B.	82

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265