Texas Instruments

TMS320DM320 CPU and Peripherals Vol - 2

Technical Reference Manual Version 1.0

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components. In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

Document Revision History

Revision	Date	Notes
0.1	15 Dec, 2003	Draft version
0.2	5 Jan, 2004	Updated according to review comments
0.3	30 Jan, 2004	Updated after reviews
1.0	12 Mar, 2004	First version

Table Of Contents

TABLE OF CONTENTS 4

1 DSP SUBSYSTEM 6

1.1 INTRODUCTION 6
1.2 DSP CORE 6
1.3 DSP SUBSYSTEM MEMORY 7
1.4 DSP INTERNAL MEMORY 9
1.5 DSP PERIPHERALS 13
1.6 DSP - POWER SAVING MODES 17

2 ARM-DSP COMMUNICATION 19

2.1 INTRODUCTION 19
2.2 HPI-16 MODE 20
2.3 MEMORY ACCESS 21
2.4 INTERRUPTS BETWEEN ARM AND DSP 21
2.5 DSP BOOT SEQUENCE 22
2.6 ARM-SIDE HPI CONTROL AND STATUS REGISTERS 23
2.7 DSP SUBSYSTEM - HPIB DMAC 25
2.8 DSP CONTROLLER (DSPC) 27
2.9 DSP CONTROLLER REGISTERS (ARM SIDE) 28
2.10 HOST PORT INTERFACE BRIDGE REGISTER MAP (HPIB) 30
2.11 HOST PORT INTERFACE BRIDGE REGISTERS 31

3 COPROCESSOR SUBSYSTEM 38

3.1 INTRODUCTION 38
3.2 COPROCESSOR SUBSYSTEM: PERIPHERAL REGISTER MAP 41
3.3 SHARED MEMORY ACCESS SWITCHING AND MEMORY MAP 42
3.4 COP DMA CONTROLLER 43
3.5 COPROCESSOR SUBSYSTEM CLOCK CONTROL 47
3.6 COPROCESSOR SUBSYSTEM INTERRUPT/STROBE CONTROL 47
3.7 BREAK-POINT FUNCTION 48
3.8 COPROCESSOR DMA (COP DMA) REGISTERS 49
3.9 COP INTERRUPT / CLOCK CONTROL REGISTER MAP (INTCLK) 59
3.10 DSP INTERRUPT / CLOCK CONTROL REGISTERS 60

4 IMAGING EXTENSION ENGINE (IMX) 75

4.1 INTRODUCTION 75
4.2 IMX CONTROL REGISTER MAP 80

5 VARIABLE LENGTH CODER/DECODER (VLCD) 90

5.2 VARIABLE LENGTH CODER/DECODER (VLCD) MODES 92
5.3 IMAGE BUFFER PORT SELECTION 93
5.4 DC PREDICTOR 94
5.5 VLC IN DMA MODE 95
5.6 ENDIANNESS 95
5.7 VLC/VLD START AND END ADDRESS 95
5.8 VLC/VLD SCAN MODES 96
5.9 MPEG CODED BLOCK PATTERN 96

5.10 LUMA_VECTOR 96
5.11 HUFFMAN TABLE 97
5.12 MPEG MAX LEVEL TABLE 97
5.13 CONTROL LOOKUP TABLE 97
5.14 SYMBOL LOOKUP TABLE 97
5.15 VARIABLE LENGTH CODER DECODER REGISTER MAP (VLCD) 99
5.16 VLCD REGISTERS 101
5.17 VLC – HUFFMAN TABLE CONFIGURATION 159
5.18 VLC – HEADER INSERTION 161
5.19 VLD – HUFFMAN TABLE CONFIGURATION 162
5.20 QIQ – SCANNING PATHS AND QUANTIZATION MATRICES USED 165
5.21 VLC TABLES USED 167
5.22 VLD TABLES USED 169

6 SEQUENCER 170

6.1 INTRODUCTION 170
6.2 ARCHITECTURAL DETAILS 170
6.3 PROGRAMMING MODEL 172
6.4 ASSEMBLY MNEMONIC 175
6.5 SYNC INPUT FOR THE SEQUENCER 187
6.6 INTERRUPT/STROBE 188
6.7 BREAK-POINT FUNCTION 188
6.8 ASSEMBLY LANGUAGE TOOL FLOW 189
6.9 ASSEMBLER 190
6.10 SEQUENCER REGISTER MAP (SEQ) 195
6.11 SEQUENCER REGISTERS 196

7 DCT/IDCT 204

7.1 INTRODUCTION 204
7.2 FEATURES OF DM320 DCT 204
7.3 LUMA/CHROMA FORMATS 205
7.4 INTERRUPT 206
7.5 PRECISION 206
7.6 DCT REGISTER MAP 207
7.7 DCT REGISTERS 208

8 APPENDIX B 214

8.1 INTRODUCTION 214
8.2 PROGRAMMING / PERFORMANCE EXAMPLE 215

1 DSP Subsystem

1.1 Introduction

The DSP subsystem is based on the TMS320C5409 DSP from TI catalog. This DSP core and its peripherals are fully code-compatible with other C54x products. The DSP software development tools for the C54x are very mature and can be used to develop, compile and debug the DSP code.

The DSP subsystem includes the C54x core, several blocks of on chip memories and DSP peripherals like McBSP, DMAC, Timer and HPI, Interrupt controller.

This chapter gives only an overview of the DSP core and focuses more on its peripherals. For detailed information on the architecture of this 16-bit, DSP core and/or on the associated tools, refer to the C54x user's guides and tools documentation (http://focus.ti.com/lit/ug/spru131g/spru131g.pdf and http://focus.ti.com/lit/ds/sprs082c/sprs082c.pdf).

In DM320, DSP has access to Coprocessor subsystem control registers and various shared memory blocks, and can control COP DMA. Also it can control HPIB DMA for data transfer between SDRAM/EMIF and DSP internal memory.

1.2 DSP core

C54x devices are fixed-point digital signal processors from the Texas Instruments TMS320 family. The C54x CPU, with its advanced modified Harvard architecture, features minimized power consumption and a high degree of parallelism.

This processor has one program memory bus and three data memory buses. It also provides an arithmetic logic unit (ALU) that has a high degree of parallelism, on-chip memory, and additional on-chip peripherals. The basis of the operational flexibility and speed of this DSP is a highly specialized instruction set.

Separate program and data spaces allow simultaneous access to program instructions and data, providing the high degree of parallelism. Two read operations and one write operation can be performed in a single cycle. In addition, data can be transferred between data and program spaces. Such parallelism supports a powerful set of arithmetic, logic, and bit manipulation operations that can all be performed in a single machine cycle. This processor includes the control mechanisms to manage interrupts, repeated operations, and function calls.

1.2.1 Features

- Low-power C54x DSP CPU
- Software-programmable wait-state generator with bank-switching wait state logic
- Program space
- Data space
- I/O space
- Scan-based emulation logic

In the DM320, the wait state generator and the external memory interface are used to connect the DSP core to the Coprocessor subsystem.

The maximum DSP cycle frequency is programmable up to 100 MHz (refer to the DM320 data sheet for the exact value).

1.3 DSP subsystem memory

DSP memory is constructed from three independent spaces: program, data, and I/O. The DSP has 64 KW of program and data memory and 32 KW of ROM. Shared memories (Image Buffer A/B/C, iMX memories, VLCD memories and Sequencer memory) are connected to the DSP's expansion memory bus (XIO 16-bit data bus) and can be accessed directly through DSP's data space. Control and configuration registers for the Image Buffer, iMX, DCT, VLCD, Sequencer, Coprocessor clock/interrupt controller and HPIB DMAC are mapped to DSP's I/O space.

1.3.1 Program, Data and IO spaces

The DSP memory is organized into three individually selectable spaces: program, data, and I/O. The program memory space contains the instructions to execute, as well as tables used in execution. The data-memory space stores data used by instructions. The I/O memory space interfaces to the Image Buffer, iMX, VLCD, DCT, Sequencer, Coprocessor Clock and Interrupt Controllers and HPIB DMAC registers.

There are three CPU status register bits that affect memory configuration. The MP/MC, OVLY, and DROM bits are located in the processor mode status register (PMST).

1.3.2 DSP Memory map

The DSP has 32 K words (1 word = 16 bits) of internal DARAM memory, 16 K words of DSP internal data SARAM, 16 K words of Page1 program SARAM and 32 K words of ROM.

Figure 1 shows the DSP memory map after reset. Each block of memory has a width of 16 bits. Some blocks are available for both program and data spaces, such as the DARAM block when OVLY=1.

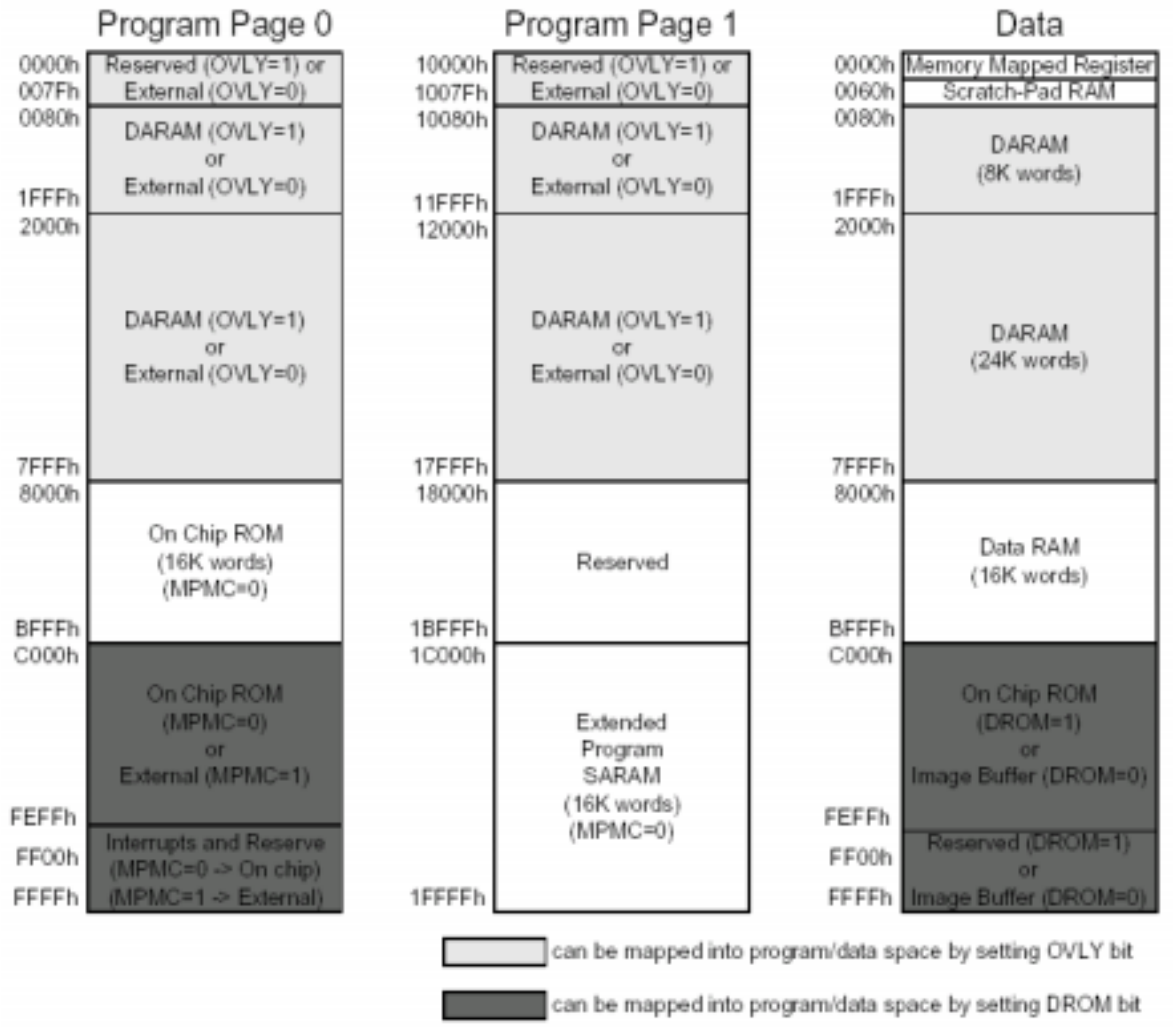


Figure 1: DSP Memory Map

1.3.3 I/O memory space

The C54x devices offer an I/O-memory space in addition to the program memory and data-memory spaces. The I/O-memory space is a 64K-word address space (0000h–FFFFh) and exists only external to the DSP device. Two instructions, PORTR and PORTW, are used to access this space. In the DM320, only some addresses from this space are used. Image Buffer registers, iMX registers, VLCD registers, DCT registers, Sequencer registers, Coprocessor Clock/Interrupt registers and HPIB DMAC registers are mapped to I/O spaces as shown in Table 1.

Module	Page	Start Addr	End Addr
Image Buffer Control	IO	0000:0000	0000:007F
iMX Control	IO	0000:0080	0000:00FF
VLCD Control	IO	0000:0100	0000:017F
DCT Control	IO	0000:0180	0000:01FF
Sequencer Control	IO	0000:0200	0000:027F
Interrupt-Clock control	IO	0000:0280	0000:02FF
HPIB	IO	0000:8000	0000:800F

Table 1:I/O space address map

1.4 DSP internal memory

1.4.1 Organization of the memory

The DSP subsystem features 32K x 16 bits of on-chip DARAM as well as 32K x 16 bits of on-chip SARAM. The OVLY bit controls the internal memory configuration. After a reset, this bit is set to ‘0’. The boot loader program on ROM changes this bit to ‘1’. In this mode, the DARAM memory block is shared between data and program space. It is possible to change this bit to ‘0’ by software, however, it is not recommended, since the internal memory will not be available anymore to store program data. The ARM can access the DSP internal memory through the HPI Bridge. This capability is explained in greater detail in the ARM-DSP Communication chapter.

1.4.2 DARAM *properties*

The DARAM consists of 4 blocks of 8K x 16-bit memory blocks. The DSP can perform two accesses to a DARAM in one machine cycle (two reads in one cycle, or a read and a write in one cycle). It can also perform multiple accesses to separate memory blocks in one machine cycle.

Figure 2 shows the RAM block organization for the DM320 DSP DARAM memory. The dotted blue lines in the figure indicate block boundaries. The first block contains the memory-mapped DSP and peripheral registers (0x0000-0x005F) and 32 words of scratch pad memory (0x0060-0x007F).

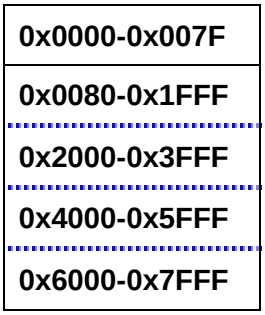


Figure 2 :Internal DSP DARAM block organization

1.4.3 SARAM Properties

DM320 DSP subsystem contains 16KW SARAM for data (0x8000 to 0xBFFF) and 16KW SARAM for program (Page 1: 0x1C000 to 0x1FFFF).

No wait state is required to access SARAM. However when there is conflict during access of memory location in SARAM, wait state is automatically inserted by the DSP.

1.4.4 Access of DSP internal memory from ARM side

The ARM can access DSP internal memory (64 KW) through the DSP HPI port. The HPI operates in 16-bit non-multiplexed mode. The C54x in the DM320 does not support 8-bit HPI modes. Refer to “ARM-DSP Communication” chapter for details

1.4.5 DMA from DSP internal Memory to SDRAM/External Memory

DM320 DSP subsystem supports DMA transfer between DSP internal memory and SDRAM/external memory. Refer to “ARM-DSP Communication” chapter for details

1.4.6 Reset and relocatable interrupt vector table

Synchronization between DSP, Image Buffer, other coprocessor modules (iMX, VLCD and DCT), Sequencer and ARM is possible using interrupts. Each coprocessor module, Coprocessor DMAC and HPIB DMAC can generate an interrupt to the DSP. The ARM also can interrupt the DSP at any time (provided this interrupt is enabled by the DSP). For the standard 5409 interrupt vector details, refer to the datasheet, http://focus.ti.com/lit/ds/sprs082c/sprs082c.pdf. The DSP external interrupts are hooked up to the various coprocessor sub system interrupts. The details of the DM320 specific interrupts connected to DSP external interrupts are shown in the Table 2.

Port name	Module name
INT0	ARM interrupting DSP via HPIB
INT1	Interrupt is generated based on the settings of DSP_SYNC_STATE and DSP_SYNC_MASK register of the coprocessor subsystem or when DSPINT1 bit in CP_INTC is set.
INT2	Interrupt is generated when DSPINT2 bit in CP_INTC register of the coprocessor subsystem is set.
INT3	Interrupt is generated when DSPINT3 bit in CP_INTC register of the coprocessor subsystem is set or on write of any value to BRKPT_TRG
INT4	HPIB DMAC Interrupt on completion of transfer

Table 2: DSP interrupt ports

The reset, interrupt, and trap vectors are addressed in program space. These vectors are soft—meaning that the processor, when taking the trap, loads the program counter (PC) with the trap address and executes the code at the vector location. Four words are reserved at each vector location to accommodate a delayed branch instruction—either two 1-word instructions or one 2-word instruction —which allows branching to the appropriate interrupt service routine with minimal overhead. For more information refer to the datasheet, http://focus.ti.com/lit/ds/sprs082c/sprs082c.pdf.

At device reset, the reset, interrupt, and trap vectors are mapped to address 0xFF80 in ROM space. However, these vectors can be remapped to the beginning of any 128-word page in program space after device reset. This is done by loading the interrupt vector pointer (IPTR) bits in the PMST register with the appropriate 128-word page boundary address. The DSP boot loader

changes the IPTR value for the interrupt vector table to be located at address 0x7F80.

1.4.7 Memory mapped registers and scratch pad RAM

Most of the DSP control registers are located within the address range 0x00 and 0x5F in the data space of the DSP subsystem memory map. MMRs in address range 0x00-0x20 controls the DSP (Table 3) and 0x20-0x5F control the DSP peripherals. The registers that control the iMX, VLCD, DCT, HPIB DMA, Sequencer and image buffer are mapped into the IO space and do not appear in the Table 4.

McBSP and DSP core DMAC have got large number of control registers. To workaround the limited space in the MMR area, a sub-bank scheme is used to control the McBSP and DMA. Refer enhanced peripheral user's guide http://focus.ti.com/lit/ug/spru302/spru302.pdf for more details.

Table 3 and Table 4 give the name and a short description of the different memory mapped registers.

Address (Hex)	Name	Description
0	IMR	Interrupt mask register
1	IFR	Interrupt flag register
2–5	–	Reserved for testing
6	ST0	Status register 0
7	ST1	Status register 1
8	AL	Accumulator A low word (bits 15–0)
9	AH	Accumulator A high word (bits 31–16)
A	AG	Accumulator A guard bits (bits 39–32)
B	BL	Accumulator B low word (bits 15–0)
C	BH	Accumulator B high word (bits 31–16)
D	BG	Accumulator B guard bits (bits 39–32)
E	T	Temporary register
F	TRN	Transition register
10	AR0	Auxiliary register 0
11	AR1	Auxiliary register 1
12	AR2	Auxiliary register 2
13	AR3	Auxiliary register 3
14	AR4	Auxiliary register 4
15	AR5	Auxiliary register 5
16	AR6	Auxiliary register 6
17	AR7	Auxiliary register 7
18	SP	Stack pointer
19	BK	Circular-buffer size register
1A	BRC	Block-repeat counter
1B	RSA	Block-repeat start address
1C	REA	Block-repeat end address
1D	PMST	Processor mode status register
1E	XPC	Program counter extension register
1E–1F	–	Reserved

Table 3: DSP Memory-Mapped control Registers

Register	Address	Description	Type
DRR20	20h	Data receive register 2	McBSP #0
DRR10	21h	Data receive register 1	McBSP #0
DXR20	22h	Data transmit register 2	McBSP #0
DXR10	23h	Data transmit register 1	McBSP #0
TIM	24h	Timer register	Timer
PRD	25h	Timer period counter	Timer
TCR	26h	Timer control register	Timer
–	27h	Reserved	
SWWSR	28h	Software wait-state register	External Bus
BSCR	29h	Bank-switching control register	External Bus
–	2Ah	Reserved	
SWCR	2Bh	Software wait-state control register	External Bus
HPIC	2Ch	HPI control register	HPI
–	2Dh–37h	Reserved	
SPSA0	38h	McBSP0 subbank address register	McBSP #0
SPSD0	39h	McBSP0 subbank data register	McBSP #0
–	3Ah–3Fh	Reserved	
DRR21	40h	Data receive register 2	McBSP #1
DRR11	41h	Data receive register 1	McBSP #1
DXR21	42h	Data transmit register 2	McBSP #1
DXR11	43h	Data transmit register 1	McBSP #1
–	44h–47h	Reserved	
SPSA1	48h	McBSP1 subbank address register	McBSP #1
SPSD1	49h	McBSP1 subbank data register	McBSP #1
–	4Ah–53h	Reserved	
DMPREC	54h	DMA channel priority and enable control register	DMA
DMSA	55h	DMA subbank address register	DMA
DMSDI	56h	DMA subbank data register with autoincrement	DMA
DMSDN	57h	DMA subbank data register	DMA
–	58h–5Fh	Reserved	

Table 4: DSP Peripheral Memory-Mapped Registers

The advantage of using the scratch pad RAM that is located between address 0x60 and 0x7F is that it can be accessed anytime through an immediate memory access regardless of the data page pointer value. This can be done by using some special assembly instruction like STM. Additionally, since writing to control registers can incur latencies, instructions like STM operate early and avoid most pipeline problems.

1.4.8 ROM usage

Please contact TI for ROM details

1.4.9 Access properties

Internal program, internal data, ROM and the IO register can be accessed with 0 wait states.

But for accessing the shared memories on the XIO data bus (0xC000-0xFEFF) 1 wait state is required.

Since the DSP has programmable software WAIT generator, software WAIT can be generated by setting the number of WAITs in the DSP memory mapped register SWWSR (Software Wait State Register).

1.5 DSP peripherals

The DSP CPU core is associated with an HPI interface, an interrupt handler, a parallel XIO interface, two multi-channel buffered serial ports (McBSP), a DMA controller and a JTAG interface.

The JTAG interface is connected to the JTAG interface of the ARM core in a daisy chained manner.

The parallel interface is used to connect the core to the expanded memory, Coprocessor Subsystem and Sequencer. Refer to the chapter on coprocessor subsystem and Sequencer for more details.

The host port interface (HPI) of the DSP is connected to the DSP Controller of the ARM subsystem. In DM320, the HPI interface operates only in the 16-bit mode. The HPI bridge is described in the ARM-DSP Communication chapter.

The DSP core features a DMA controller, which can transfer data between the internal memories (data memory and program memory). Also it has the ability to transfer data between internal memories and McBSPs.

The PLL that can normally be found on a standard C54x device is not present in DM320 system. The DSP PLL is a part of the ARM Subsystem clock controller module and fully controlled by the ARM.

The DSP can communicate directly with external devices to the DM320 through McBSPs or XF signal. The DSP can control the XF signal by writing into the corresponding memory map register (ST1) or by executing SSBX and RSBX instructions to set and reset XF signal respectively. This signal is multiplexed with the ARM GIO9 pin.

1.5.1 Multi-channel buffered serial ports

The DM320 has two high-speed, full-duplex multi-channel buffered serial ports (McBSP0 and McBSP1). McBSP1 has its pins multiplexed with the ARM GIO10 to GIO16. Refer chapter on General Purpose IO for more details. The McBSP ports allow direct interface to other C54x devices, audio codecs etc.

A brief description of the McBSP included in the DM320‘s DSP core is given in this section. For detailed information on the DSP’s McBSP, refer to the C54x peripheral manual http://focus.ti.com/lit/ug/spru131g/spru131g.pdf and enhanced peripheral user’s guide http://focus.ti.com/lit/ug/spru302/spru302.pdf.

1.5.1.1 Capabilities

The McBSPs have the following capabilities:

- Full-duplex communication
- Double-buffered data registers that allow a continuous data stream
- Independent framing and clocking for receive and transmit
- Direct interface to:
  - T1/E1 framers
  - MVIP switching-compatible and ST-BUS-compliant device

- IOM-2-compliant devices
- AC97-compliant devices
- IIS-compliant devices
- Serial peripheral interface (SPI) devices

- Multichannel transmit and receive of up to 128 channels
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices
- A wide selection of data sizes, including 8, 12, 16, 20, 24, and 32 bits
- μ-law and A-law companding
- External shift clock or an internal, programmable-frequency shift clock for data transfer
- Autobuffering capability through the six-channel DMA controller
- Programmable polarity for both frame synchronization and data clocks
- 8-bit data transfers with option of LSB or MSB first

1.5.1.2 External Interface

The McBSPs consist of separate transmit and receive channels that operate independently. The external interface of each McBSP consists of the following pins:

- BCLKX: Transmit reference clock
- BDX: Transmit data
- BFSX: Transmit frame synchronization
- BCLKR: Receive reference clock
- BDR: Receive data
- BFSR: Receive frame synchronization
- CLKS: External clock reference for the programmable clock generator

1.5.1.3 Transmitter/Receiver Pins

On the transmitter, transmit frame synchronization and clocking is indicated by the BFSX and BCLKX pins, respectively. The DSP or DMA can initiate transmission of data by writing to the data transmit register (DXR). Data written to DXR is shifted out on the BDX pin through a transmit shift register (XSR). This structure allows DXR to be loaded with the next word to be sent while the transmission of the current word is in progress.

On the receiver, receive frame synchronization and clocking is indicated by the BFSR and BCLKR pins, respectively. The DSP or DMA can read received data from the data receive register (DRR). Data received on the BDR pin is shifted into a receive shift register (RSR) and then buffered in the receive buffer register (RBR). If the DRR is empty, the RBR contents are copied into the DRR. If not,

the RBR holds the data until the DRR is available. This structure allows storage of the two previous words while the reception of the current word is in progress.

1.5.1.4 Data Movement

The DSP and DMA can move data to and from the McBSPs and can synchronize transfers based on McBSP interrupts, event signals, and status flags. The DMA is capable of handling data movement between the McBSPs and memory with no intervention from the DSP.

1.5.1.5 Programmable Functions

McBSP provides programmable clock and frame sync generation. Among the programmable functions are:

- ❑ Frame synchronization pulse width
- ❑ Frame period
- ❑ Frame synchronization delay
- ❑ Clock reference (internal vs. external)
- ❑ Clock division
- ❑ Clock and frame sync polarity

The on-chip companding hardware allows compression and expansion of data in either μ-law or A-law format. When companding is used, transmit data is encoded according to specified companding law and received data is decoded to 2s-complement format.

1.5.1.6 Multiple Channel Selection

The McBSPs allow multiple channels to be independently selected for the transmitter and receiver. When multiple channels are selected, each frame represents a time-division multiplexed (TDM) data stream. In using TDM data streams, the DSP may only need to process a few of them. Thus, to save memory and bus bandwidth, multichannel selection allows independent enabling of particular channels for transmission and reception. Up to 32 channels can be enabled in an up-to-128-channel bit-stream.

1.5.1.7 Clock-Stop Mode

The clock-stop mode (CLKSTP) in the McBSP provides compatibility with the serial peripheral interface (SPI) protocol. Clock-stop mode works with only single-phase frames and one word per frame. The word sizes supported by the McBSP are programmable for 8, 12, 16, 20, 24, or 32-bit operation. When the McBSP is configured to operate in SPI mode, both the transmitter and the receiver operate together as a master or as a slave.

The McBSP is fully static. Even when clock is stopped the McBSP registers will retain their contents. The maximum frequency is DSP clock frequency divided by two.

1.5.1.8 Data Clock Generation

A variety of data bit-clocks can be selected independently for the receiver and transmitter. These options include:

- The input clock to the sample rate generator can be either the DSP clock or an external clock input (CLKS).
- The input clock (DSP clock or external clock CLKS) source to the sample rate generator can be divided down by a programmable value.

1.5.2 Hardware Timer

The DSP subsystem features one independent software-programmable timer. Three memory-mapped registers (MMRs) control the operation of the timer. These registers are: the timer control register (TCR), the timer period register (PRD), and the timer counter register (TIM). The timer resolution is the DSP clock rate. The timer consists of a programmable 16-bit main counter (TIM), and a programmable 4-bit prescalar, and thus has a 20-bit dynamic range.

When the Timer is triggered the TIM register is loaded with the PRD register value. The main counter is driven by the 4-bit prescalar, which decrements by one at every CPU clock. Once the prescalar reaches zero, the 16-bit counter decrements by one. When the 16-bit counter decrements to zero, a maskable interrupt (TINT) is generated, and the counter is reloaded with the period value defined in the PRD. The timer can be stopped, restarted, reset, or disabled via the bits of the timer control register (TCR).

Refer to http://focus.ti.com/lit/ug/spru131g/spru131g.pdf for more information.

1.5.3 Direct Memory Access Controller (DMAC)

A brief description of the DMAC included in the DM320's DSP core is given in this section. For detailed information on the DSP's DMAC, refer to the C54x datasheet http://focus.ti.com/lit/ds/sprs082c/sprs082c.pdf and enhanced peripheral user's guide http://focus.ti.com/lit/ug/spru302/spru302.pdf. In DM320, DMAC will not have access to the shared memories section and Page1 program memory.

1.5.3.1 Features

The DSP subsystem includes a six-channel DMA controller, for performing data transfers independent of the CPU.

The DMA Controller includes the following features:

- Six independent configurable channels (DMA controller can keep track of the context of six independent block transfers)
- Higher priority for memory accesses than CPU
- Each channel has an independently programmable priority level
- There are configurable indexing mode registers to modify each channel's source and destination address during the data transfer. The address can remain constant, be post-incremented/decremented, or be adjusted by a programmable index.

- Each read or write transfer can be triggered by either a McBSP event or a timer event. Alternatively DMAC can be programmed to start the transfer immediately without waiting for a trigger event.
- Each DMA channel can interrupt the DSP upon completion of a half or entire data transfer.
- Supports double word transfers; i.e., a 32-bit transfer of two 16-bit words

The DSP DMA supports the following data transfers:

DARAM	<—>	DARAM	
SARAM (data)	<—>	DARAM	
SARAM (data)	<—>	SARAM	(data)
DARAM	<—>	IO	
SARAM (data)	<—>	IO	

Note: DARAM is 0x80 to 0x7FFF, SARAM (data) is 0x8000 to 0xBFFF (See also Figure 1 for DSP Memory Map)

1.6 DSP - Power Saving Modes

1.6.1 DSP Idle modes

The C54x DSP embedded in the DM320 has power-down modes in which it enters a dormant state and dissipates less power than normal operation while maintaining the CPU contents. This allows operations to continue unaltered when the power-down mode is terminated.

One of the power-down modes can be entered either by executing the IDLE 1 / IDLE 2 instructions or by using the ARM to drive the HOLD signal low when the HM status bit of ST1 register is set to ‘1’.

Power-down operation is summarized in the Table 5 and described in detail in the following sub-sections.

Operation/Feature	**IDLE1**	**IDLE2**	**HOLD**
CPU halted	Yes	Yes	Yes
CPU clock stopped	Yes	Yes	No
Peripheral clock stopped	No	Yes	No
Power-down terminated by			
HOLD driven high	No	No	Yes
Unmasked internal hardware interrupts	Yes	No	No
Unmasked interrupts from ARM/iMX/VLCD/DMAC/DCT	Yes	Yes	No
RS signal controlled by ARM DSP Controller module	Yes	Yes	No

Table 5: Operation During the Three Power-Down Modes

Depending on the state of the HM bit, DSP continues to execute unless the execution requires an external memory access.

1.6.1.1 IDLE1 Mode

The IDLE1 mode halts all DSP activities except the system clock. Because the system clock remains applied to the peripheral modules, the peripheral circuits

continue operating and the CLKOUT pin remains active. Thus, peripherals such as McBSP and timers can take DSP out of its power-down state.

Use the IDLE1 instruction to enter the IDLE1 mode. To terminate IDLE1, use a wake-up interrupt. If INTM = 0 when the wake-up interrupt takes place, DSP enters the ISR when IDLE1 is terminated. If INTM = 1, it continues with the instruction following the IDLE1 instruction. All wake-up interrupts must set to enable the corresponding bits in the IMR register regardless of the INTM value. The only exceptions are the non-maskable interrupts, RS and NMI.

1.6.1.2 IDLE2 Mode

The IDLE2 mode halts the on-chip peripherals as well as the CPU. Because the on-chip peripherals are stopped in this mode, they cannot be used to generate the interrupt to wake up the DSP as with IDLE1. However, power is significantly reduced because the device is completely stopped.

Use the IDLE2 instruction to enter the IDLE2 mode. To terminate IDLE2, assert any of the external interrupt signals (RS, NMI, and INT0) with a 10-ns minimum pulse. If INTM = 0 when the wake-up interrupt takes place, DSP enters the ISR when IDLE2 is terminated. If INTM = 1, it continues with the instruction following IDLE2 instruction. All wake-up interrupts must be set to enable the corresponding bits in the IMR register regardless of the INTM value. Reset all peripherals when IDLE2 terminates, especially the McBSP when the CLKS pin is used.

When RS is the wake-up interrupt in IDLE2, a 10-ns minimum pulse of RS can activate the reset sequence.

1.6.1.3 Hold Mode

The Hold mode is another power-down mode. Depending on the value of the HM bit (register ST1), this mode can be used to halt DSP.

This power-down mode is initiated by the HOLD signal. The ARM controls this signal by clearing the DHOLD bit in the HPIB_CNTRL register (0x30600). The effect of HOLD depends on the value of HM. If HM = 1, DSP stops executing. If HM = 0, DSP continues to execute internally. The DSP continues to operate normally unless an off-chip access is required by an instruction, then the processor halts until HOLD is released.

This mode does not stop the operation of on-chip peripherals (such as timers and serial ports). They continue to operate regardless of the HOLD level or the condition of the HM bit. This mode is terminated when HOLD becomes inactive.

1.6.2 Other Power-Down Capability

Switching the internal CLKOUT signal can reduce the DM320 DSP core power consumption. This internal signal is coming out of the DSP core but it is not used within the DM320 chip.

The CLKOUT-OFF feature allows DSP to disable CLKOUT using software instructions. The CLKOFF bit of PMST determines whether CLKOUT is enabled or disabled. At reset, CLKOUT is enabled. To gain a little bit on power consumption, you can disable this signal.

2 ARM-DSP Communication

2.1 Introduction

The Host Port Interface Bridge (HPIB) module in the DM320 controls data transfer and interrupts between the ARM and DSP. DM320 has the Texas Instruments TMS320VC5409-equivalent DSP built into it. This DSP has a Host Port Interface (HPI), which allows communication between a host CPU, such as the ARM926 core used in the DM320.

As shown in Figure 1, the ARM and DSP can communicate through the HPI. The ARM (or external host) can access the DSP memory through a 128-Kbyte window into the DSP memory map. Users can define a communication protocol and data structures understood by the ARM and DSP to pass command requests, acknowledgements, and datagram between the ARM and DSP. This would use the DSP shared memory and interrupts between ARM and DSP.

HPIB module also has a DMA controller, which enables data transfer from/to DSP internal memory to/from external memory (SDRAM and EMIF). This HPIB DMA controller is managed by DSP through XIO bus of DSP. HPIB DMA can interrupt the DSP using INT4 of the DSP interrupts, once the DMA transfer is completed.

In addition, DSP can also access MTC (SDRAM, Flash ROM etc) via Co-processor (COP) DMA. COP DMA can do data transfer between COP memory (Image buffers, iMX memory, VLCD memory and sequencer memory) and MTC (EMIF and SDRAM). This is separate from HPIB DMAC module. For details refer chapter on “Co-processor sub-system”. COP DMA can interrupt DSP using INT1 of the DSP interrupts, once the DMA transfer is completed.

Standard C5409 has DMA controller (DSP DMA) inside DSP. In DM320, DSP DMA can be used for data transfer between DSP internal memory and DSP peripherals like McBSP. Also, DSP DMA is separate from HPIB DMA and COP DMA. For details refer chapter on “DSP Subsystem”.

From ARM side, ARM, EHIF and MTC can access HPIB module on DSP side. Arbitrator in HPIB module controls each access as per the priority. Access priority of ARM side is given below:

1: EHIF (highest)

2: MTC

3: ARM (lowest)

HPIB, DSP core and DSP DMA can access DSP internal memory. Access priority is:

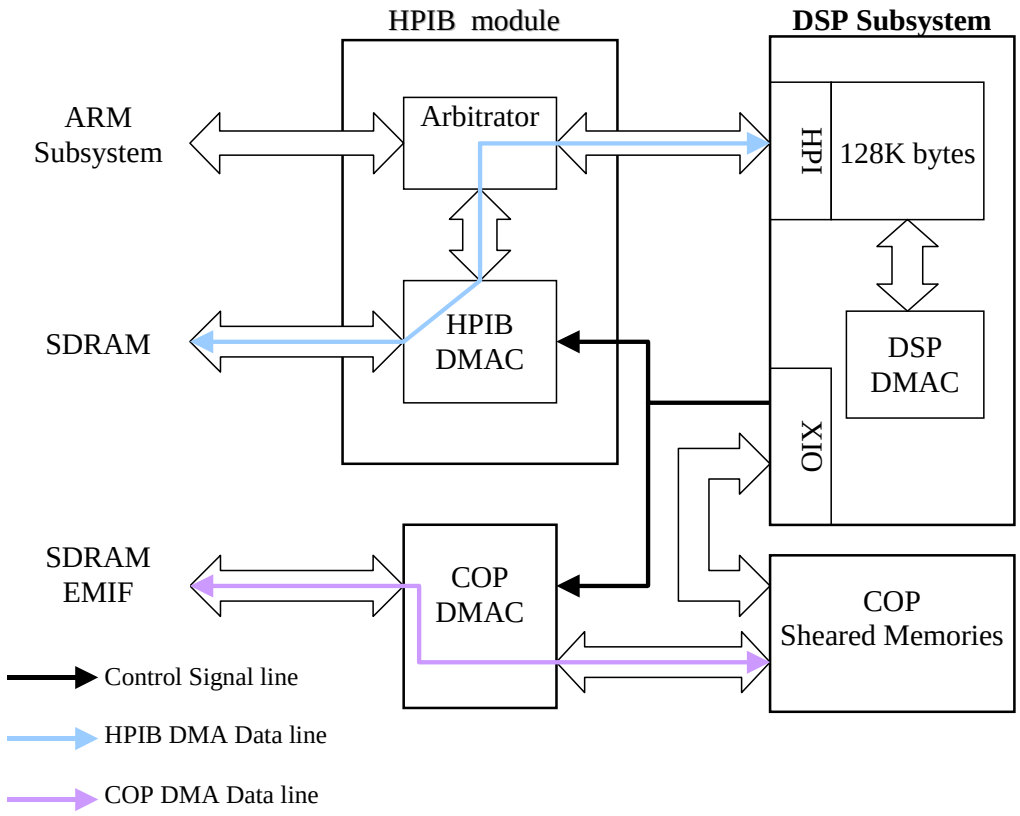
1: HPIB (highest)

2: DSP DMA

3: DSP core (lowest)

NOTE: HPIB DMAC is different from the DMA controller built into the shared memory interface of the COP subsystem and DSP DMA

Figure 1. ARM/DSP Communication



2.2 HPI-16 mode

HPI in TMS320VC5409 DSP can be used in two modes, HPI-8 mode and HPI-16 mode. However the TMS320VC5409-equivalent DSP built into the DM320 can be used only in non-multiplexed HPI-16 mode. In non-multiplexed HPI-16 mode, ARM can read and write the DSP memory through a 16-bit bus. However, in this mode handshaking is not possible. HINT interrupt is not supported in HPI-16 mode in DM320. Although handshaking is not possible, the ARM can still use the INT0 and NMI interrupts to interrupt the DSP, and the DSP can use co-processor interrupt controller to interrupt the ARM. For additional information on non-multiplexed HPI16 mode, refer to TMS320C54x DSP Reference Set, Volume 5: Enhanced Peripherals (http://focus.ti.com/lit/ug/spru302/spru302.pdf).

For more details on how DSP generates interrupt to ARM, refer Co-processor system module.

2.3 Memory Access

The DSP internal memory, extended data memory and page-1 program memory can be read and written by the ARM or an external host through the HPI-Bridge.

The memory maps of the ARM and external HOST are shown in ***Table 6*** and ***Table 7*** respectively. EXCHG bit in HPIBCTL register can be used to swap the upper and lower 8-bits of data (endian swap) transferred between the ARM and DSP.

Access speed from ARM to DSP memory via HPIB depends on ARM clock, DSP clock and HPIB state. Refer DM320 datasheet for more details.

ARM address(byte)	DSP address(word)	DSP memory	Size(byte)
0x40000-0x4FFFF	0x0000-0x7FFF*	Data & Program(Page0)	64k
0x50000-0x57FFF	0x8000-0xBFFF	Data	32k
0x58000-0x5FFFF	0xC000-0xFFFF	Program(Page1)	32k

Table 6: *ARM Memory Map*

ARM address(byte)	DSP address(word)	DSP memory	Size(byte)
0x20000-0x2FFFF	0x0000-0x7FFF*	Data & Program(Page0)	64k
0x30000-0x37FFF	0x8000-0xBFFF	Data	32k
0x38000-0x3FFFF	0xC000-0xFFFF	Program(Page1)	32k

Table 1.

Table 7: External HOST Memory Map

* DSP address 0x0000-0x005F is memory mapped register (MMR) area. This area is not accessible from ARM. DSP region 0x60-0x7F is reserved area.

2.4 Interrupts between ARM and DSP

ARM can interrupt the DSP by writing “0” to DINT0 bit or HPNMI bit of HPIBCTL register. This generates INT0 or NMI to DSP respectively. After an interrupt is triggered to DSP, ARM must write “1” to DINT0 bit or DNMI bit of HPIBCTL register. The interrupt signal must be held low for at-least two DSP clocks to be registered as an interrupt by the DSP.

The DSP can interrupt the ARM through the “Co-processor interrupt controller”. The DSP’s HINT pin is not used to interrupt the ARM in DM320. The DSP to ARM interrupt signal is connected to the ARM’s interrupt controller via INT11 interrupt. Refer to the module on co-processor sub-system for more details.

2.5 DSP Boot sequence

The boot-image for DSP is part of the ARM boot-image. There could be many different boot-images for DSP for the different tasks DSP needs to execute. ARM downloads the image related to the specific task to be executed by the DSP.

ARM resets the DSP via the DRST bit of HPIBCTL register and then bring the DSP out of reset. At this stage the DSP begins execution at 0x0FF80. The ROM code at this address initializes the DSP internal registers and places the DSP into IDLE1 mode. At this point ARM downloads the DSP code by using the HPI interface. After it completes downloading the DSP image, the ARM can send an interrupt to the DSP, which wakes it up from IDLE1 mode and jump to address 0x07F80 where it starts running the application code loaded by the ARM.

On the 'C54x DSP, by default the reset vector is located at address 0x0FF80. Within the DM320, this memory location has been ROM'ed with a small boot loader. The DSP boot sequence is given below.

- ARM resets DSP and then brings it out of reset.
- DSP gets out of reset and load its program counter (PC) register with 0x0FF80
- The ROM code in this location branches the DSP to address 0x0F800 where the initialization routine resides.
- DSP status register PMST is initialized to move the vector table to 0x07F80, all the interrupts are disabled except for INT0 and the DSP is set to IDLE1 mode.
- While DSP is in IDLE1 mode, the ARM loads the DSP Program/Data memory with the DSP code / data.
- When the ARM finishes downloading the DSP code, it wakes up DSP from IDLE1 mode by asserting INT0.
- The DSP then branches to address 0x07F80 where the new interrupt vector table is located. The ARM should have loaded this location with at least a branch to the start code.

Important:

- When INT0 is issued from ARM, DBIO bit in DSPC HPIB_CTL register should be "1". If DBIO = "0", DSP will enter test mode.
- In DSP ROM code, DSP address 0x7FFF will be cleared to "0". This is for indicating whether DSP can enter IDLE1 mode or not.
- Do not use HPNMI bit (NMI to DSP) when wake-up from DSP IDLE state in ROM is desired. HPNMI can wake up DSP from IDLE state but DSP branches to user NMI service routine and not to reset vector.

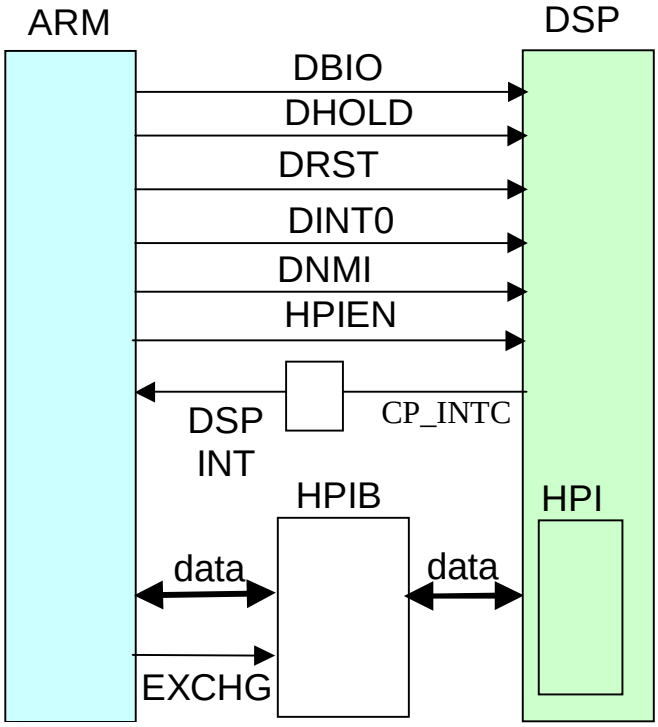
2.6 ARM-Side HPI Control and Status Registers

On the ARM side, the DSP Controller module has two registers, listed in DSP Controller Registers.

The registers HPIBCTL and HPIBSTAT contain DSP and HPIB control and status bits, which facilitate communication between the ARM and DSP. HPIBCTL is used to control various DSP signals and the HPIB as well as generate interrupts to the DSP. HPIBSTAT is a read only register that provides status information on the various DSP signals and HPI bridge control signals and interrupts.

Figure 2 shows ARM & DSP Communication interface. Table 8 lists DSP signals that have corresponding control and status bits in HPIBCTL and HPIBSTAT. Refer to the TMS320VC5409 Datasheet for detailed information on these signals.

Figure 2. ARM & DSP Communication interface



Pin Name	Module name
BIO	Branch control: A branch can be conditionally executed when BIO is active. If low, the processor executes the conditional instruction. For the XC instruction, the BIO condition is sampled during the decode phase of the pipeline; all other instructions sample BIO during the read phase of the pipeline. When waking up DSP from IDLE state in ROM, BIO should be “1”.
HOLDN	HOLD is asserted to request control of the address, data, and control lines. When acknowledged by the ’C54x, these lines go into the high-impedance state.
HOLDA	Hold acknowledge: HOLDA indicates that the ’5409 is in a hold state and that the address, data and control lines are in the high-impedance state, allowing the external memory interface to be accessed by other devices.
RSN	Reset: RSN causes the DSP to terminate execution and causes a re-initialization of the DSP and peripherals. When RSN is brought to a high level, execution begins at location 0xFF80h of program memory.
INT0	External user-interrupt: INT0 is prioritized and maskable by the interrupt mask register (IMR) and the interrupt mode bit. INT0 can be polled and reset by using interrupt flag register (IFR).
DNMI	Non-maskable interrupt. DNMI is an external interrupt that cannot be masked by way of the INTM or the IMR. When DNMI is activated, the processor traps to the appropriate vector location.
HPIEN	HPIENA must be driven high during reset to enable the HPI. An internal pull-down resistor is always active and the HPIENA pin is sampled on the rising edge of RSN. If HPIENA is left open or is driven low during reset, the HPI module is disabled. Once the HPI is disabled, the HPIENA pin has no effect until the ’5409 is reset.

Table 8: DSP Signal description

2.7 DSP Subsystem - HPIB DMAC

DM320 has capability to perform DMA from DSP internal memory to SDRAM or EMIF via the Memory Traffic Controller. This DMA is controlled by the DSP

To transfer data using the HPIB DMA controller, the DSP should setup the DMA transfer registers as follows:

Perform the following steps depending on the SDRAM or EMIF data source/destination device:

- To transfer data between SDRAM, set the SDRAM source/destination address in SDEM_ADDRL and DSP_ADDRH
- To transfer data between EMIF region CS0, CS3, or CS4; set the EMIF destination/source address in SDEM_ADDRL and SDEM_ADDRH
- To transfer data between EMIF region CS1 or CS2, set the EMIF destination/source address in the EMIF module registers IMGDSPADDH and IMGDSPADDL.
- For EMIF data transfer, select the EMIF source/destination region (CS0/CS1/CS2/CS3/CS4) by configuring DDST in IMGDSPDEST register
- For SDRAM data transfer, select the SDRAM channel.
- Select whether SDRAM or EMIF would be used for data transfer using DEST bit in DMA_CTRL register
- NOTE: SDRAM, EMIF address is specified as byte offset from start of respective SDRAM, EMIF region. See MTC section for EMIF register details. Address must be 4 byte aligned
- Set the DSP internal memory source/destination address in DSP_ADDRL and DSP_ADDRH. Valid DSP memory addresses are,0x0080 to 0x7FFF – DARAM (data/program), 0x8000 to 0xBFFF – SARAM (data) and 0x1C000 to 0x1FFFF – SARAM (program)
- Set the size of the data to be transferred in DMA_S. The size must be multiple of 4 bytes
- Set the ENDI field in DMA_CTRL register to set any byte/word swapping requirements during the DMA transfer
- Select the direction of the DMA transfer by configuring DIR field of DMA_CTRL register
- Initiate the DMA request by setting RUN field of DMA_TRG register
- Check for DMA completion by polling on bit RUN field of DMA_TRG
- Remaining bytes of DMA transfer can be found from the DMA_REST register.

Important: DSP can check if HPIB DMA is running or not using DMATRG bit. However, DMATRG bit has 2 DSP clock latency. After starting HPIB DMA by writing “1” into DMATRG bit, it is necessary to wait at least 2 DSP cycle for polling.

NOTE1: HPIB DMAC is separate from the DMA controller built into the shared memory interface of the COP subsystem.

NOTE2: HPIB DMAC is separate from the DMA controller built into DSP.

NOTE3: HPIB DMA would be slower than COP DMA. Since HPIB DMA access is not done in bursts of 32bytes and hence it takes more cycles to complete.

NOTE4: Select SDRAM DMA channel for the HPIB DMA to SDRAM in DMASEL register of SDRAM controller.

2.8 DSP Controller (DSPC)

Address	Register	Description
0003:0600	HPIBCTL	HPIB Control Register
0003:0602	HPIBSTAT	HPIB Status Register

2.9 DSP Controller Registers (ARM side)

2.9.1 HPIBCTL

DSPC HPIB Control Register

HPIBCTL		0003:0600		offset:	0x00										default:	0x0789
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	DBIO	DHOLD	DRST	DINT0	RSV	EXCHG	RSV	HPNMI	RSV	RSV	HPIEN
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	-	R/W	-	R/W	-	-	R/W
Default	-	-	-	-	-	1	1	1	1	-	0	-	1	-	-	1

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10	DBIO	1	R/W	DSP BIO Control » 0: DSP BION = low « » 1: DSP BION = high « DBIO controls the state of the DSP's BIOn pin. When the DSP is the BIO state, it can branch and control the program.
9	DHOLD	1	R/W	DSP Hold Request Control » 0: Hold « » 1: Normal « DHOLD controls the state of the DSP's HOLDn pin.
8	DRST	1	R/W	DSP Reset » 0: DSP reset « » 1: Normal « DRST controls the state of the DSP's RSn pin, and is used to reset the DSP.
7	DINT0	1	R/W	DSP INT0 Control (ARM-To-DSP Interrupt) » 0: INT0 « » 1: Normal « DINT0 controls the state of the DSP's INT0n pin, and it is used interrupt the DSP from the ARM. The INT0 interrupt is generated in the DSP when the the ARM sets this bit to "0". After the interrupt is generated, the ARM must return this bit to "1".
6	RSV			Reserved
5	EXCHG	0	R/W	Byte Exchange » 0: No exchange « » 1: Exchange upper/lower bytes of 16-bit HPI16 transfers « The exchange is done when data is transfered in both directions ARM to DSP and DSP to ARM (write and read).
4	RSV			Reserved
3	HPNMI	1	R/W	HPI NMI Control (NMI Interrupt) » 0: NMI « » 1: Normal « HPNMI controls the state of the DSP's NMIn pin, and it is used to interrut the DSP from the ARM. The NMI interrupt is generated in the DSP when the ARM sets this bit to '0'. After the NMI interrupt is generated, the ARM must return this bit to '1'.
2:1	RSV			Reserved
0	HPIEN	1	R/W	Enables HPI Control » 0: Disabled « » 1: Enabled « HPIEN controls the DSP's HPIEN pin. When HPIEN is disabled, all HPI functions are invalid.

2.9.2 HPIBSTAT

DSPC HPIB Status Register

HPIBSTAT		0003:0602		offset:	0x02										default:	0x1100
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	DXF	RSV	RSV	RSV	HOLDA	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R/W	-	-	-	R	-	-	-	R	-	-	-	-	-	-	-	-
Default	-	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-

Bit	Name	Reset Value	R/W	Function
15:13	RSV			Reserved
12	DXF	1	R	DSP XF Status Indicates status of DSP XF signal output
11:9	RSV			Reserved
8	HOLDA	1	R	DSP Hold Acknowledge » 0: DSP hold status « » 1: Normal « Indicates status of the DSP's HOLDA signal output. After a hold request is sent to the DSP by DHOLD in HPIBCLT, the acknowledgement can be checked by this bit.
7:0	RSV			Reserved

2.10 Host Port Interface Bridge Register Map (HPIB)

Page	Address	Register	Description
IO	8000	SDEM_ADDRL	SDRAM/EMIF Address Low
IO	8001	SDEM_ADDRH	SDRAM/EMIF Address High
IO	8002	DSP_ADDRL	DSP Address Low
IO	8003	DSP_ADDRH	DSP Address High
IO	8004	DMA_SIZE	DMA Transfer Size
IO	8005	DMA_CTRL	DMA Control
IO	8006	DMA_TRG	DMA Trigger
IO	8007	DMA_REST	DMA Transfer Remaining Amount

2.11 Host Port Interface Bridge Registers

2.11.1 SDEM_ADDRL

HPIB DMA SDRAM Address Low Register

SDEM_ADDRL	IO	8000	offset:	0x00										default:	0x0000
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	ADDRL[15]	ADDRL[14]	ADDRL[13]	ADDRL[12]	ADDRL[11]	ADDRL[10]	ADDRL[9]	ADDRL[8]	ADDRL[7]	ADDRL[6]	ADDRL[5]	ADDRL[4]	ADDRL[3]	ADDRL[2]	ADDRL[1]	ADDRL[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	ADDRL	0	R/W	bit[15:0] of SDRAM/EMIF DMA address (unit = bytes) Note: Byte offset from start of SDRAM/EMIF base address 2 LSBs are fixed at zero to force 32-bit alignment

2.11.2 SDEM_ADDRH

HPIB DMA SDRAM Address High Register

SDEM_ADDRH	IO	8001	offset:	0x01										default:	0x0000
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	ADDRH[11]	ADDRH[10]	ADDRH[9]	ADDRH[8]	ADDRH[7]	ADDRH[6]	ADDRH[5]	ADDRH[4]	ADDRH[3]	ADDRH[2]	ADDRH[1]	ADDRH[0]
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	ADDRH	0	R/W	bit[27:16] of SDRAM/EMIF DMA address Note: Byte offset from start of SDRAM/EMIF base address

2.11.3 DSP_ADDRL

HPIB DMA DSP Address Low Register

DSP_ADDRL	IO	8002	offset:	0x02										default:	0x0000
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	ADDRL[15]	ADDRL[14]	ADDRL[13]	ADDRL[12]	ADDRL[11]	ADDRL[10]	ADDRL[9]	ADDRL[8]	ADDRL[7]	ADDRL[6]	ADDRL[5]	ADDRL[4]	ADDRL[3]	ADDRL[2]	ADDRL[1]	ADDRL[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	ADDRL	0	R/W	bit[15:0] of DSP DMA start address - word (16-bit) address

2.11.4 DSP_ADDRH

HPIB DMA DSP Address Start Register

DSP_ADDRH	IO	8003	offset:	0x03										default:	0x0000
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	ADDRH
R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R/W
Default	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Bit	Name	Reset Value	R/W	Function
15:1	RSV			Reserved
0	ADDRH	0	R/W	bit[16] of DSP DMA start address - word (16-bit) address

2.11.5 DMA_SIZE

HPIB DMA transfer size

DMA_SIZE		IO	8004	offset:	0x04										default:	0x0000
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	DSZ[15]	DSZ[14]	DSZ[13]	DSZ[12]	DSZ[11]	DSZ[10]	DSZ[9]	DSZ[8]	DSZ[7]	DSZ[6]	DSZ[5]	DSZ[4]	DSZ[3]	DSZ[2]	DSZ[1]	DSZ[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	DSZ	0	R/W	DMA Transfer Size (bytes) - 2 LSBs are fixed at zero to force 32-bit units

2.11.6 DMA_CTRL

HPIB DMA Control

DMA_CTRL	IO	8005	offset:	0x05										default:	0x0000
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	ENDI[1]	ENDI[0]	RSV	RSV	RSV	MEM	RSV	RSV	RSV	DIR
R/W	-	-	-	-	-	-	R/W	R/W	-	-	-	R/W	-	-	-	R/W
Default	-	-	-	-	-	-	0	0	-	-	-	0	-	-	-	0

Bit	Name	Reset Value	R/W	Function
15:10	RSV			Reserved
9:8	ENDI	0	R/W	Endianness of transferred bytes » 00: D3 D2 D1 D0 -> D3 D2(A1) D1 D0(A0) « No swap (little endian) » 01: D3 D2 D1 D0 -> D0 D1(A1) D2 D3(A0) « Byte and word swap (big endian) » 10: D3 D2 D1 D0 -> D2 D3(A1) D0 D1(A0) « Byte Swap within words (little endian with byte swap) » 11: D3 D2 D1 D0 -> D1 D0(A1) D3 D2(A0) « Word swap (big endian with byte swap)
7:5	RSV			Reserved
4	MEM	0	R/W	DMA Destination Memory Select » 0: SDRAM « » 1: External Memory IF «
3:1	RSV			Reserved
0	DIR	0	R/W	DMA Transfer Direction » 0: SDRAM/EMIF to DSP « » 1: DSP to SDRAM/EMIF «

2.11.7 DMA_TRG

HPIB DMA Trigger

DMA_TRG		IO	8006	offset:	0x06										default:	0x0000
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	TRG
R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R/W
Default	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Bit	Name	Reset Value	R/W	Function
15:1	RSV			Reserved
0	TRG	0	R/W	DMA transfer trigger DMA transfer starts when '1' is written. Writing '0' during transfer cancels this transfer. '1' is read during transfer and reverts to '0' automatically after completing transfer. NOTE: When polling and the like, it takes several cycles from writing 1 to setting of transfer status.

2.11.8 DMA_REST

HPIB DMA Remaining bytes

DMA_REST		IO	8007	offset:	0x07										default:	0x0000
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	REM[15]	REM[14]	REM[13]	REM[12]	REM[11]	REM[10]	REM[9]	REM[8]	REM[7]	REM[6]	REM[5]	REM[4]	REM[3]	REM[2]	REM[1]	REM[0]
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	REM	0	R	DMA Bytes Remaining

3 Coprocessor Subsystem

3.1 Introduction

The coprocessor (COP) subsystem is designed to accelerate image processing and computation-intensive applications. The coprocessor subsystem consists of the following components:

- Coprocessors (iMX, VLCD and DCT)
- Shared memories (Image buffers, iMX command memory etc)
- Control registers
- Coprocessor DMA controller
- Coprocessor Clock controller
- Coprocessor Interrupt controller

There are three coprocessors in the system, Imaging Extension (iMX), Variable Length Coder/Decoder (VLCD) and Discrete Cosine Transform (DCT). All the memories and control registers are accessible by ARM, DSP and Sequencer through the coprocessor bus. All the shared memories reside in the data memory of the DSP data address space and control registers for various co-processors and DMA reside in the DSP I/O space. All the shared memories and control registers are memory mapped for ARM and sequencer. **Figure 3** shows the architecture of the Coprocessor Subsystem.

The iMX accelerates computation-intensive imaging processing functions such as color conversion, filtering and interpolation.

The VLCD provides acceleration for Huffman coding and decoding as well as quantization and inverse quantization required in various still image and video algorithms.

The DCT engine implements the compression flow to transform blocks of image in the spatial domain to the 2-D frequency domain. IDCT (Inverse DCT) operation converts from frequency domain to spatial domain. DCT module in DM320 performs 2-D 8x8 DCT and IDCT. Performance will be 2 cycles per DCT coefficient for forward or inverse 2-D transforms.

The sequencer is a simple microcontroller optimized for control / sequencing tasks. It also has simple arithmetic functions to perform simple buffer management, register programming, and looping control.

The COP DMA controller provides a channel to SDRAM and EMIF, allowing high-speed data transfer between the SDRAM / EMIF and COP shared memories.

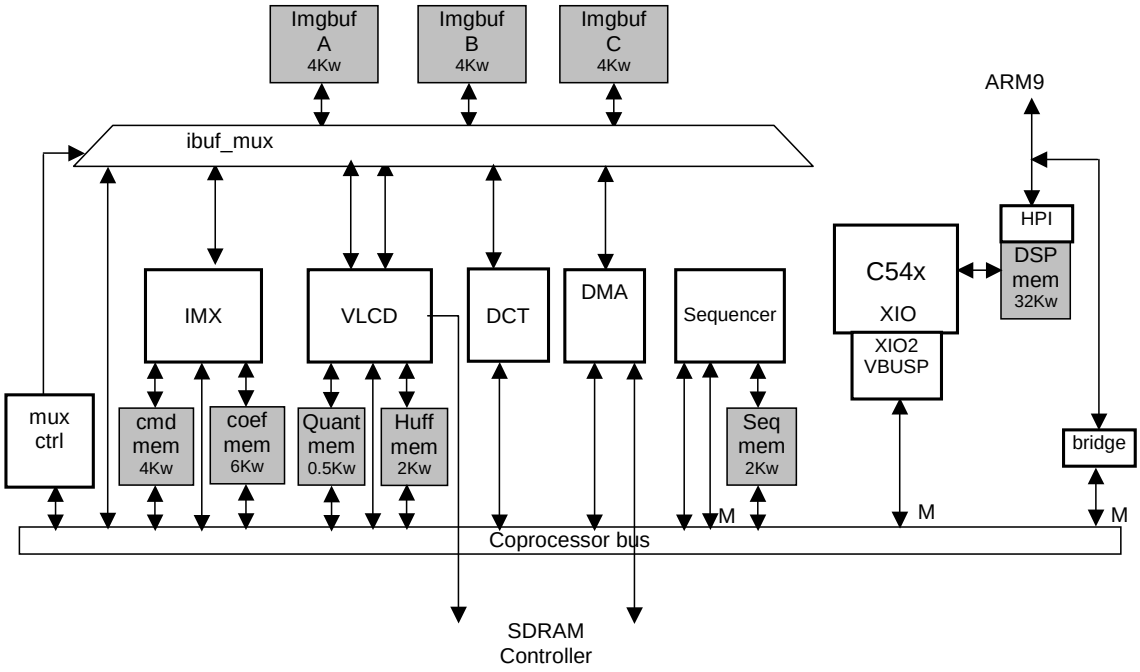


Figure 3: ***Coprocessor Subsystem Block Diagram***

Each coprocessor except DCT has their own private memory to store data structures and / or instructions unique to the functional unit. In addition ARM, Sequencer, DSP, Co-processors and COP DMA controller also communicate with each other. The communication is made possible by shared memory, control registers and interrupts.

Figure 4 shows the shared memory blocks. There are a total of 8 shared memory blocks. Each memory block is connected through a memory wrapper to the accessing hosts. Multiplexing is statically controlled; DSP, ARM or sequencer writes to the image buffer control register to set the host that can access a particular shared memory block. Each coprocessor and the COP DMA controller can interrupt the hosts. The details of each shared memory is explained in the following sections.

Note: The user needs to configure the DSP's Software Wait State Register SWWSR to add one wait state when accessing these memory blocks (Image Buffer A/B/C, iMX coefficient/command buffer, and VLCD table buffer). No wait state is required to access the control registers in the DSP's I/O space.

Note: DSP should never accesses any coprocessor register that sequencer/ARM might be accessing. Algorithm schedule and DSP/sequencer/ARM synchronization via software should be used to enforce this.

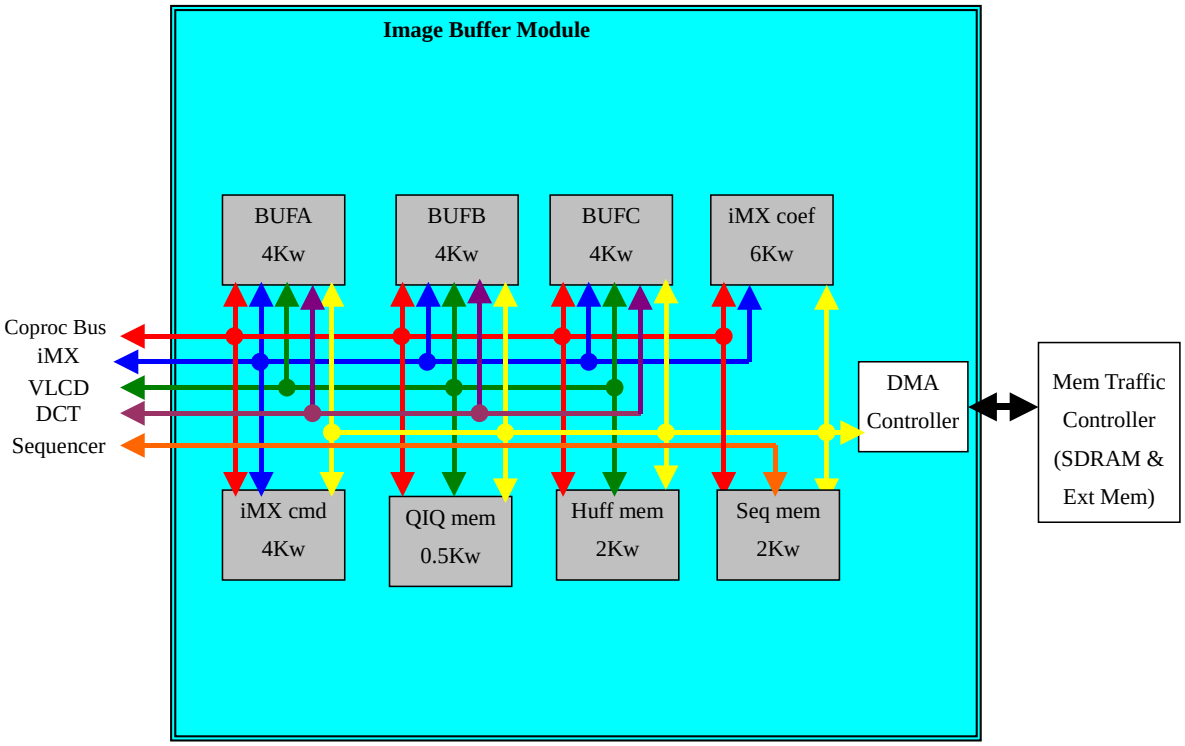


Figure 4: ***Shared Memory Block Diagram***

Image Buffers

There are three image buffers: image buffer A, image buffer B and image buffer C. Image buffers are used as working memory for image processing and compression/decompression, and are shared among iMX, DCT, VLCD_ibuf0, VLCD_ibuf1, DMA, and Coprocessor Bus, via 6-to-1 multiplexing. ARM, DSP and sequencer access these buffers through the Coprocessor Bus. For the DSP all the three Image Buffers appears in the same data memory space and only one buffer can be accessed at a time. This is because of the limited data memory space on the DSP XIO bus. iMX has the option of accessing two image buffers at the same time. In the two-buffer mode, iMX addresses BUFA/B in the lower 4K-word address, and BUFC in the 4K-word range above that as illustrated in Table 9:

Mode	Byte Address	Buffer
Normal	0x0000 – 0x1FFF (4Kw)	BUFA/B/C
IMX_sequential_buffer	0x0000 – 0x1FFF (4Kw) 0x2000 – 0x3FFF (4Kw)	BUFA/B BUFC

Table 9 iMX address space: Sequential Buffer Mode vs Normal Mode

IMX coefficient memory

This memory is used for coefficients and temporary working space for iMX. Host initializes the filter coefficients, lookup tables, and scaling constants into this memory (through direct write or DMA) and transfers control to iMX.

IMX command memory

This memory is used as the program/command memory for the iMX. Before iMX is active, host needs to download the iMX commands into this memory (through direct write or DMA). During computation, while iMX is idle, host may switch the command memory to itself and modify the commands or parameters. Due to DSP's limited address space, its access to iMX command memory is paged. IMXCMD bit of IMG_MODE register controls which half of the iMX command memory DSP sees from its 0xE800 – 0xEFFF address range.

VLCD QIQ table memory

This memory is used to store the quantization/inverse quantaization tables for the VLCD's QIQ compute engine. The host loads the tables through direct write or DMA, and then the control is transferred to VLCD.

VLCD Huffman table memory

This memory is used to store the Huffman tables for the VLCD's Huffman encode/decode engine. Host loads the tables through direct write or DMA. Host can place bit segment data (for MB header, as value-length pairs) to be encoded prior to the main Huffman data. VLCD will process these before processing the main Huffman data.

Sequencer memory

Sequencer memory is used as the program/data memory for the sequencer. ARM or DSP initializes this memory (through direct write or DMA) with sequencer program and data. This memory is on-the-fly arbitrated among sequencer's pap (program access port) and Coprocessor Bus, and is statically switched between this arbitrated access and DMA access. This memory is thus useful for small amount of data exchange between sequencer and ARM / DSP. Due to DSP's limited address space, its access to sequencer memory is paged. SEQMEM bit in IMG_MODE register controls which half of the sequencer memory DSP sees from its 0xF800 – 0xFBFF address range.

3.2 Coprocessor subsystem: Peripheral Register Map

Using control and configuration registers mapped into I/O space, DSP controls co-processors and shared memory. ARM and sequencer also can access these control registers within their address space. The memory map for the control registers from the DSP's address space is shown in the Table 10

Module	Page	Start Addr	End Addr
Image Buffer Control	IO	0000:0000	0000:007F
iMX Control	IO	0000:0080	0000:00FF
VLCD Control	IO	0000:0100	0000:017F
DCT Control	IO	0000:0180	0000:01FF
Sequencer Control	IO	0000:0200	0000:027F
Interrupt-Clock control	IO	0000:0280	0000:02FF
HPIB	IO	0000:8000	0000:800F

Table 10: DSP Peripheral Register Map for Coprocessor control registers

Within the ARM and Sequencer address space, the control register address map is as shown in Table 11.

Module	ARM address	Sequencer Address
Image Buffer Control	0x9E000-0x9E0FF	0xF000-0xF07F
iMX Control	0x9E100-0x9E1FF	0xF080-0xF0FF
VLCD Control	0x9E200-0x9E2FF	0xF100-0xF17F
DCT Control	0x9E300-0x9E3FF	0xF180-0xF1FF
Sequencer Control	0x9E400-0x9E4FF	0xF200-0xF27F
Interrupt/Clock Control	0x9E500-0x9E5FF	0xF280-0xF2FF

Table 11: Arm and Sequencer Memory map for Coprocessor control registers

3.3 Shared Memory Access Switching and Memory Map

Table 12 shows which modules can access each shared memory. The DSP, ARM or Sequencer can control which memory is connected to which access module by configuring the BUF_MUX0 and BUF_MUX1.

Memory	Accessing Host				
	CP Bus (Arm, DSP and Sequencer)	IMX	VLCD	DMAC	DCT
Image BUFA/B/C	Yes	Yes	Yes	Yes	Yes
iMX coefficient	Yes	Yes	No	Yes	No
iMX command	Yes	Yes	No	Yes	No
VLCD Huffman	Yes	No	Yes	Yes	No
VLCD QIQ	Yes	No	Yes	Yes	No
Sequencer	Yes	No	No	Yes	No

Table 12: Shared Memory Access Control

The memory map for the various shared are given in Table 13.

Memory	DSP Address (Word)	Arm Address (Byte)	Sequencer Address (Word)	iMX Address (Byte)	Size (Words)
Image BUFA/B/C[1]	0xC000-0xCFFF	0x80000-0x81FFF	0x0000-0x0FFF	0x0000-0x1FFF	4k
iMX coefficient	0xD000-0xE7FF	0x82000-0x84FFF	0x1000-0x27FF	0x8000-0xAFFF	6.5k
iMX command	0xE800-0xEFFF	0x85000-0x85FFF	0x2800-0x2FFF	0x0000-0x0FFF [2]	2k
VLCD Huffman	0xF000-0xF7FF	0x86000-0x86FFF	0x3000-0x37FF	-	2k
Sequencer (Paged)	0xF800-0xFBFF	0x87000-0x877FF	0x3800-0x3BFF	-	1k
VLCD Coefficient	0xFC00-0xFDFF	0x87800-0x87B00	0x3C00-0x3DFF	-	512

Table 13: DM320 Coprocessor Memory Map

[1] IMGBUFSEL field of BUF_MUX0 register is valid only for this selection.

[2] Separate address space for iMX and word addressable

ARM and Sequencer has the option of accessing the image buffers in the overlapped address space for the image buffers as given in Table 13 or accessing them separately as three buffers. ARM and sequencer can access the whole iMX command and Sequential memory, unlike DSP which access them in paged mode. The address map for the individual buffer access by ARM and Sequencer is given below in Table 14.

Memory	ARM Address (Byte)	Sequencer Address (Word)
Image BUFA	0x88000-0x89000	0x4000-0x4FFF
Image BUFB	0x8A000-0x8BFFF	0x5000-0x5FFF
Image BUFC	0x8C000-0x8DFFF	0x6000-0x6FFF
iMX command (Whole)	0x8E000-0x8FFFF	0x7000-0x7FFF
Sequencer(Whole)	0x90000-0x90FFF	0x8000-0x8FFF

Table 14: Arm and Sequencer Address map for individual access of shared memories

3.4 COP DMA Controller

The COP subsystem has a DMA controller that allows the hosts to initiate data transfers between any of the shared memory blocks (Image Buffers A/B/C, iMX memories, VLCD memories and sequencer memory) and SDRAM or external memory. The DMA controller will handle the handshaking sequence to/from SDRAM or the External Memory controller automatically.

The COP DMA controller supports 2-dimensional data transfers. As shown in **Figure 5**, DMA_XNUM specifies the number of pixels in the horizontal direction, and DMA_YNUM specifies the number of lines in the vertical direction. This function is useful in cases where processing in MACRO block units is required, such as in JPEG compression.

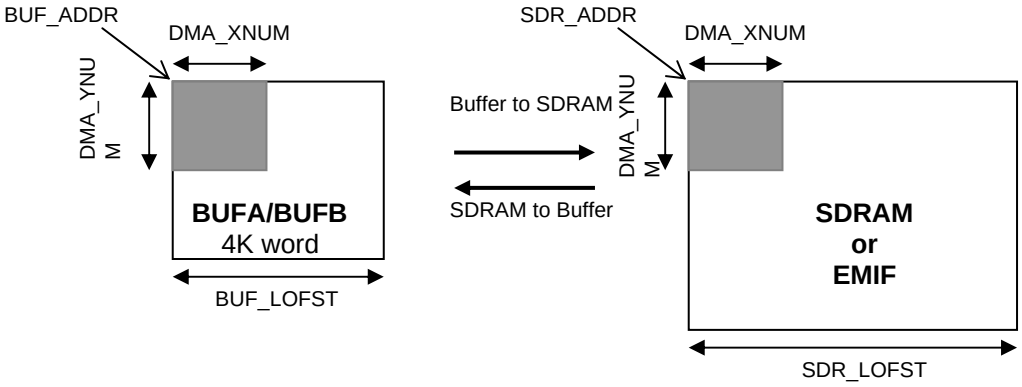


Figure 5: 2-D Data Transfer using Image Buffer DMA Controller

The start address offset of shared memory is specified in BUF_ADDR. The start address offset of SDRAM is specified in SDEM_ADDRH and SDEM_ADDRL. Line offset value for shared memory is set in BUF_LOFST and for SDRAM is set in SDEM_LOFST. When the line-offset values (BUF_LOFST and SDEM_LOFST) is less than the value of DMA_XNUM , it is overwritten as shown in **Figure 6**.

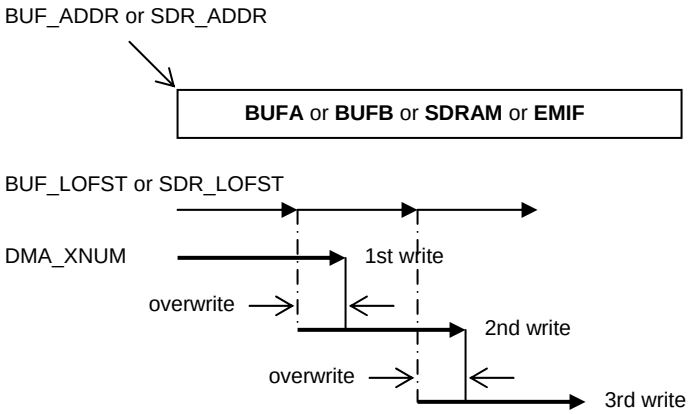


Figure 6: Data overwrite when LOFST < DMA_XNUM

3.4.1 Interrupting the DSP after DMA Transfer Completion

When the DMA completes, the DMA controller can interrupt the DSP. The DMA controller interrupt is connected to the DSP's INT1. The DSP can enable this interrupt (through interrupt mask register IMR) so that an interrupt service routine (ISR) is called upon transfer completion. Alternatively, the DSP can disable this interrupt and poll the bit 1 in the DMA_CTRL.

3.4.2 Byte to Word Conversion

The Image buffer DMA controller also has a function mode of byte–to-word conversion. In this mode, the 32bit(4 bytes) read data from SDRAM is converted to 64bit(4 words), and stored into image buffer. **Table 15** shows data format in image buffer when byte to word conversion is enabled. Byte to word conversion can be enabled using WORD field in IMG_MODE.

SDRAM read data	**0(LSB)**			**31(MSB)**				
	Byte0	Byte1	Byte2	Byte3				
Image Buffer Address	4n (word)		4n+1		4n+2		4n+3	
Write data	byte0	0	Byte1	0	Byte2	0	Byte3	0

Table 15: Byte to word conversion

3.4.3 Steps to Configure the COP DMA Controller

To transfer data using the COP DMA controller, the DSP, ARM or Sequencer can setup the DMA transfer registers as follows:

- Enable the clock to the shared memory involved in the DMA transfer using the CP_CLKC register.
- Perform the following steps depending on the SDRAM or EMIF data source/destination device:
  - To transfer data between SDRAM, set the SDRAM source/destination address SDEM_ADDRH and SDEM_ADDRL
  - For EMIF data transfer, select the EMIF source/destination region (CS0/CS1/CS2/CS3/CS4) by configuring IDST in IMGDSPDEST register in EMIF.
  - To transfer data between CS0, CS3, or CS4; set the EMIF destination/source address in SDEM_ADDRH and SDEM_ADDRL
  - To transfer data between CS1 or CS2, set the EMIF destination/source address in the EMIF module registers IMGDSPADDH and IMGDSPADDL
  - Select whether SDRAM or EMIF would be used for data transfer using DESTSEL bit in DMA_CTRL.
  - NOTE: SDRAM, EMIF address is specified as word offset from start of respective SDRAM, EMIF region.
- Set the SDRAM/EMIF data line offset in SDEM_LOFST
- Set the shared memory source/destination address for shared memory in BUF_ADDR.

  Note: Address is specified as offset from start of the respective memory.
- Connect the corresponding shared memory to/from which DMA transfer is expected using the Buffer control Mux registers BUF_MUX0 or BUF_MUX1. In case of iMX command memory or Sequencer memory, which are paged memories in the DSP's XIO address space, select the page through IMG_MODE register.
- Set the shared memory buffer line offset in BUF_LOFST
- Set the access block size in the X and Y directions in DMA_XNUM and DMA_YNUM. Ensure that block sizes and line offsets are so chosen that there is no overwriting issues as described in **Figure 6**.
- Set the bits in IMG_MODE to specify whether data byte swapping, data shifting and/or byte-to-word conversion is required during the DMA transfer.
- Select the direction of the DMA transfer by configuring bit DIR of DMA_CTRL
- Initiate the DMA request by setting DMAON bit of DMA_CTRL
- Check for DMA completion by either polling DMAON bit of DMA_CTRL. DMAON will be cleared once the DMA transfer is over. This interrupt status is also available at SEQ_SYNC, DSP_SYNC, ARM_SYNC and SEQ_INT_STATE registers.

3.4.4 COP DMA controller restrictions

The COP DMA controller has certain restrictions:

- When the COP DMAC transfers data from Image Buffer A/B/C or iMX coefficient Buffer, it can transfer from even and odd address (in DSP word units) to SDRAM/EMIF. But in case of iMX Command, VLCD Huffman/QIQ buffer and Sequencer memory, the DMAC can transfer from only an even address location to SDRAM or EMIF.
- DMA transfers are most efficient when done in units of SDRAM/EMIF bursts. A burst consists of 16 word units. It is possible to transfer data in word units, however, the time required will be equal to the time required for a burst transfer. For example, to transfer one word from image buffer A to SDRAM, the time required will be equal to the time required to transfer one burst (16 words).
- DMA transfers are most efficient when the starting SDRAM/EMIF address is 256-bit (256-bit = 16 words) aligned. When the starting address is not 256 bit aligned, then the DMA controller requires up to 2 (1 additional) burst transfers to account for the unaligned data.
- The DMA controller supports 2-dimensional SDRAM/EMIF accesses. The DMA controller can read an NxM block of data out of a larger block of data; however, the transfer is most efficient when the block is a multiple of 256-bits wide.
- Allow to change one bit only, not more than two bits in BUF_MUX[9:2] if BufA or B or C selects DMA on BUF_MUX0 or IMX Coef Mem selects DMA on BUF_MUX1[1:0]. Also, Don't set DMA state in this change. So the change of pallarel registers as both VLCD Huffman and VLCD quantization are not acceptable. And the "1"(01) -> "2"(10) switching in one memory category like VLCD Huffman is not acceptable.

VLCD Huffman memory selection don't use DMA during BUFA is using DMA.

OK: VLCD->CPBUS->VLCD->CPBUS

NG: VLCD->CPBUS->DMA->CPBUS

If you want to change the access VLCD Huff mem as VCLD->CPUBS->VLCD during BUFA is using DMA, Pls refer the below sequence.

…….

BUF_MUX1 = 0x0050; <- VLCDHuf,Q->VLCD mode

BUF_MUX0 = 0x0003; <- BufA : DMA mode

…….

BUF_MUX1 = 0x0010; <- VLCDHuf->CPBUS, VLCDQ->VLCD mode

BUF_MUX1 = 0x0050; <- VLCDHuf,Q->VLCD mode

3.5 Coprocessor subsystem Clock Control

ARM, DSP and Sequencer can enable/disable clock to iMX, VLCD, Image Buffer, Sequencer and DCT using CP_CLKC.

Note: Image buffer, iMX, VLCD, Sequencer and DCT clocks can also be enabled by configuring the MOD1 register of ARM Clock Controller. These clocks are enabled when either corresponding bits in register MOD1 are set **OR** when corresponding bits in CP_CLKC are set.

3.6 Coprocessor subsystem Interrupt/Strobe Control

The various coprocessors can generate interrupt/strobe signals. The interrupts from coprocessor subsystem is hooked up to the INT1, INT2, and INT3 external interrupts of the DSP.

- INT1: Interrupt is generated based on the settings of DSP_SYNC_STATE and registers or when DSPINT1 bit in CP_INTC is set.
- INT2: Interrupt is generated when DSPINT2 bit in CP_INTC is set.
- INT3: Interrupt is generated when DSPINT3 bit in CP_INTC is set or on write of any value to BRKPT_TRG.

The Coprocessor subsystem can generate an interrupt signal to ARM. This active-low interrupt pulse is sent when

- ARMINT bit in CP_INTC is set.
- Based on the settings of ARM_SYNC_STATE and ARM_SYNC_MASK registers.

iMX, VLCD, and DCT sends strobe pulses upon task completion. Sequencer sends its strobe pulse by setting SEQSTRB bit of CP_INTC. DSP sends its strobe pulse by setting DSPSTRB bit of CP_INTC. ARM sends its strobe by setting ARMSTRB bit of CP_INTC. As the Coprocessor Bus arbitrates and serializes accesses from ARM, DSP and sequencer, there is no coherence problem if ARM, DSP and Sequencer each write 1 to the bit position it is assigned to. Note that CP_INTC register is write-only. Writing 1 alone sends a pulse**;** there is no need to write 0 to close the pulse.

SEQ_SYNC_STATE register contains sync status bits from the coprocessors, DMA, sequencer, DSP, and ARM. The status bit is set when a strobe pulse is received from the corresponding module. It is cleared by writing a 1 to the bit position. The status bits read back to ARM, DSP, and sequencer are pre-mask, that is, not affected by the setting of the SYNC_MSK registers.

SEQ_SYNC_MASK register controls which of the sync status bits get ANDed / ORed to produce the 1-bit sync signal sent to the sequencer.

The DSP_SYNC_STATE and DSP_SYNC_MASK behaves similar to SEQ_SYNC_STATE and SEQ_SYNC_MASK, but are intended for DSP to program, and the generated sync signal goes to DSP interrupt.

The ARM_SYNC_STATE and ARM_SYNC_MASK behave similar to SEQ_SYNC_STATE and SEQ_SYNC_MASK, but are intended for ARM to program, and the generated sync signal goes to ARM interrupt.

3.7 Break-Point Function

Sequencer breakpoint feature is implemented by the BRKPT_TRG and BRKPT_CLR registers. These provides a way for inserting break points in the sequencer code, and have DSP to come in and examine coprocessors, image buffers, and so on upon these break points. Sequencer/DSP interaction for break points is shown in the Figure 7.

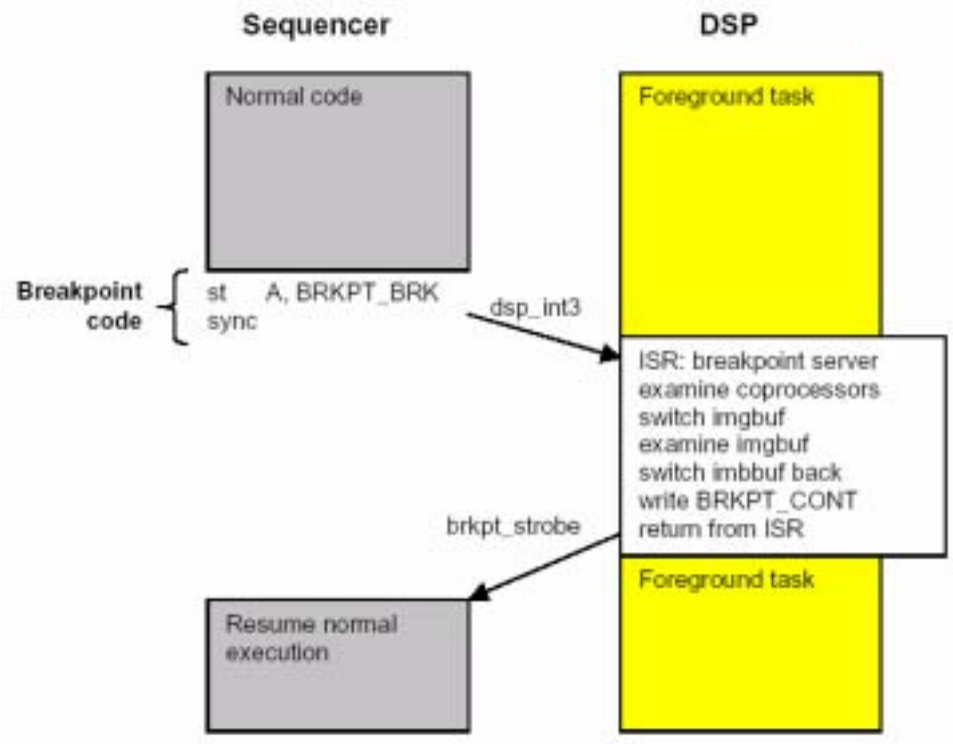


Figure 7: ***Sequencer/DSP interaction during break point***

The mechanism to trigger the break is to write any value to the corresponding registers. This simplifies the breakpoint code on the sequencer by avoiding saving and restoring registers, and thereby reduces memory cost per breakpoint.

3.8 Coprocessor DMA (COP DMA) Registers

Note: The register address given here are in the DSP IO space. These registers appear in the address space of ARM and Sequencer also. Refer Table 11 for the corresponding register address locations in ARM and Sequencer address spaces.

DSP	SEQ	ARM	Register	Description
0000	F000	0009:E000	SDEM_ADDRH	SDRAM Address High Register
0001	F001	0009:E002	SDEM_ADDRL	SDRAM Address Low Register
0002	F002	0009:E004	SDEM_LOFST	SDRAM Offset Register
0003	F003	0009:E006	BUF_ADDR	Buffer Address Register
0004	F004	0009:E008	BUF_LOFST	Buffer Offset Register
0005	F005	0009:E00A	DMA_XNUM	DMA X Direction Transfer Number Register
0006	F006	0009:E00C	DMA_YNUM	DMA Y Direction Transfer Number Register
0007	F007	0009:E00E	DMA_CTRL	DMA Control Register
0008	F008	0009:E010	BUF_MUX0	Buffer multiplex 0 register
0009	F009	0009:E012	BUF_MUX1	Buffer multiplex 1 register
000A	F00A	0009:E014	IMG_MODE	Image mode register

3.8.1 SDEM_ADDRH

SDRAM / EMIF Address High Register

SDEM_ADDRH		DSP	IO	0000	offset:	0x00									default:	0x0000
		SEQ		F000												
		ARM		0009:E000												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	ADDRH[10]	ADDRH[9]	ADDRH[8]	ADDRH[7]	ADDRH[6]	ADDRH[5]	ADDRH[4]	ADDRH[3]	ADDRH[2]	ADDRH[1]	ADDRH[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	ADDRH	0	R/W	bit[26:16] of SDRAM/EMIF offset address

3.8.2 SDEM_ADDRL

SDRAM / EMIF Address Low Register

SDEM_ADDRL		DSP	IO	0001	offset:	0x01									default:	0x0000
		SEQ		F001												
		ARM		0009:E002												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	ADDRL[15]	ADDRL[14]	ADDRL[13]	ADDRL[12]	ADDRL[11]	ADDRL[10]	ADDRL[9]	ADDRL[8]	ADDRL[7]	ADDRL[6]	ADDRL[5]	ADDRL[4]	ADDRL[3]	ADDRL[2]	ADDRL[1]	ADDRL[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	ADDRL	0	R/W	bit[15:0] of SDRAM/EMIF offset address (unit = DSP word) *For efficient DMA transfers set SDEM_ADDRL[3:0] = 0

3.8.3 SDEM_LOFST

SDRAM / EMIF Offset Register

SDEM_LOFST		DSP	IO	0002	offset:	0x02									default:	0x0000
		SEQ		F002												
		ARM		0009:E004												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	SLOFST[15]	SLOFST[14]	SLOFST[13]	SLOFST[12]	SLOFST[11]	SLOFST[10]	SLOFST[9]	SLOFST[8]	SLOFST[7]	SLOFST[6]	SLOFST[5]	SLOFST[4]	SLOFST[3]	SLOFST[2]	SLOFST[1]	SLOFST[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	SLOFST	0	R/W	SDRAM/EMIF address offset (unit = DSP word). Offset added to the starting SDRAM address after each DMA_XNUM words are transferred. This register must be set even value.

3.8.4 BUF_ADDR

Buffer Address Register (Image buffer, iMX, VLCD, SEQ memory offset address)

BUF_ADDR		DSP	IO	0003	offset:	0x03									default:	0x0000
		SEQ		F003												
		ARM		0009:E006												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	ADDR[12]	ADDR[11]	ADDR[10]	ADDR[9]	ADDR[8]	ADDR[7]	ADDR[6]	ADDR[5]	ADDR[4]	ADDR[3]	ADDR[2]	ADDR[1]	ADDR[0]
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:13	RSV			Reserved
12:0	ADDR	0	R/W	Start address of buffer (unit = DSP word) Note: For DMA from / to iMX Cmd, SEQ and VLCD memories 32 bit alignment (even) is required. For Image buffers and iMX Coefficient memory only 16-bit alignment (odd) is enough.

3.8.5 BUF_LOFST

Buffer Offset Register

BUF_LOFST		DSP	IO	0004	offset:	0x04									default:	0x0000
		SEQ		F004												
		ARM		0009:E008												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	BLOFST[12]	BLOFST[11]	BLOFST[10]	BLOFST[9]	BLOFST[8]	BLOFST[7]	BLOFST[6]	BLOFST[5]	BLOFST[4]	BLOFST[3]	BLOFST[2]	BLOFST[1]	BLOFST[0]
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:13	RSV			Reserved
12:0	BLOFST	0	R/W	Buffer address offset (unit = DSP word) Note: Offset added to the starting Buffer address after each DMA_XNUM words are transferred.

3.8.6 DMA_XNUM

DMA X Direction Transfer Number Register

DMA_XNUM		DSP	IO	0005	offset:	0x05									default:	0x0000
		SEQ		F005												
		ARM		0009:E00A												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	XNUM [12]	XNUM [11]	XNUM [10]	XNUM [9]	XNUM [8]	XNUM [7]	XNUM [6]	XNUM [5]	XNUM [4]	XNUM [3]	XNUM [2]	XNUM [1]	XNUM [0]
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:13	RSV			Reserved
12:0	XNUM	0	R/W	Number of words in each row to transfer in X-direction (unit = DSP word) 1 - 8191

3.8.7 DMA_YNUM

DMA Y Direction Transfer Number Register

DMA_YNUM		DSP	IO	0006	offset:	0x06									default:	0x0000
		SEQ		F006												
		ARM		0009:E00C												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	YNUM [12]	YNUM [11]	YNUM [10]	YNUM [9]	YNUM [8]	YNUM [7]	YNUM [6]	YNUM [5]	YNUM [4]	YNUM [3]	YNUM [2]	YNUM [1]	YNUM [0]
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:13	RSV		R/W	Reserved
12:0	YNUM	0	R/W	Number of rows to transfer in Y-direction 1 - 8191

3.8.8 DMA_CTRL

DMA Control Register

DMA_CTRL		DSP	IO	0007	offset:	0x07									default:	0x0000
		SEQ		F007												
		ARM		0009:E00E												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	DESTSEL	RSV	RSV	RSV	RSV	RSV	DMAINT	DMAON	DIR
R/W	-	-	-	-	-	-	-	R/W	-	-	-	-	-	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	0	-	-	-	-	-	0	0	0

Bit	Name	Reset Value	R/W	Function
15:9	RSV			Reserved
8	DESTSEL	0	R/W	DMA Destination Select » 0: SDRAM « » 1: External Memory IF «
7:3	RSV			Reserved
2	DMAINT	0	R/W	DMA Request » 0: Interrupt disable « » 1: Interrupt enable «
1	DMAON	0	R/W	DMA Request » 1: DMA go « Writing "1" initiates the DMA transfer is in progress. The request will be cleared automaticlly when the DMA transfer completes.
0	DIR	0	R/W	Read or Write » 0: Buffer to SDRAM/External Memory « » 1: SDRAM/External Memory to Buffer «

3.8.9 BUF_MUX0

Image buffer multiplex Control Register

BUF_MUX0		DSP	IO	0008	offset:	0x08									default:	0x0000
		SEQ		F008												
		ARM		0009:E010												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RSV	RSV	IMGBUFSEL[1]	IMGBUFSEL[0]	IMGBUFC[3]	IMGBUFC[2]	IMGBUFC[1]	IMGBUFC[0]	IMGBUFB[3]	IMGBUFB[2]	IMGBUFB[1]	IMGBUFB[0]	IMGBUFA[3]	IMGBUFA[2]	IMGBUFA[1]	IMGBUFA[0]
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:14	RSV			Reserved
13:12	IMGBUFSEL	0	R/W	Image Buffer Switch These bits select which Image Buffer is visible in the common Image Buffer memory region. Since the DSP can only access this common region, use these bits to select which buffer will be visible to the DSP regardless of the Image Buffers enabled to the co-processor bus; i.e., available to ARM and/or Sequencer. » 00: Normal Mode « » 01: ImgBuf-A visible in common memory if on CP bus « » 10: ImgBuf-B visible in common memory if on CP bus « » 11: ImgBuf-C visible in common memory if on CP bus « NOTE - 00: Accesses to the common region are OR-ed if multiple Image Buffers are connected to co-processor bus. Note: - 01,10,11: These settings are valid only if the corresponding Image Buffer is connected to co-processor bus
11:8	IMGBUFC	0	R/W	Switch for Image Buffer C » 0000: Co-processor bus « » 0001: iMX (Normal mode, mapped at byte addr 0x0000) « » 0010: DCT « » 0011: DMA « » 0100: VLCD_ibuf0 « » 0101: VLCD_ibuf1 « » 0110: no connection « » 0111: iMX (Sequential buffer mode) « Maps Image Buffer C into iMX address space at address 0x2000 - 0x3FFF (byte address) others: reserved - do not use
7:4	IMGBUFB	0	R/W	Switch for Image Buffer B » 0000: Co-processor bus « » 0001: iMX (Normal mode, mapped at byte addr 0x0000) « » 0010: DCT « » 0011: DMA « » 0010: VLCD_ibuf0 « » 0101: VLCD_ibuf1 « » 0110: no connection « others: reserved - do not use
3:0	IMGBUFA	0	R/W	Switch for Image Buffer A » 0000: Co-processor bus « » 0001: iMX (Normal mode, mapped at byte addr 0x0000) « » 0010: DCT « » 0011: DMA « » 0010: VLCD_ibuf0 « » 0101: VLCD_ibuf1 « » 0110: no connection « others: reserved - do not use

3.8.10 BUF_MUX1

Co-processor memory-multiplex Control Register

BUF_MUX1		DSP	IO	0009	offset:	0x09									default:	0x0000
		SEQ		F009												
		ARM		0009:E012												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	SEQ[1]	SEQ[0]	VLCDHUF[1]	VLCDHUF[0]	LCDQUANT	LCDQUANT	IMXCMD[1]	IMXCMD[0]	IMXCOEFF[1]	IMXCOEFF[0
R/W	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:10	RSV			Reserved
9:8	SEQ	0	R/W	Switch for sequencer memory » 00: Coprocessor bus « » 01: COP DMA « » 10: reserved « » 11: no operation «
7:6	VLCDHUF	0	R/W	Switch for VLCD huffman memory » 00: Coprocessor bus « » 01: VLCD « » 10: COP DMA « » 11: no operation «
5:4	VLCDQUANT	0	R/W	Switch for VLCD Quantization memory » 00: Coprocessor bus « » 01: VLCD « » 10: DMA « » 11: no operation «
3:2	IMXCMD	0	R/W	Switch for iMX command memory » 00: Coprocessor bus « » 01: iMX « » 10: COP DMA « » 11: no operation «
1:0	IMXCOEFF	0	R/W	Switch for iMX coefficient memory » 00: Coprocessor bus « » 01: iMX « » 10: COP DMA « » 11: no operation «

3.8.11 IMG_MODE

Image Buffer Mode Register

IMG_MODE		DSP	IO	000A	offset:	0x0A									default:	0x0000
		SEQ		F00A												
		ARM		0009:E014												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	IMXCMD	SEQMEM	SWAPRD	SWAPWR	RSV	WORD	SHIFT[1]	SHIFT[0]
R/W	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	0	0	0	0	-	0	0	0

Bit	Name	Reset Value	R/W	Function
15:8	RSV			Reserved
7	IMXCMD	0	R/W	Page of iMX command memory visible to DSP (paged region) » 0: lower 2K words « » 1: upper 2K words «
6	SEQMEM	0	R/W	Page of sequencer memory visible to DSP (paged region) » 0: lower 2K words « » 1: upper 2K words «
5	SWAPRD	0	R/W	Byte SWAP for SDRAM/External Memory Reading (via DMA) » 0: No SWAP « » 1: Byte SWAP «
4	SWAPWR	0	R/W	Byte SWAP for SDRAM/External Memory Writing (via DMA) » 0: No SWAP « » 1: Byte SWAP «
3	RSV			Reserved
2	WORD	0	R/W	Byte data to word data transform » 0: No transformation « » 1: Byte data in SDRAM/ExtMem stored as word data. « NOTE: - Number of words in the X-direction represents the source size, and the data written to buffers A/B/C/Coeff memory will be twice that size. - This function applies when buffers A, B, C or IMX coefficient memory are the destination of the transfer.
1:0	SHIFT	0	R/W	(DMA) Data Bus Shift Down » 00: 0 bit « » 01: 4 bit « » 10: 6 bit « » 11: 8 bit « Only applies on transfer from SDRAM/EM to Image Buffers A/B/C

3.9 COP Interrupt / Clock Control Register Map (INTCLK)

DSP	SEQ	ARM	Register	Description
0280	F280	0009:E500	CP_INTC	Coprocessor Interrupt controller
0281	F281	0009:E502	CP_CLKC	Coprocessor Clock controller
0282	F282	0009:E504	SEQ_SYNC_STATE	State of Sequencer sync
0283	F283	0009:E506	SEQ_SYNC_MASK	Mask of Sequencer sync
0284	F284	0009:E508	DSP_SYNC_STATE	State of DSP sync
0285	F285	0009:E50A	DSP_SYNC_MASK	Mask of DSP sync
0286	F286	0009:E50C	ARM_SYNC_STATE	State of ARM sync
0287	F287	0009:E50E	ARM_SYNC_MASK	Mask of ARM sync
0288	F288	0009:E510	SEQ_INT_STATE	State of Sequencer interrupts
0289	F289	0009:E512	SEQ_INT_MASK	Mask of Sequencer interrupts
028A	F28A	0009:E514	SEQ_INT_B4MASK	State of Sequencer interrupts before mask
028B	F28B	0009:E516	SEQ_INT_VECTOR	Vectors for Sequencer interrupts
028C	F28C	0009:E518	BRKPT_TRG	Trigger Sequencer breakpoint
028D	F28D	0009:E51A	BRKPT_CLR	Clear Sequencer breakpoint

3.10 DSP Interrupt / Clock Control Registers

3.10.1 CP_INTC

Co-processor Interrupt Control Register

CP_INTC		DSP	IO	0280	offset:	0x00									default:	0x0000
		SEQ		F280												
		ARM		0009:E500												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	DSPINT3	DSPINT2	DSPINT1	ARMINT	ARMSTRB	SEQSTRB	DSPSTRB
R/W	-	-	-	-	-	-	-	-	-	W	W	W	W	W	W	W
Default	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:7	RSV			Reserved
6	DSPINT3	0	W	Writing a 1 to this field will send INT3 on DSP
5	DSPINT2	0	W	Writing a 1 to this field will send INT2 on DSP
4	DSPINT1	0	W	Writing a 1 to this field will send INT1 on DSP
3	ARMINT	0	W	Writing a 1 to this field will send an interrupt on ARM
2	ARMSTRB	0	W	Writing a 1 to this field will send a strobe pulse on arm_strobe internal signal
1	SEQSTRB	0	W	Writing a 1 to this field will send a strobe pulse on seq_strobe internal signal
0	DSPSTRB	0	W	Writing a 1 to this field will send a strobe pulse on dsp_strobe internal signal

3.10.2 CP_CLKC

Co-processor Clock Control Register

CP_CLKC		DSP	IO	0281	offset:	0x01									default:	0x0000
		SEQ		F281												
		ARM		0009:E502												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	DCT	RSV	SEQ	VLCD	IMX	IMGBUF
R/W	-	-	-	-	-	-	-	-	-	-	R/W	-	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	-	0	-	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:6	RSV			Reserved
5	DCT	0	R/W	DCT Clock Enable » 0: Disable DCT clock « » 1: Enable DCT clock «
4	RSV	0		Reserved
3	SEQ	0	R/W	Sequencer Clock Enable » 0: Disable sequencer clock « » 1: Enable sequencer clock «
2	VLCD	0	R/W	VLCD Clock Enable » 0: Disable VLCD clock « » 1: Enable VLCD clock «
1	IMX	0	R/W	IMX Clock Enable » 0: Disable iMX clock « » 1: Enable iMX clock «
0	IMGBUF	0	R/W	Image Buffer Clock Enable » 0: Disable IMGBUF clock « » 1: Enable IMGBUF clock «

3.10.3 SEQ_SYNC_STATE

Sequencer sync state Register

SEQ_SYNC_STATE		DSP	IO	0282	offset:	0x02									default:	0x0000
		SEQ		F282												
		ARM		0009:E504												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	DSPSYNC	DCTSYNC	ARMSYNC	SEQSYNC	VLCDSYNC	IMXSYNC	DMASYNC
R/W	-	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:7	RSV			Reserved
6	DSPSYNC	0	R/W	DSP sync state: » 0: False « » 1: True « Writing a 1 will clear this field
5	DCTSYNC	0	R/W	DCT sync state: » 0: False « » 1: True « Writing a 1 will clear this field
4	ARMSYNC	0	R/W	ARM sync state: » 0: False « » 1: True « Writing a 1 will clear this field
3	SEQSYNC	0	R/W	Sequencer sync state: » 0: False « » 1: True « Writing a 1 will clear this field
2	VLCDSYNC	0	R/W	VLCD sync state: » 0: False « » 1: True « Writing a 1 will clear this field
1	IMXSYNC	0	R/W	IMX sync state: » 0: False « » 1: True « Writing a 1 will clear this field
0	DMASYNC	0	R/W	DMA sync state: » 0: False « » 1: True « Writing a 1 will clear this field

3.10.4 SEQ_SYNC_MASK

Sequencer sync mask Register

SEQ_SYNC_MASK		DSP	IO	0283	offset:	0x03									default:	0x0000
		SEQ		F283												
		ARM		0009:E506												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	ANDOR	DSPMASK	DCTMASK	ARMMASK	SEQMASK	VLCDMASK	IMXMASK	DMAMASK
R/W	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:8	RSV			Reserved
7	ANDOR	0	R/W	AND / OR mode of Sequencer sync: » 0: AND « » 1: OR «
6	DSPMASK	0	R/W	DSP Mask state: » 0: masked « » 1: enabled «
5	DCTMASK	0	R/W	DCT Mask state: » 0: masked « » 1: enabled «
4	ARMMASK	0	R/W	ARM Mask state: » 0: masked « » 1: enabled «
3	SEQMASK	0	R/W	Sequencer Mask state: » 0: masked « » 1: enabled «
2	VLCDMASK	0	R/W	VLCD Mask state: » 0: masked « » 1: enabled «
1	IMXMASK	0	R/W	iMX Mask state: » 0: masked « » 1: enabled «
0	DMAMASK	0	R/W	DMA Mask state: » 0: masked « » 1: enabled «

3.10.5 DSP_SYNC_STATE

DSP Sync state Register

DSP_SYNC_STATE		DSP	IO	0284	offset:	0x04									default:	0x0000
		SEQ		F284												
		ARM		0009:E508												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	DSPSYNC	DCTSYNC	ARMSYNC	SEQSYNC	VLCDSYNC	IMXSYNC	DMASYNC
R/W	-	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0

Bit	Name	eset Valu	R/W	Function
15:7	RSV			Reserved
6	DSPSYNC	0	R/W	DSP sync state: » 0: False « » 1: True « Writing a 1 will clear this field
5	DCTSYNC	0	R/W	DCT sync state: » 0: False « » 1: True « Writing a 1 will clear this field
4	ARMSYNC	0	R/W	ARM sync state: » 0: False « » 1: True « Writing a 1 will clear this field
3	SEQSYNC	0	R/W	Sequencer sync state: » 0: False « » 1: True « Writing a 1 will clear this field
2	VLCDSYNC	0	R/W	VLCD sync state: » 0: False « » 1: True « Writing a 1 will clear this field
1	IMXSYNC	0	R/W	IMX sync state: » 0: False « » 1: True « Writing a 1 will clear this field
0	DMASYNC	0	R/W	DMA sync state: » 0: False « » 1: True « Writing a 1 will clear this field

3.10.6 DSP_SYNC_MASK

DSP Sync mask Register

DSP_SYNC_MASK		DSP	IO	0285	offset:	0x05									default:	0x0000
		SEQ		F285												
		ARM		0009:E50A												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	ANDOR	DSPMASK	DCTMASK	ARMMASK	SEQMASK	VLCDMASK	IMXMASK	DMAMASK
R/W	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0

Bit	Name	eset Valu	R/W	Function
15:8	RSV			Reserved
7	ANDOR	0	R/W	AND / OR mode of DSP Sync: » 0: AND « » 1: OR «
6	DSPMASK	0	R/W	DSP Mask state: » 0: masked « » 1: enabled «
5	DCTMASK	0	R/W	DCT Mask state: » 0: masked « » 1: enabled «
4	ARMMASK	0	R/W	ARM Mask state: » 0: masked « » 1: enabled «
3	SEQMASK	0	R/W	Sequencer Mask state: » 0: masked « » 1: enabled «
2	VLCDMASK	0	R/W	VLCD Mask state: » 0: masked « » 1: enabled «
1	IMXMASK	0	R/W	iMX Mask state: » 0: masked « » 1: enabled «
0	DMAMASK	0	R/W	DMA Mask state: » 0: masked « » 1: enabled «

3.10.7 ARM_SYNC_STATE

ARM sync state Register

ARM_SYNC_STATE	DSP	IO	0286	offset:	0x06									default:	0x0000
	SEQ		F286												
	ARM		0009:E50C												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	DSPSYNC	DCTSYNC	ARMSYNC	SEQSYNC	VLCDSYNC	IMXSYNC	DMASYNC
R/W	-	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0

Bit	Name	eset Valu	R/W	Function
15:7	RSV			Reserved
6	DSPSYNC	0	R/W	DSP sync state: » 0: False « » 1: True « Writing a 1 will clear this field
5	DCTSYNC	0	R/W	DCT sync state: » 0: False « » 1: True « Writing a 1 will clear this field
4	ARMSYNC	0	R/W	ARM sync state: » 0: False « » 1: True « Writing a 1 will clear this field
3	SEQSYNC	0	R/W	Sequencer sync state: » 0: False « » 1: True « Writing a 1 will clear this field
2	VLCDSYNC	0	R/W	VLCD sync state: » 0: False « » 1: True « Writing a 1 will clear this field
1	IMXSYNC	0	R/W	IMX sync state: » 0: False « » 1: True « Writing a 1 will clear this field
0	DMASYNC	0	R/W	DMA sync state: » 0: False « » 1: True « Writing a 1 will clear this field

3.10.8 ARM_SYNC_MASK

ARM sync mask Register

ARM_SYNC_MASK		DSP	IO	0287	offset:	0x07									default:	0x0000
		SEQ		F287												
		ARM		0009:E50E												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	ANDOR	DSPMASK	DCTMASK	ARMMASK	SEQMASK	VLCDMASK	IMXMASK	DMAMASK
R/W	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0

Bit	Name	eset Valu	R/W	Function
15:8	RSV			Reserved
7	ANDOR	0	R/W	AND / OR mode of ARM Sync: » 0: AND « » 1: OR «
6	DSPMASK	0	R/W	DSP Mask state: » 0: masked « » 1: enabled «
5	DCTMASK	0	R/W	DCT Mask state: » 0: masked « » 1: enabled «
4	ARMMASK	0	R/W	ARM Mask state: » 0: masked « » 1: enabled «
3	SEQMASK	0	R/W	Sequencer Mask state: » 0: masked « » 1: enabled «
2	VLCDMASK	0	R/W	VLCD Mask state: » 0: masked « » 1: enabled «
1	IMXMASK	0	R/W	iMX Mask state: » 0: masked « » 1: enabled «
0	DMAMASK	0	R/W	DMA Mask state: » 0: masked « » 1: enabled «

3.10.9 SEQ_INT_STATE

Sequencer interrupt state Register

SEQ_INT_STATE		DSP	IO	0288	offset:	0x08									default:	0x0000
		SEQ		F288												
		ARM		0009:E510												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	DSPINT	DCTINT	ARMINT	RSV	VLCDINT	IMXINT	DMAINT
R/W	-	-	-	-	-	-	-	-	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	0	0	0	-	0	0	0

Bit	Name	eset Valu	R/W	Function
15:7	RSV			Reserved
6	DSPINT	0	R/W	DSP interrupt state: » 0: False « » 1: True « Writing a 1 will clear this field
5	DCTINT	0	R/W	DCT interrupt state: » 0: False « » 1: True « Writing a 1 will clear this field
4	ARMINT	0	R/W	ARM interrupt state: » 0: False « » 1: True « Writing a 1 will clear this field
3	RSV			Reserved
2	VLCDINT	0	R/W	VLCD interrupt state: » 0: False « » 1: True « Writing a 1 will clear this field
1	IMXINT	0	R/W	IMX interrupt state: » 0: False « » 1: True « Writing a 1 will clear this field
0	DMAINT	0	R/W	DMA interrupt state: » 0: False « » 1: True « Writing a 1 will clear this field

3.10.10 SEQ_INT_MASK

Sequencer interrupt mask Register

SEQ_INT_MASK		DSP	IO	0289	offset:	0x09									default:	0x0000
		SEQ		F289												
		ARM		0009:E512												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	DSPMASK	DCTMASK	ARMMASL	RSV	VLCDMASK	IMXMASK	DMAMASK
R/W	-	-	-	-	-	-	-	-	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	0	0	0	-	0	0	0

Bit	Name	eset Valu	R/W	Function
15:7	RSV			Reserved
6	DSPMASK	0	R/W	DSP interrupt mask: » 0: masked « » 1: enabled «
5	DCTMASK	0	R/W	DCT interrupt mask: » 0: masked « » 1: enabled «
4	ARMMASL	0	R/W	ARM interrupt mask: » 0: masked « » 1: enabled «
3	RSV			Reserved
2	VLCDMASK	0	R/W	VLCD interrupt mask: » 0: masked « » 1: enabled «
1	IMXMASK	0	R/W	iMX interrupt mask: » 0: masked « » 1: enabled «
0	DMAMASK	0	R/W	DMA interrupt mask: » 0: masked « » 1: enabled «

3.10.11 SEQ_INT_B4MASK

Sequencer interrupts state before mask Register

SEQ_INT_B4MASK	DSP	IO	028A	offset:	0x0A									default:	0x0000
	SEQ		F28A												
	ARM		0009:E514												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	DSPINT	DCTINT	ARMINT	RSV	VLCDINT	IMXINT	DMAINT
R/W	-	-	-	-	-	-	-	-	-	R	R	R	-	R	R	R
Default	-	-	-	-	-	-	-	-	-	0	0	0	-	0	0	0

Bit	Name	eset Valu	R/W	Function
15:7	RSV			Reserved
6	DSPINT	0	R	DSP interrupt state: » 0: False « » 1: True «
5	DCTINT	0	R	DCT interrupt state: » 0: False « » 1: True «
4	ARMINT	0	R	ARM interrupt state: » 0: False « » 1: True «
3	RSV		R	Reserved
2	VLCDINT	0	R	VLCD interrupt state: » 0: False « » 1: True «
1	IMXINT	0	R	IMX interrupt state: » 0: False « » 1: True «
0	DMAINT	0	R	DMA interrupt state: » 0: False « » 1: True «

3.10.12 SEQ_INT_VECTOR

Sequencer interrupt vector Register

SEQ_INT_VECTOR		DSP	IO	028B	offset:	0x0B									default:	0x0000
		SEQ		F28B												
		ARM		0009:E516												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	VSPACE[1]	VSPACE[0]	BASE_PG[7	BASE_PG[6	BASE_PG[5	BASE_PG[4	BASE_PG[3	BASE_PG[2	BASE_PG[1	BASE_PG[0
R/W	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0

Bit	Name	eset Valu	R/W	Function
15:10	RSV			Reserved
9:8	VSPACE	0	R/W	Interrupt vector spacing in bytes (4 << VSPACE)
7:0	VBASE_PG	0	R/W	Vector base address: VBASE = Sequencer_memory + (VBASE_PG << 8) DMA vector = VBASE iMX vector = VBASE + (4 << VSPACE) VLCD vector = VBASE + 2*(4 << VSPACE) SEQ vector = VBASE + 3*(4 << VSPACE) ARM vector = VBASE + 4*(4 << VSPACE) DCT vector = VBASE + 5*(4 << VSPACE) DSP vector = VBASE + 6*(4 << VSPACE)

3.10.13 BRKPT_TRG

Break point trigger register

BRKPT_TRG		DSP	IO	028C	offset:	0x0C									default:	0x0000
		SEQ		F28C												
		ARM		0009:E518												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	TRG[15]	TRG[14]	TRG[13]	TRG[12]	TRG[11]	TRG[10]	TRG[9]	TRG[8]	TRG[7]	TRG[6]	TRG[5]	TRG[4]	TRG[3]	TRG[2]	TRG[1]	TRG[0]
R/W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	eset Valu	R/W	Function
15:0	TRG	0	W	Write any value to trigger a sequencer breakpoint Note: 1) Sends an INT3 interrupt to DSP and 2) Sets brkpt-state flip-flop, which pulle sync_seq low

3.10.14 BRKPT_CLR

Break point clear register

BRKPT_CLR		DSP	IO	028D	offset:	0x0D									default:	0x0000
		SEQ		F28D												
		ARM		0009:E51A												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	CLR[15]	CLR[14]	CLR[13]	CLR[12]	CLR[11]	CLR[10]	CLR[9]	CLR[8]	CLR[7]	CLR[6]	CLR[5]	CLR[4]	CLR[3]	CLR[2]	CLR[1]	CLR[0]
R/W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	eset Valu	R/W	Function
15:0	CLR	0	W	Write any value to clear a break point Note: 1) Clears the brkpt_state flip-flop and 2) Releases the pulldown on sync_seq so the Sequencer can resume (if being stalled)

4 Imaging Extension Engine (iMX)

4.1 Introduction

Details describing the internal architecture of the iMX co-processor are generally not available to the customers. The description of the iMX given in this chapter is very limited. To use the iMX, you should use the '**C'** callable iMX API's provided by **Texas Instruments (TI)**. Refer to the documentation provided with the "**C**" callable API library for more information on these APIs.

iMX, image processing extension, is a parallel MAC engine with flexible control and memory interface for extending image processing performance of TI's programmable DSPs. DM320 iMX datapath consists of 8 parallel multiply-accumulate units, and in most of the above computation, can sustain 8*180 = 1,440 million MACs/sec processing rate (at 180 MHz).

The iMX is a parallel MAC engine for increasing DSP image processing performance. The iMX hardware module consists of a flexible control and memory interface and 8 MACs. The iMX works well in a shared memory configuration with the DSP, so that regular block-based image processing tasks can be accelerated by the iMX, and the DSP can handle irregular processing as well as task-level control.

The iMX can be used to increase the computation speed of many computations, such as the following computations:

- 2-D FIR filtering
- CFA interpolation
- Color space conversion
- Chroma down-sampling
- Edge enhancement
- False color suppression
- DCT/ IDCT
- Table lookup
- Motion estimation
- Generic matrix multiplication
- Generic linear transform

iMX architecture can be categorized as a memory-to-memory vector DSP processor. Programming iMX on the hardware level involves specifying of many hardware parameters on block-processing tasks, but does not involve cycle-by-cycle instructions. One task can keep iMX busy for a few cycles, or many thousands of cycles, but it's much more efficient to process large blocks of data.

Each task with one set of hardware parameters is called an iMX command. The parameters control how long the command runs, the nested loops, operand access, operation being performed, how often we write-back versus accumulate, and so on.

4.1.1 Features

DM320 iMX is an 8-MAC version with the following features

- Logical AND/OR/XOR function to support extracting lower bits, for post-lookup interpolation
- MIN-ID and MAX-ID functions. Basically preserves the pointer value to which the minimal value resides. This helps motion estimation.
- New “CALL” command which enables re-use of a iMX command module
- Sequential access to image buffer A and B for non concurrent process

4.1.2 iMX Structure

As shown in Figure 8, the iMX consists of 8 parallel multiply-accumulate units (MACs). For most of the calculations mentioned above, the iMX processes 1440(180x8) million MACs/sec (NOTE: iMX can run upto clock frequency of about 180Mhz. Refer datasheet for exact maximum frequency). Each MAC can perform multiplication, addition, subtraction, absolute value, and accumulate.

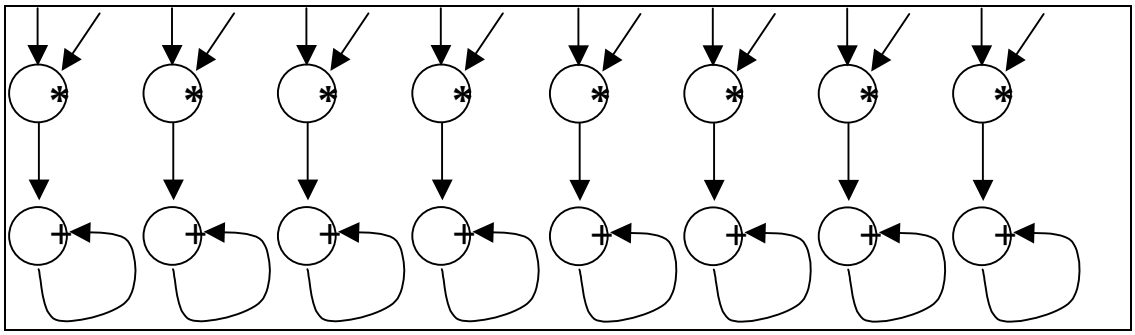


Figure 8: iMX Structure- Eight parallel MACs

4.1.3 DSP/iMX Shared Memory Interface

DM320 Coprocessor Subsystem has an ARM9, C54x DSP and a Sequencer communicating with iMX via control/status registers and shared memory blocks. The following block diagram shows the connection of coprocessor modules and memory blocks to the ARM, DSP and the Sequencer. The Image Buffers A, B, and C are for generic data storage. The other memory blocks have specific functions normally, but the iMX coefficient memory can also be used for intermediate data storage.

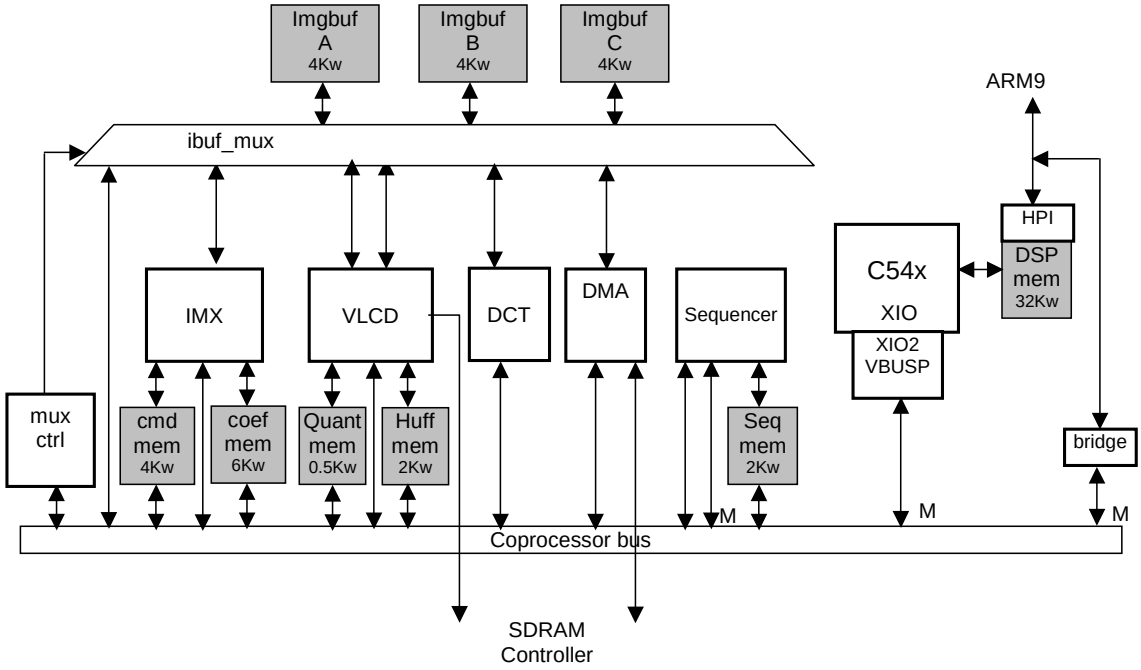


Figure 9: Coprocessor Subsystem Block Diagram

To simplify buffer swapping programming, shared memory interface is designed such that iMX does not access multiple image buffers at the same time. To iMX all three buffers are on the same memory space, and ARM/DSP/Sequencer controls which of the three are accessible from iMX. iMX can access bytes or shorts (16-bit words), and its programming model uses byte addressing. iMX sees the range 0x8000 ~ 0xAFFF as the coefficient memory.

For the image buffers iMX has two configuration options. In the normal mode, one of three buffers A/B/C is switched to iMX, and iMX sees it in the lowest 8K bytes (0 ~ 0x1FFF). In the sequential buffer mode, one of two buffers A/B is switched to iMX in the lowest 8K bytes, 0 ~ 0x1FFF, and iMX also sees buffer C in the next 8K bytes, 0x2000 ~ 0x3FFF. The following table summarizes the image buffer and coefficient memory address map as shown in Table 16.

Address range (byte addr)	Normal Mode	Sequential Mode
0x0000 ~ 0x1FFF	Image buffer A, B or C	Image buffer A or B
0x2000 ~ 0x3FFF	Reserved	Image buffer C
0x4000 ~ 0x7FFF	Reserved	
0x8000 ~ 0xAFFF	Coefficient memory	
0xB000 ~ 0xFFFF	Reserved	

Table 16: iMX Image Buffer and Coefficient Memory Address Map

The iMX command memory is in a separate memory space, 0x0000 - 0x0FFF (Word address). Only the Write command can access the command memory.

iMX commands refer to data pointer, coefficient pointer, and output pointer. Each of these pointers can point to image buffer or coefficient memory, without any restriction. The only restriction is that data object (or array) cannot be allowed to straddle across image buffer and coefficient memory. This programming model provides good level of flexibility to optimize iMX code for performance and storage.

There is performance penalty for memory contention. For example, point-by-point addition of two arrays in the image buffer and writing the result back to the image buffer will see a 3x slow down, compared to adding a constant (which requires no steady-state access no matter where it is) to an array in image buffer and writing the result to coefficient memory. The flexibility in data/coefficient/output object allocation helps minimize memory contention and improve overall performance.

Table 17 Shows the shared memory configuration in the ARM, DSP, Sequencer and iMX memory maps. The iMX co-processor controls iMX coefficient memory, iMX command memory and Image buffer A/B/C (in the normal mode). All shared memory blocks shown are single-ported, and the ARM/DSP/Sequencer must set which module has access to which shared memory block (no cycle-by-cycle arbitration is implemented) by setting the register MINMAX_IDADDR.

In a typical application, the DSP places input data in Image buffer A/B/C and iMX commands in command memory and filter coefficients, DCT/IDCT cosine constants, and lookup tables in the iMX coefficient memory. Then, ARM/DSP/Sequencer configures the register MINMAX_IDADDR and then initiates iMX execution. The iMX accesses the memory blocks, computes the commands specified in the command buffer, and writes the result back to the data buffer.

The three general-purpose data buffers A, B and C, can be used for efficient concurrent processing. For example, while the iMX is accessing data in buffer A, B can be transferred to SDRAM through the COP DMA controller. When the iMX finishes computation and the result computation result is stored into buffer A, B and C can be swapped so that the iMX can access data in buffer B and the processed data in buffer A can be sent to SDRAM through the COP DMA controller.

To simplify the software that implements buffer swapping, the shared memory interface is designed such that iMX cannot access both buffer A and B at the same time. Buffer A and buffer B have the same memory map with respect to the iMX, and the DSP controls which buffer the iMX can access.

Image Buffers A/B/C, the coefficient buffer, and the command buffer are 16-bit wide from DSP‘s point of view. Also the iMX can access data of type short or byte from Buffers A/B/C, the coefficient buffer, and the command buffer. Short accesses must be aligned to even 16-bit alignments.

Memory	DSP Address (Word)	Arm Address (Byte)	Sequencer Address (Word)	iMX Address (Byte)	Size (Words)
Image BUFA/B/C	0xC000-0xCFFF	0x80000-0x81FFF	0x0000-0x0FFF	0x0000-0x1FFF	4k
iMX coefficient	0xD000-0xE7FF	0x82000-0x84FFF	0x1000-0x27FF	0x8000-0xAFFF	6k
iMX command	0xE800-0xEFFF	0x85000-0x85FFF	0x2800-0x2FFF	0x4000-0x5FFF	4k

Table 17: shared memory configuration

4.1.4 IMX Control and Status Register Interface

The registers listed in this section control the operation of the iMX. Each bit in these registers is described 4.2. The iMX control registers are accessible from DSP's I/O space, ARM and Sequencer address spaces as shown in **Table 18**.

Register Name	ARM address Byte address	DSP address Word address	Sequencer Address Word address
START	0x9E100	0x0080	0xF080
INTR_EN	0x9E102	0x0081	0xF081
BUSY	0x9E104	0x0082	0xF082
CMDPTR	0x9E106	0x0083	0xF083
ABORT	0x9E108	0x0084	0xF084
CLKCNTRL	0x9E10A	0x0085	0xF085
MINMAX_ID	0x9E10C	0x0086	0xF086
MINMAX_IDADDR	0x9E10E	0x0087	0xF087
MINMAX_VALUE	0x9E110	0x0088	0xF088

Table 18: iMX Control register address Map

Writing and address in the START initiates iMX execution. The iMX will begin execution from the address written to START. When the iMX is executing, the bit BUSY is set to '1' in the BUSYregister and it is used to determine the busy status of the iMX.

INTR_EN is used to enable the iMX interrupt. After the iMX completes execution, the iMX will interrupt the ARM, DSP and Sequencer if the bit INTREN

is set to '1' in INTR_EN register. The iMX can send an interrupt to ARM, DSP and Sequencer.

CMDPTR shows the address of the next instruction to be decoded by the iMX. CMDPTR provides feedback on the progress of iMX. However, due to the pipelining nature of iMX hardware, the returned pointer value will be ahead of actual execution by roughly one command.

ABORT is used to abort iMX when handling errors.

MINMAX_ID is used to count the result from the last min-sad/max-sad command.

MINMAX_VALUE is the value from the last min-sad/max-sad command.

Note: Since there is no way to know exactly where the iMX stops, it may be very difficult to restart iMX computation after aborting.

4.2 IMX Control Register Map

DSP	SEQ	ARM	Register	Description
0080	F080	0009:E100	START	iMX Start Register
0081	F081	0009:E102	INTR_EN	iMX INTR Enable Register
0082	F082	0009:E104	BUSY	iMX Busy Register
0083	F083	0009:E106	CMDPTR	iMX Command Pointer Register
0084	F084	0009:E108	ABORT	iMX Abort Register
0085	F085	0009:E10A	CLKCTRL	iMX Clock Control
0086	F086	0009:E10C	MINMAX_IDCNT	Min/Max SAD Block - ID Count
0087	F087	0009:E10E	MINMAX_IDADDR	Min/Max SAD Block - ID Address
0088	F088	0009:E110	MINMAX_VALUE	Min/Max SAD Block - Value

4.2.1 START

iMX Start Register

START		DSP	IO	0080	offset:	0x00									default:	0x0000
		SEQ		F080												
		ARM		0009:E100												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	START[11]	START[10]	START[9]	START[8]	START[7]	START[6]	START[5]	START[4]	START[3]	START[2]	START[1]	START[0]
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	START	0	R/W	Writing a command memory address to this register starts IMX execution from this address.

4.2.2 INTR_EN

iMX INTR Enable Register

INTR_EN		DSP	IO	0081	offset:	0x01									default:	0x0000
		SEQ		F081												
		ARM		0009:E102												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	INTREN
R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R/W
Default	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Bit	Name	Reset Value	R/W	Function
15:1	RSV			Reserved
0	INTREN	0	R/W	iMX Interrupt Enable » 0: Disable interrupt « » 1: Enable interrupt «

4.2.3 BUSY

iMX Busy Register

BUSY		DSP	IO	0082	offset:	0x02									default:	0x0000
		SEQ		F082												
		ARM		0009:E104												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	BUSY
R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R
Default	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Bit	Name	Reset Value	R/W	Function
15:1	RSV			Reserved
0	BUSY	0	R	iMX Idle/Busy Status » 0: iMX idle « » 1: iMX busy «

4.2.4 CMDPTR

iMX Command Pointer Register

CMDPTR		DSP	IO	0083	offset:	0x03									default:	0x0000
		SEQ		F083												
		ARM		0009:E106												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	CMDPTR[15]	CMDPTR[14]	CMDPTR[13]	CMDPTR[12]	CMDPTR[11]	CMDPTR[10]	CMDPTR[9]	CMDPTR[8]	CMDPTR[7]	CMDPTR[6]	CMDPTR[5]	CMDPTR[4]	CMDPTR[3]	CMDPTR[2]	CMDPTR[1]	CMDPTR[0]
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	CMDPTR	0	R	Points to iMX command to be fetched by command decode

4.2.5 ABORT

iMX Abort Register

ABORT		DSP	IO	0084	offset:	0x04									default:	0x0000
		SEQ		F084												
		ARM		0009:E108												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	ABORT
R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R/W
Default	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Bit	Name	Reset Value	R/W	Function
15:1	RSV			Reserved
0	ABORT	0	R/W	Writing 1 to this register stops iMX processing. When iMX returns to idle state, this register is cleared automatically

4.2.6 CLKCNTRL

iMX Clock Control Register

CLKCTRL		DSP	IO	0085	offset:	0x05									default:	0x0000
		SEQ		F085												
		ARM		0009:E10A												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	CLKON
R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R/W
Default	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Bit	Name	Reset Value	R/W	Function
15:1	RSV			Reserved
0	CLKON	0	R/W	Enable/Disable iMX Clock » 0: Clock ON only when working « » 1: Clock always ON «

4.2.7 MINMAX_ID

Minimum / Maximum SAD Block – ID Register

MINMAX_IDCNT		DSP	IO	0086	offset:	0x06									default:	0x0000
		SEQ		F086												
		ARM		0009:E10C												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	IDCNT[15]	IDCNT[14]	IDCNT[13]	IDCNT[12]	IDCNT[11]	IDCNT[10]	IDCNT[9]	IDCNT[8]	IDCNT[7]	IDCNT[6]	IDCNT[5]	IDCNT[4]	IDCNT[3]	IDCNT[2]	IDCNT[1]	IDCNT[0]
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	IDCNT	0	R	Minimum / Maximum ID count from last min/max SAD command

4.2.8 MINMAX_IDADDR

Minimum / Maximum SAD Block – ID ADDR Register

MINMAX_IDADDR		DSP	IO	0087	offset:	0x07									default:	0x0000
		SEQ		F087												
		ARM		0009:E10E												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	IDADDR[15]	IDADDR[14]	IDADDR[13]	IDADDR[12]	IDADDR[11]	IDADDR[10]	IDADDR[9]	IDADDR[8]	IDADDR[7]	IDADDR[6]	IDADDR[5]	IDADDR[4]	IDADDR[3]	IDADDR[2]	IDADDR[1]	IDADDR[0]
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	IDADDR	0	R	Minimum / Maximum ID address from last min/max SAD command

4.2.9 MINMAX_VALUE

Minimum / Maximum value Register

MINMAX_VALUE		DSP	IO	0088	offset:	0x08									default:	0x0000
		SEQ		F088												
		ARM		0009:E110												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	VALUE[15]	VALUE[14]	VALUE[13]	VALUE[12]	VALUE[11]	VALUE[10]	VALUE[9]	VALUE[8]	VALUE[7]	VALUE[6]	VALUE[5]	VALUE[4]	VALUE[3]	VALUE[2]	VALUE[1]	VALUE[0]
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	VALUE	0	R	Minimum / Maximum value from last min/max SAD command

5 Variable Length Coder/Decoder (VLCD)

DM320 VLCD module has QIQ (Q->Quantization, IQ->Inverse Quantization), VLC (Variable Length Coder) and VLD (Variable Length Decoder) functions for JPEG, MPEG1/2/4 standard. Quantizer and Inverse Quantizer (Q/IQ) module of the DM320 DSP subsystem is a high-performance, intelligent and highly programmable hardware accelerator for the quantization and de-quantization of DCT coefficients. The Q/IQ is highly autonomous and is capable of carrying out almost all the steps of quantization and de-quantization process with little or no help from the DSP.

5.1.1 Features

Quantization, zigzag scan for JPEG, MPEG1/2, MPEG4 and H.263

- Quantization, zigzag (also supports linear, alternate Vertical and alternate Horizontal) scan for JPEG, MPEG1/2, MPEG4 and H.263.
- Inverse Quantization, zigzag (also supports linear, alternate Vertical and alternate Horizontal) scan for JPEG, MPEG1/2, MPEG4 and H.263
- Huffman encode for JPEG, MPEG1/2, MPEG4 and H.263
- Huffman decode for JPEG, MPEG1/2, MPEG4 and H.263
- CBP (Coded Block Pattern) detection
- Auto block skip for un-coded block (CBP)
- Auto insertion of Macro Block Header data
- Auto insertion and deletion of stuffed zero byte in JPEG

The VLCD Block Diagram is shown in Figure 10

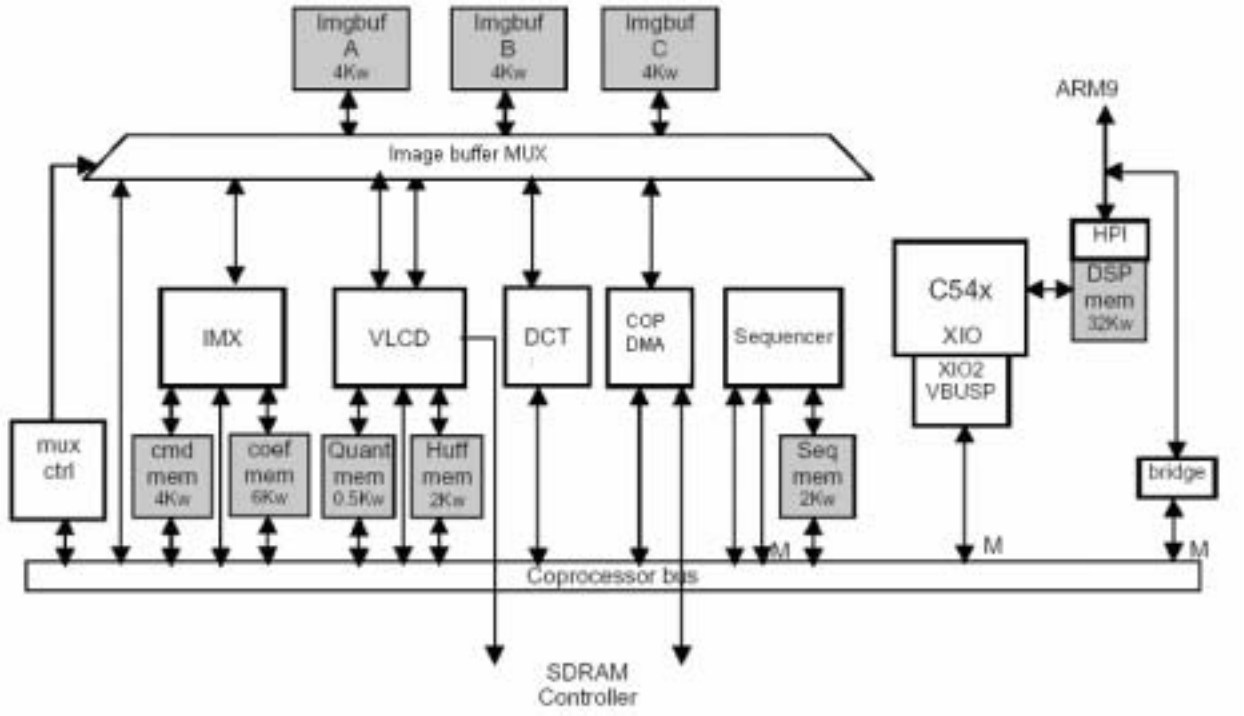


Figure 10: VLCD Block Diagram

DM320 VLCD has two image buffer ports, ibuf0 and ibuf1. Each port can be mapped to image buffer A, B, C, or none.

The VLCD module controls mapping input/output of Q, IQ, VLC, and VLD functions to ibuf0/ibuf1 ports. Mapping of ibuf0/ibuf1 to buffers A/B/C is controlled by the Image Buffer module.

Note that image buffer does not handle on-the-fly memory arbitration, and thus does not allow ibuf0 and ibuf1 ports to map to the same image buffer. When only one buffer is assigned to VLCD, VLCD must use just one port mapping to that buffer, and arbitration is implemented in the VLCD module.

5.1.2 Q/IQ Engine

With highly pipelined architecture, at its peak speed, Q/IQ module is capable of processing 180M DCT coefficients per second.

Q/IQ interfaces with the rest of the DSP subsystem via Image Buffers (A/B/C), Q/IQ Table and a set of control/status registers. The DCT coefficients and the quantized data both reside in one image buffer, selected by DSP, Sequencer, or ARM9 writing into the image buffer control register.

Q/IQ can access the Image Buffers (A/B/C). Q/IQ can also transfer data to and from the Image Buffer (A/B/C). Input DCT data must be placed in the image buffers (A/B/C) prior to the start of the quantization or de-quantization process.

Q/IQ reads the input data from and writes output data to the Image Buffer (A/B/C).

Q/IQ Table contains more configuration parameters in addition to what is specified in the control registers. During the Q/IQ process, Q/IQ engine reads the parameters from the Q/IQ Table. The additional parameters provide more flexible control over the Q/IQ process. The parameters include quantization/de-quantization coefficients, DC predictor data, zigzag scan path and block operators.

5.1.3 VLCD Engine

The VLCD coprocessor handles Huffman encode and decode operation for image compression algorithms such as JPEG, MPEG-1/2/4, and H.263. It operates with Huffman tables preloaded by DSP, via shared memory blocks. The VLCD is highly autonomous and is capable of carrying out all the steps of Huffman coding and decoding process with little or no help from the DSP.

VLC/VLD interfaces with the rest of the DSP subsystem via Image Buffers (A/B/C), VLC/VLD Huffman Table and a set of control registers. VLC/VLD can access the Image Buffer (A/B/C). VLC/VLD can also transfers data to and from the Image Buffer (A/B/C). The quantized DCT data must be placed in the image buffers (A/B/C) prior to the start of the VLC. The JPEG/MPEG bitstream data must be placed in the image buffers (A/B/C) prior to the start of the VLD. VLC/VLD read the input data from and writes output data to the Image Buffer (A/B/C). VLC/VLD Huffman Table contains more configuration parameters in addition to what is specified in the control registers. During the VLC/VLD process, VLC/VLD engine reads the parameters from the VLCD Huffman Table.

DSP configures and controls VLCD module via the control registers, which are mapped into DSP's IO space. The control registers allow the DSP to program Q/IQ and VLC/VLD or various industrial or user-defined standards, start the quantization or/and de-quantization process and VLC or VLD process and monitor its progress. The control registers also contain a set of address pointers to identify the location of input data and output data in the Image buffers (A/B/C) and the location of additional configuration parameters in the Q/IQ Table and VLC/VLD Huffman Table.

The VLC mode has an exception that its output data can also be transferred to SDRAM, instead of the image buffers.

5.2 Variable Length Coder/Decoder (VLCD) Modes

The VLCD module is fairly autonomous in processing one JPEG MCU (minimum coding unit) or MPEG Macroblock without intervention from DSP or sequencer. VLCD supports the following functions

- Q
- IQ
- VLC

- VLD
- Q + IQ
- Q + VLC
- Q + IQ + VLC
- VLD + Q

The various functions for VLCD can be selected by setting appropriate bits in FUNC field in MODE register.

VLCD supports two types of Quantizations type-0 and type-1. Quantization type in encode and decode for MPEG4 is either type-0 or type-1 can be selected by setting the bit QMPEGTYPE in MODE register.

VLCD supports the following codec modes

- JPEG
- MPEG1
- MPEG2
- H.263
- MPEG4
- MPEG4 data partition
- MPEG4 RVLC/D

The Codec Modes for VLCD can be selected by setting appropriate bits in MODSEL field in MODE register.

VLCD supports intra macro block and non-intra macro block. This can be selected by setting bit MPEGINTRA field in MODE register.

VLCD supports MPEG4 escape coding types 1, 2, 3, and 4 for MPEG4. MPEG4 escape coding type can be selected by setting bit ESCTYPE in MODE register.

VLCD supports MPEG picture coding types B, P, D, and I for MPEG4. MPEG4 picture coding type can be selected by setting the bit ESCTYPE in MODE register. Number of 8x8 blocks to be processed can be specified by setting appropriate bits in NUMBLKS of MODE register.

5.3 Image Buffer Port Selection

VLCD has two image buffer ports, ibuf0 and ibuf1. Each port can be mapped to image buffer A, B, C, or none.

Image Buffer selection for input qunatization can be selected by setting the bit MEMSEL in QIN_ADDR register and start address for input data used for quantization can be set by writing appropriate address in ADDR field of QIN_ADDR register.

Image Buffer selection for output qunatization can be selected by setting the bit MEMSEL in QOUT_ADDR register and start address for output data used for quantization can be set by writing appropriate address in ADDR field of QOUT_ADDR register.

Image Buffer selection for input inverse qunatization can be selected by setting the bit MEMSEL in IQIN_ADDR register and start address for input data used for inverse quantization can be set by writing appropriate address in ADDR field of IQIN_ADDR register.

Image Buffer selection for output, inverse quantization can be selected by setting the bit MEMSEL in IQOUT_ADDR register and start address for output data used for inverse quantization can be set by writing appropriate address in ADDR field of IQOUT_ADDR register.

Image Buffer selection for input VLCD can be selected by setting MEMSEL field in VLCDIN_ADDR register and start address for input data used for VLC/D can be set by writing appropriate address in ADDR field of VLCDIN_ADDR register.

Image Buffer selection for VLCD can be selected by setting MEMSEL field in VLCDOUT_ADDR. Start address for output data used for VLC/D can be set by writing appropriate address in ADDR field of VLCDOUT_ADDR register. VLC mode has an exception that its output data can also be transferred to SDRAM, instead of the image buffers, by writing “10” in MEMSEL field of VLCDOUT_ADDR register.

5.4 DC Predictor

VLCD has 6 qunatization DC predictor registers. Initial/Final DC predictors for quantization can be specified in PRED0 field of DC_PRED0 register, PRED1 of DC_PRED1 register, PRED2 of DC_PRED2 register, PRED3 of DC_PRED3 register, PRED4 of DC_PRED4 register, and PRED5 of DC_PRED5 register.

VLCD has 6 inverse qunatization DC predictor registers. Initial/Final DC predictors for inverse quantization can be specified in IPRED0 field of IDC_PRED0 register, IPRED1 of IDC_PRED1 register, IPRED2 of IDC_PRED2 register, IPRED3 of IDC_PRED3 register, IPRED4 of IDC_PRED4 register, and IPRED5 of IDC_PRED5 register.

VLCD has 2 scale register one for MPEG quantization and another for inverse quantization. The data can be scaled to required precision by writing appropriate value in MQ field of MPEG_Q register during MPEG quantization and in MINVQ field of MPEG_INVQ register during MPEG inverse quantization.

VLCD has 2 delta register one for MPEG delta quantization and another for delta inverse quantization. The data can be scaled to required precision by writing appropriate value in MDELQ field of MPEG_DELTA_Q register during MPEG delta quantization and in MDELIQ field of MPEG_DELTA_IQ register during MPEG delta inverse quantization.

It has one MPEG threshold register. The threshold value for use in MPEG quantization can be set by writing appropriate threshold value in MTHRED of MPEG_THRED register.

5.5 VLC in DMA Mode

VLCD supports DMA mode only for VLC function. The data transfer direction is always from VLC to SDRAM. The SDRAM address can be set in two registers by writing approtiate lower and higher addresses in SDRAM_ADDRL register and SDRAM_ADDRH register respectively. VLC function can be selected by writing appropriate value in FUNC field of MODE register. DMA Mode can be set by writing "10" in MEMSEL of VLCDOUT_ADDR register.

If one of the FIFOs fails to complete a DMA transfer by the time the other FIFO becomes full or the request is dropped by MTC due to long access latency, the higher prioritized requests than VLC, etc., the FIFO Overflow Error will be detected. If that occurs, further DMA transfers are no longer possible to initiate unless the user initializes the FIFO pointers. As a recovery action to get the FIFOs back in normal, the following steps need to be taken:

1. Ensure the DMA transfer as well as the VLC operation are completed (notified via interrupt signal)

2. Check to see whether FIFO(bit 4) of VLD_ERRSTAT has been set to '1'

3. If detected, set '1' to INITFIFOPTR(bit 3) of START register in order to clear the FIFO pointers

5.6 Endianness

VLCD supports endianness for VLC and VLD functions. It supports little and big endian byte address. Endian for VLC/VLD can be selected by writing appropriate value in Endian field of CTRL register. Clock for VLC/VLD can be made on or off by writing in CLKON field of register.

5.7 VLC/VLD Start and End Address

VLCD supports circular type of addressing for VLC/VLD function. In circular address when the read/write pointer reaches the end address it automatically wraps back to start address.

The start address for VLC function can be set in RING_START register, when input pointer reaches end address, input pointer starts at start address automatically.

The end address for VLC function can be set in RING_END register, when output pointer reaches end address, output pointer starts at start address automatically.

The start address for VLD function can be set in RING_START register, when input pointer reaches end address, input pointer starts at start address automatically.

The end address for VLD function can be set in RING_END register, when output pointer reaches end address, output pointer starts at start address automatically.

5.8 VLC/VLD Scan Modes

VLCD supports the following scan modes

- Linear scan
- Zig-zag scan
- Alternate vertical scan
- Alternate horizontal scan

VLCD has 6 QIQ configuration registers. Scan mode for VLC/VLD can be set by writing appropriate value in scan field of QIQ_CONFIG0 register, QIQ_CONFIG1 register, QIQ_CONFIG2 register, QIQ_CONFIG3 register, QIQ_CONFIG4 register, and QIQ_CONFIG5 register.

VLCD has 8 banks each for quantization and inverse quantization. Banks can be selected for quantization by setting value in QMATSEL of QIQ_CONFIG0 register, QIQ_CONFIG1 register, QIQ_CONFIG2 register, QIQ_CONFIG3 register, QIQ_CONFIG4 register, and QIQ_CONFIG5 register. Banks can be selected for inverse quantization by setting value in IQMATSEL of QIQ_CONFIG0 register, QIQ_CONFIG1 register, QIQ_CONFIG2 register, QIQ_CONFIG3 register, QIQ_CONFIG4 register, and QIQ_CONFIG5 register.

VLCD has 6 DC predictor registers. Predictor value for each bank can be selected by setting value in DCPREDSEL of QIQ_CONFIG0 register, QIQ_CONFIG1 register, QIQ_CONFIG2 register, QIQ_CONFIG3 register, QIQ_CONFIG4 register, and QIQ_CONFIG5 register.

5.9 MPEG coded Block Pattern

CBP control can be made on or off by writing the value in CBPON of MPEG_CBP register. Coded block pattern can be selected by writing value in CBP of MPEG_CBP register. Coded block pattern, indicating at least one nonzero in a block. For the standard six-block macro block, CBP of the first block would be in bit position five. The second block's CBP bit in position 4, down to the last block being in bit position 0.

5.10 LUMA_VECTOR

Luma bit vector can be set by writing the value in LUMVECT of LUMA_VECTOR register. This register contains the Luma bit vector. Each block will have a 1-bit entry into this vector specifying whether the block has luminance data or chrominance data. Setting this bit high indicates that block has luminance data. This bit vector shall be arranged such that the Luma bit for each block shall be in the bit position of number of blocks minus the block count number. Thus for the

standard six block macro block, the first block's luma bit would be in bit position five. The second block's coded bit in position 4, down to the last block being in bit position 0.

5.11 Huffman Table

The VLCD coprocessor handles Huffman encode and decode operation for image compression algorithms such as JPEG, MPEG-1/2/4, and H.263. It operates with Huffman tables preloaded by DSP, via shared memory blocks. The Huffman DC Y table base address can be set in HDCY of HUFFTAB_DCY register. The Huffman DC UV table base address can be set in HDCUV of HUFFTAB_DCUV register. The Huffman AC0 table base address can be set in HAC0 of register HUFFTAB_AC0. The Huffman AC1 table base address can be set in HAC1 of HUFFTAB_AC1 register.

5.12 MPEG Max Level Table

VLCD has two MPEG max level table register to specify the base address. The base address for MPEG max 0 level tables can be set in MAX0 of OFLEV_MAXOTAB register. The base address for MPEG max 1 level tables can be set in MAX0 of OFLEV_MAX1TAB register.

5.13 Control Lookup Table

VLCD has 4 control registers to specify the control table base address. DC Y control lookup table base address is set in CDCY of CTLTAB_DCY register. This register contains the address pointer to the starting word of the control table DC Y lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32-bit word boundary. DC UV control lookup table base address is set in CDCUV of CTLTAB_DCUV register. This register contains the address pointer to the starting word of the control table DC UV lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32-bit word boundary. AC0 control lookup table base address is set in CAC0 of CTLTAB_AC0 register. This register contains the address pointer to the starting word of the control table DC AC0 lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32-bit word boundary. AC1 control lookup table base address is set in CAC1 of CTLTAB_AC1 register. This register contains the address pointer to the starting word of the control table DC AC1 lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32-bit word boundary.

5.14 Symbol Lookup Table

VLCD has 8 symbol registers to specify the symbol table offset address. DC Y symbol lookup table offset address is set in ODCY of OFFSET_DCY register. This register contains the offset value used to address the symbol table DC Y lookup table. The purpose of this offset is to avoid negative numbers being used in the control table. DC UV symbol lookup table offset address is set in ODCUV of OFFSET_DCUV register. This register contains the offset value used to

address the symbol table DC UV lookup table. The purpose of this offset is to avoid negative numbers being used in the control table. AC0 symbol lookup table offset address is set in OACO of OFFSET_AC0 register. This register contains the offset value used to address the symbol table DC AC0 lookup table. The purpose of this offset is to avoid negative numbers being used in the control table. AC1 symbol lookup table offset address is set in OAC1 of OFFSET_AC1 register. This register contains the offset value used to address the symbol table DC AC1 lookup table. The purpose of this offset is to avoid negative numbers being used in the control table. DC Y symbol lookup table base address is set in SDCY of SYMTAB_DCY register. This register contains the address pointer to the starting word of the symbol table DC Y lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32-bit word boundary. DC UV symbol lookup table base address is set in SDCUV of SYMTAB_DCUVregister. This register contains the address pointer to the starting word of the symbol table DC UV lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32-bit word boundary. AC0 symbol lookup table base address is set in SAC0 of SYMTAB_AC0 register. This register contains the address pointer to the starting word of the symbol table DC AC0 lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32-bit word boundary. AC1 symbol lookup table base address is set in SAC1 of SYMTAB_AC1 register. This register contains the address pointer to the starting word of the symbol table DC AC1 lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32-bit word boundary.

5.15 Variable Length Coder Decoder Register Map (VLCD)

DSP	SEQ	ARM	Register	Description
0100	F100	0009:E200	START	VLCD Start Register
0101	F101	0009:E202	MODE	VLCD Mode Register
0102	F102	0009:E204	QIN_ADDR	Quantization Input Address Register
0103	F103	0009:E206	QOUT_ADDR	Quantization Output Address Register
0104	F104	0009:E208	IQIN_ADDR	Inverse Quantization Input Address Register
0105	F105	0009:E20A	IQOUT_ADDR	Inverse Quantization Output Address Register
0106	F106	0009:E20C	VLCDIN_ADDR	VLCD Input Address
0107	F107	0009:E20E	VLCDOUT_ADDR	VLCD Output Address
0108	F108	0009:E210	DC_PRED0	Quantization DC Predictor 0 Register
0109	F109	0009:E212	DC_PRED1	Quantization DC Predictor 1 Register
010A	F10A	0009:E214	DC_PRED2	Quantization DC Predictor 2 Register
010B	F10B	0009:E216	DC_PRED3	Quantization DC Predictor 3 Register
010C	F10C	0009:E218	DC_PRED4	Quantization DC Predictor 4 Register
010D	F10D	0009:E21A	DC_PRED5	Quantization DC Predoctor 4 Register
010E	F10E	0009:E21C	IDC_PRED0	Inverse Quantization DC Predictor 0 Register
010F	F10F	0009:E21E	IDC_PRED1	Inverse Quantization DC Predictor 1 Register
0110	F110	0009:E220	IDC_PRED2	Inverse Quantization DC Predictor 2 Register
0111	F111	0009:E222	IDC_PRED3	Inverse Quantization DC Predictor 3 Register
0112	F112	0009:E224	IDC_PRED4	Inverse Quantization DC Predictor 4 Register
0113	F113	0009:E226	IDC_PRED5	Inverse Quantization DC Predoctor 4 Register
0114	F114	0009:E228	MPEG_INVQ	MPEG Inverse Quantization Scale Register
0115	F115	0009:E22A	MPEG_Q	MPEG Quantization Scale Register
0116	F116	0009:E22C	MPEG_DELTA_Q	MPEG Quantization Delta Register
0117	F117	0009:E22E	MPEG_DELTA_IQ	MPEG Inverse Quantization Delta Register
0118	F118	0009:E230	MPEG_THRED	MPEG Thred Register
0119	F119	0009:E232	MPEG_CBP	MPEG coded Block Pattern Register
011A	F11A	0009:E234	LUMA_VECTOR	LUMA Bit Vector Register
011B	F11B	0009:E236	HUFFTAB_DCY	Huffman DC Y Table Base Address Register
011C	F11C	0009:E238	HUFFTAB_DCUV	Huffman DC UV Table Base Address Register
011D	F11D	0009:E23A	HUFFTAB_AC0	Huffman AC0 Table Base Address Register
011E	F11E	0009:E23C	HUFFTAB_AC1	Huffman AC1 Table Base Address Register
011F	F11F	0009:E23E	OFLEV_MAXOTAB	MPEG Max 0 Level Table Base Address Register
0120	F120	0009:E240	OFLEV_MAX1TAB	MPEG Max 1 Level Table Base Address Register
0121	F121	0009:E242	CTLTAB_DCY	DC Y Control Lookup Table Base Address Register
0122	F122	0009:E244	CTLTAB_DCUV	DC UV Control Lookup Table Base Address Register
0123	F123	0009:E246	CTLTAB_AC0	AC0 Control Lookup Table Base Address Register
0124	F124	0009:E248	CTLTAB_AC1	AC1 Control Lookup Table Base Address Register
0125	F125	0009:E24A	OFFSET_DCY	DC Y Symbol Lookup Table Address Offset Register
0126	F126	0009:E24C	OFFSET_DCUV	DC UV Symbol Lookup Table Address Offset Register
0127	F127	0009:E24E	OFFSET_AC0	AC0 Symbol Lookup Table Address Offset Register
0128	F128	0009:E250	OFFSET_AC1	AC1 Symbol Lookup Table Address Offset Register
0129	F129	0009:E252	SYMTAB_DCY	DC Y Symbol Lookup Table Base Address Register

012A	F12A	0009:E254	SYMTAB_DCUV	DC UV Symbol Lookup Table Base Address Register
012B	F12B	0009:E256	SYMTAB_AC0	AC0 Symbol Lookup Table Base Address Register
012C	F12C	0009:E258	SYMTAB_AC1	AC1 Symbol Lookup Table Base Address Register
012D	F12D	0009:E25A	CTL	VLD Control Register
012E	F12E	0009:E25C	VLD_NRBIT_DC	DC Number of Bits Register
012F	F12F	0009:E25E	VLD_NRBIT_AC	AC Number of Bits Register
0130	F130	0009:E260	BITS_BPTR	Bits Pointer Register
0131	F131	0009:E262	BITS_WORD	Bits Word Register
0132	F132	0009:E264	BYTE_ALIGN	Byte Align Register
0133	F133	0009:E266	HEAD_ADDR	Header Address Register
0134	F134	0009:E268	HEAD_NUM	Number of Header Data Register
0135	F135	0009:E26A	QIQ_CONFIG0	QIQ Configuration Register #0
0136	F136	0009:E26C	QIQ_CONFIG1	QIQ configuration Register #1
0137	F137	0009:E26E	QIQ_CONFIG2	QIQ Congiguration Register #2
0138	F138	0009:E270	QIQ_CONFIG3	QIQ configuration Register #3
0139	F139	0009:E272	QIQ_CONFIG4	QIQ Congiguration Register #4
013A	F13A	0009:E274	QIQ_CONFIG5	QIQ configuration Register #5
013B	F13B	0009:E276	VLD_ERRCTL	VLD Error Control Register
013C	F13C	0009:E278	VLD_ERRSTAT	VLD Error Status Register
013D	F13D	0009:E27A	RING_START	Ring Buffer Start Address Register
013E	F13E	0009:E27C	RING_END	Ring Buffer End Address Register
013F	F13F	0009:E27E	VLD_PREFIX_DC	VLD Prefix Register - DC
0140	F140	0009:E280	VLD_PREFIX_AC	VLD Prefix Register - AC
0141	F141	0009:E282	CTRL	VLCD Clock Control
0142	F142	0009:E284	SDRAM_ADDRH	High address of SDRAM bits[22:16]
0143	F143	0009:E286	SDRAM_ADDRL	Low address of SDRAM bits[15:0]

5.16 VLCD Registers

5.16.1 START

VLCD Start Register

START		DSP	IO	0100	offset:	0x00									default:	0x0000
		SEQ		F100												
		ARM		0009:E200												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	INITFIFOPTR	FLUSHFIFO	INT	GO
R/W	-	-	-	-	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:4	RSV			Reserved
3	INITFIFOPTR	0	R/W	INITFIFO Pointer Write: 0: Not used (do not write 0) 1: Init R/W pointers of FIFO for SDRAM transfer Read: 0: FIFO flushed 1: Flushing FIFO Note: This is used for FIFO overflow recovery and is valid only during Huffman encode, SDRAM o/p is selected and when VLCD is in idle mode
2	FLUSHFIFO	0	R/W	VLCD Go Write: 0: Not used (do not write 0) 1: Flush FIFO Read: 0: Idle 1: Busy Note: Valid only during Huffman encode, SDRAM o/p is selected and when VLCD is in idle mode Writing '1' to the FLUSHFIFO field initiates a DMA transfer if FIFO still retains some encoded bitstream when a VLC operation is completed, and VLCD sends the remaining data in either of FIFOs to SDRAM via MTC. The FLUSHFIFO field is intended in use when a total amount of encoded data is not a multiple of 32bytes(i.e. 16 words), since each DMA transfer is performed on a 32-bytes basis. The user can obtain information on whether it is a multiple or not by reading the BITS_BPTR and BITS_WORD registers.
1	INT	0	R/W	VLCD to DSP Interrupt » 0: Disable interrupt « » 1: Enable interrupt «
0	GO	0	R/W	VLCD Go Write: 0: Not used (do not write 0) 1: Start VLCD module Read: 0: Idle 1: Busy Writing "1" starts the VLCD module. Reading provides VLCD execution status.

5.16.2 MODE

VLCD Mode Register

MODE		DSP	IO	0101	offset:	0x01									default:	0x0000
		SEQ		F101												
		ARM		0009:E202												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MPEGRND	NUMBLKS[2]	NUMBLKS[1]	NUMBLKS[0]	JPEGSYNC	MPEGTYPE	ESCTYPE	INTRAVLC	MPEGINTRA	MODESEL[2]	MODESEL[1]	MODESEL[0]	QMPEGTYPE	FUNC[2]	FUNC[1]	FUNC[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15	MPEGRND	0	R/W	Round control for MPEG INVQ » 0: round off « » 1: round on «
14:12	NUMBLKS	0	R/W	Specify number of 8x8 blocks to process Number of blocks can be 1 through 6.
11	JPEGSYNC	0	R/W	JPEG Sync Insertion » 0: Off « » 1: On « Writing “1” enables inserting 00 byte after FF byte
10	MPEGTYPE	0	R/W	MPEG Picture Coding Type » 0: I,P,B picture coding « » 1: D picture coding «
9	ESCTYPE	0	R/W	MPEG4 escape coding type » 0: escape type 3 only « » 1: escape type 1, 2 and 3 «
8	INTRAVLC	0	R/W	MPEG Intra VLC Format
7	MPEGINTRA	0	R/W	Intra Macro Block Select » 0: Non-intra macro block « » 1: Intra macro block « (In JPEG mode should be set to '1') Selects whether blocks are encoded as Intra Macro blocks
6:4	MODESEL	0	R/W	Codec Mode Select » 000: JPEG « » 001: MPEG1 « » 010: MPEG2 « » 011: H.263 « » 100: MPEG4 « » 101: MPEG4 data partition « » 110: MPEG4 RVLC/D «
3	QMPEGTYPE	0	R/W	Quantization type for Encode and Decode in MPEG4 » 0: TYPE 0 « » 1: TYPE 1 «
2:0	FUNC	0	R/W	Function Select » 000: Q « » 001: IQ « » 010: VLC « » 011: VLD « » 100: Q + IQ « » 101: Q + VLC « » 110: Q + IQ + VLC « » 111: VLD + IQ «

5.16.3 QIN_ADDR

Quantization Input Address Register

QIN_ADDR		DSP	IO	0102	offset:	0x02									default:	0x0000
		SEQ		F102												
		ARM		0009:E204												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	MEMSEL	RSV	RSV	ADDR[12]	ADDR[11]	ADDR[10]	ADDR[9]	ADDR[8]	ADDR[7]	ADDR[6]	ADDR[5]	ADDR[4]	ADDR[3]	ADDR[2]	ADDR[1]	ADDR[0]
R/W	R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15	MEMSEL	0	R/W	Image Buffer Port Selection » 0: IBUF0 « » 1: IBUF1 «
14:13	RSV			Reserved
12:0	ADDR	0	R/W	Start address of input data used for Quantization

5.16.4 QOUT_ADDR

Quantization Output Address Register

QOUT_ADDR		DSP	IO	0103	offset:	0x03									default:	0x0000
		SEQ		F103												
		ARM		0009:E206												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	MEMSEL	RSV	RSV	ADDR[12]	ADDR[11]	ADDR[10]	ADDR[9]	ADDR[8]	ADDR[7]	ADDR[6]	ADDR[5]	ADDR[4]	ADDR[3]	ADDR[2]	ADDR[1]	ADDR[0]
R/W	R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15	MEMSEL	0	R/W	Image Buffer Port Selection » 0: IBUF0 « » 1: IBUF1 «
14:13	RSV			Reserved
12:0	ADDR	0	R/W	Start address of output data used for Quantization

5.16.5 IQIN_ADDR

Inverse Quantization Input Address Register

IQIN_ADDR		DSP	IO	0104	offset:	0x04									default:	0x0000
		SEQ		F104												
		ARM		0009:E208												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	MEMSEL	RSV	RSV	ADDR[12]	ADDR[11]	ADDR[10]	ADDR[9]	ADDR[8]	ADDR[7]	ADDR[6]	ADDR[5]	ADDR[4]	ADDR[3]	ADDR[2]	ADDR[1]	ADDR[0]
R/W	R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15	MEMSEL	0	R/W	Image Buffer Port Selection » 0: IBUF0 « » 1: IBUF1 «
14:13	RSV			Reserved
12:0	ADDR	0	R/W	Start address of input data used for inverse quantization

5.16.6 IQOUT_ADDR

Inverse Quantization Output Address Register

IQOUT_ADDR		DSP	IO	0105	offset:	0x05									default:	0x0000
		SEQ		F105												
		ARM		0009:E20A												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	MEMSEL	RSV	RSV	ADDR[12]	ADDR[11]	ADDR[10]	ADDR[9]	ADDR[8]	ADDR[7]	ADDR[6]	ADDR[5]	ADDR[4]	ADDR[3]	ADDR[2]	ADDR[1]	ADDR[0]
R/W	R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15	MEMSEL	0	R/W	Image Buffer Port Selection » 0: IBUF0 « » 1: IBUF1 «
14:13	RSV			Reserved
12:0	ADDR	0	R/W	Start address of output data used for inverse quantization

5.16.7 VLCDIN_ADDR

VLCD Input Address

VLCDIN_ADDR		DSP	IO	0106	offset:	0x06									default:	0x0000
		SEQ		F106												
		ARM		0009:E20C												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	MEMSEL	RSV	RSV	ADDR[12]	ADDR[11]	ADDR[10]	ADDR[9]	ADDR[8]	ADDR[7]	ADDR[6]	ADDR[5]	ADDR[4]	ADDR[3]	ADDR[2]	ADDR[1]	ADDR[0]
R/W	R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15	MEMSEL	0	R/W	Image Buffer Port Selection » 0: IBUF0 « » 1: IBUF1 «
14:13	RSV			Reserved
12:0	ADDR	0	R/W	Start address of input data used for VLC/VLD

5.16.8 VLCDOUT_ADDR

VLCD Output Address

VLCDOUT_ADDR		DSP	IO	0107	offset:	0x07									default:	0x0000
		SEQ		F107												
		ARM		0009:E20E												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MEMSEL[1]	MEMSEL[0]	RSV	ADDR[12]	ADDR[11]	ADDR[10]	ADDR[9]	ADDR[8]	ADDR[7]	ADDR[6]	ADDR[5]	ADDR[4]	ADDR[3]	ADDR[2]	ADDR[1]	ADDR[0]
R/W	R/W	R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:14	MEMSEL	0	R/W	Image Buffer Port Selection » 00: IBUF0 « » 01: IBUF1 « » 10: Bitstram DMA to SDRAM « » 11: None (bitstream output discarded) « Note that '10' and '11' are only valid when doing Huffman encode: VLCD_MODE[2:0] = {2, 5, 6}
13	RSV			Reserved
12:0	ADDR	0	R/W	Start address of output data used for VLC/VLD When VLD, the address should be 32bit aligned.

5.16.9 DC_PRED0

Quantization DC Predictor 0 Register

DC_PRED0		DSP	IO	0108	offset:	0x08									default:	0x0000
		SEQ		F108												
		ARM		0009:E210												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	PRED0[11]	PRED0[10]	PRED0[9]	PRED0[8]	PRED0[7]	PRED0[6]	PRED0[5]	PRED0[4]	PRED0[3]	PRED0[2]	PRED0[1]	PRED0[0]
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	PRED0	0	R/W	Initial/final DC predictors for quantization

5.16.10 DC_PRED1

Quantization DC Predictor 1 Register

DC_PRED1		DSP	IO	0109	offset:	0x09									default:	0x0000
		SEQ		F109												
		ARM		0009:E212												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	PRED1[11]	PRED1[10]	PRED1[9]	PRED1[8]	PRED1[7]	PRED1[6]	PRED1[5]	PRED1[4]	PRED1[3]	PRED1[2]	PRED1[1]	PRED1[0]
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	PRED1	0	R/W	Initial/final DC predictors for quantization

5.16.11 DC_PRED2

Quantization DC Predictor 2 Register

DC_PRED2		DSP	IO	010A	offset:	0x0A									default:	0x0000
		SEQ		F10A												
		ARM		0009:E214												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	PRED2[11]	PRED2[10]	PRED2[9]	PRED2[8]	PRED2[7]	PRED2[6]	PRED2[5]	PRED2[4]	PRED2[3]	PRED2[2]	PRED2[1]	PRED2[0]
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	PRED2	0	R/W	Initial/final DC predictors for quantization

5.16.12 DC_PRED3

Quantization DC Predictor 3 Register

DC_PRED3		DSP	IO	010B	offset:	0x0B									default:	0x0000
		SEQ		F10B												
		ARM		0009:E216												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	PRED3[11]	PRED3[10]	PRED3[9]	PRED3[8]	PRED3[7]	PRED3[6]	PRED3[5]	PRED3[4]	PRED3[3]	PRED3[2]	PRED3[1]	PRED3[0]
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	PRED3	0	R/W	Initial/final DC predictors for quantization

5.16.13 DC_PRED4

Quantization DC Predictor 4 Register

DC_PRED4		DSP	IO	010C	offset:	0x0C									default:	0x0000
		SEQ		F10C												
		ARM		0009:E218												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	PRED4[11]	PRED4[10]	PRED4[9]	PRED4[8]	PRED4[7]	PRED4[6]	PRED4[5]	PRED4[4]	PRED4[3]	PRED4[2]	PRED4[1]	PRED4[0]
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	PRED4	0	R/W	Initial/final DC predictors for quantization

5.16.14 DC_PRED5

Quantization DC Predictor 5 Register

DC_PRED5		DSP	IO	010D	offset:	0x0D									default:	0x0000
		SEQ		F10D												
		ARM		0009:E21A												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	PRED5[11]	PRED5[10]	PRED5[9]	PRED5[8]	PRED5[7]	PRED5[6]	PRED5[5]	PRED5[4]	PRED5[3]	PRED5[2]	PRED5[1]	PRED5[0]
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	PRED5	0	R/W	Initial/final DC predictors for quantization

5.16.15 IDC_PRED0

Inverse Quantization DC Predictor 0 Register

IDC_PRED0		DSP	IO	010E	offset:	0x0E									default:	0x0000
		SEQ		F10E												
		ARM		0009:E21C												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	IPRED0[11]	IPRED0[10]	IPRED0[9]	IPRED0[8]	IPRED0[7]	IPRED0[6]	IPRED0[5]	IPRED0[4]	IPRED0[3]	IPRED0[2]	IPRED0[1]	IPRED0[0]
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	IPRED0	0	R/W	Initial/final DC predictors for inverse quantization

5.16.16 IDC_PRED1

Inverse Quantization DC Predictor 1 Register

IDC_PRED1		DSP	IO	010F	offset:	0x0F									default:	0x0000
		SEQ		F10F												
		ARM		0009:E21E												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	IPRED1[11]	IPRED1[10]	IPRED1[9]	IPRED1[8]	IPRED1[7]	IPRED1[6]	IPRED1[5]	IPRED1[4]	IPRED1[3]	IPRED1[2]	IPRED1[1]	IPRED1[0]
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	IPRED1	0	R/W	Initial/final DC predictors for inverse quantization

5.16.17 IDC_PRED2

Inverse Quantization DC Predictor 2 Register

IDC_PRED2		DSP	IO	0110	offset:	0x10										default:	0x0000
		SEQ		F110													
		ARM		0009:E220													
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Name	RSV	RSV	RSV	RSV	IPRED2[11]	IPRED2[10]	IPRED2[9]	IPRED2[8]	IPRED2[7]	IPRED2[6]	IPRED2[5]	IPRED2[4]	IPRED2[3]	IPRED2[2]	IPRED2[1]	IPRED2[0]	
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	IPRED2	0	R/W	Initial/final DC predictors for inverse quantization

5.16.18 IDC_PRED3

Inverse Quantization DC Predictor 3 Register

IDC_PRED3		DSP	IO	0111	offset:	0x11										default:	0x0000
		SEQ		F111													
		ARM		0009:E222													
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Name	RSV	RSV	RSV	RSV	IPRED3[11]	IPRED3[10]	IPRED3[9]	IPRED3[8]	IPRED3[7]	IPRED3[6]	IPRED3[5]	IPRED3[4]	IPRED3[3]	IPRED3[2]	IPRED3[1]	IPRED3[0]	
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	IPRED3	0	R/W	Initial/final DC predictors for inverse quantization

5.16.19 IDC_PRED4

Inverse Quantization DC Predictor 4 Register

IDC_PRED4		DSP	IO	0112	offset:	0x12										default:	0x0000
		SEQ		F112													
		ARM		0009:E224													
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Name	RSV	RSV	RSV	RSV	IPRED4[11]	IPRED4[10]	IPRED4[9]	IPRED4[8]	IPRED4[7]	IPRED4[6]	IPRED4[5]	IPRED4[4]	IPRED4[3]	IPRED4[2]	IPRED4[1]	IPRED4[0]	
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	IPRED4	0	R/W	Initial/final DC predictors for inverse quantization

5.16.20 IDC_PRED5

Inverse Quantization DC Predictor 5 Register

IDC_PRED5		DSP	IO	0113	offset:	0x13										default:	0x0000
		SEQ		F113													
		ARM		0009:E226													
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Name	RSV	RSV	RSV	RSV	IPRED5[11]	IPRED5[10]	IPRED5[9]	IPRED5[8]	IPRED5[7]	IPRED5[6]	IPRED5[5]	IPRED5[4]	IPRED5[3]	IPRED5[2]	IPRED5[1]	IPRED5[0]	
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	IPRED5	0	R/W	Initial/final DC predictors for inverse quantization

5.16.21 MPEG_INVQ

MPEG Quantization Scale Register

MPEG_INVQ		DSP	IO	0114	offset:	0x14									default:	0x0000
		SEQ		F114												
		ARM		0009:E228												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	MINVQ[15]	MINVQ[14]	MINVQ[13]	MINVQ[12]	MINVQ[11]	MINVQ[10]	MINVQ[9]	MINVQ[8]	MINVQ[7]	MINVQ[6]	MINVQ[5]	MINVQ[4]	MINVQ[3]	MINVQ[2]	MINVQ[1]	MINVQ[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	MINVQ	0	R/W	2^15/quant_scale for MPEG quantization

5.16.22 MPEG_Q

MPEG de-Quantization Scale Register

MPEG_Q		DSP	IO	0115	offset:	0x15									default:	0x0000
		SEQ		F115												
		ARM		0009:E22A												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	MQ[7]	MQ[6]	MQ[5]	MQ[4]	MQ[3]	MQ[2]	MQ[1]	MQ[0]
R/W	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:8	RSV			Reserved
7:0	MQ	0	R/W	quant_scale for MPEG de-quantization

5.16.23 MPEG_DELTA_Q

MPEG Quantization Delta Register

MPEG_DELTA_Q		DSP	IO	0116	offset:	0x16									default:	0x0000
		SEQ		F116												
		ARM		0009:E22C												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	MDELQ[8]	MDELQ[7]	MDELQ[6]	MDELQ[5]	MDELQ[4]	MDELQ[3]	MDELQ[2]	MDELQ[1]	MDELQ[0]
R/W	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:9	RSV			Reserved
8:0	MDELQ	0	R/W	Number of delta value for use in MPEG quantization (2's compliment)

5.16.24 MPEG_DELTA_IQ

MPEG Inverse Quantization Delta Register

MPEG_DELTA_IQ		DSP	IO	0117	offset:	0x17									default:	0x0000
		SEQ		F117												
		ARM		0009:E22E												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	MDELIQ[8]	MDELIQ[7]	MDELIQ[6]	MDELIQ[5]	MDELIQ[4]	MDELIQ[3]	MDELIQ[2]	MDELIQ[1]	MDELIQ[0]
R/W	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:9	RSV			Reserved
8:0	MDELIQ	0	R/W	Number of delta value for use in MPEG inverse quantization (2's compliment)

5.16.25 MPEG_THRED

MPEG Thred Register

MPEG_THRED		DSP	IO	0118	offset:	0x18									default:	0x0000
		SEQ		F118												
		ARM		0009:E230												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	MTHRED[11]	MTHRED[10]	MTHRED[9]	MTHRED[8]	MTHRED[7]	MTHRED[6]	MTHRED[5]	MTHRED[4]	MTHRED[3]	MTHRED[2]	MTHRED[1]	MTHRED[0]
R/W	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:12	RSV			Reserved
11:0	MTHRED	0	R/W	Number of threds value for use in MPEG Quantization

5.16.26 MPEG_CBP

MPEG coded Block Pattern Register

MPEG_CBP		DSP	IO	0119	offset:	0x19									default:	0x0000
		SEQ		F119												
		ARM		0009:E232												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	CBPON	RSV	RSV	CBP[5]	CBP[4]	CBP[3]	CBP[2]	CBP[1]	CBP[0]
R/W	-	-	-	-	-	-	-	R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	0	-	-	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:9	RSV			Reserved
8	CBPON	0	R/W	CBP Detection Control » 0: CBP detect off (JPEG) « » 1: CBP detect on «
7:6	RSV			Reserved
5:0	CBP	0	R/W	Coded block pattern, indicating at least one nonzero in a block. For the standard six block macro block, CBP of the first block would be in bit position five. The second block's CBP bit in position 4, down to the last block being in bit position 0.

5.16.27 LUMA_VECTOR

LUMA Bit Vector Register

LUMA_VECTOR		DSP	IO	011A	offset:	0x1A									default:	0x0000
		SEQ		F11A												
		ARM		0009:E234												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	LUMVECT[5]	LUMVECT[4]	LUMVECT[3]	LUMVECT[2]	LUMVECT[1]	LUMVECT[0]
R/W	-	-	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:6	RSV			Reserved
5:0	LUMVECT	0	R/W	Luma Bit Vector This register contains the Luma bit vector. Each block will have a 1 bit entry into this vector specifying whether the block has luminance data or chrominance data. Setting this bit high indicates that block has luminance data. This bit vector shall be arranged such that the Luma bit for each block shall be in the bit position of number of blocks minus the block count number. Thus for the standard six block macro block, the first block's luma bit would be in bit position five. The second block's coded bit in position 4, down to the last block being in bit position 0

5.16.28 HUFFTAB_DCY

Huffman DC Y Table Base Address Register

HUFFTAB_DCY		DSP	IO	011B	offset:	0x1B									default:	0x0000
		SEQ		F11B												
		ARM		0009:E236												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	HDCY[10]	HDCY[9]	HDCY[8]	HDCY[7]	HDCY[6]	HDCY[5]	HDCY[4]	HDCY[3]	HDCY[2]	HDCY[1]	HDCY[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	HDCY	0	R/W	Base address for Huffman DC Y tables, should be even (32-bit aligned).

5.16.29 HUFFTAB_DCUV

Huffman DC UV Table Base Address Register

HUFFTAB_DCUV		DSP	IO	011C	offset:	0x1C									default:	0x0000
		SEQ		F11C												
		ARM		0009:E238												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	HDCUV[10]	HDCUV[9]	HDCUV[8]	HDCUV[7]	HDCUV[6]	HDCUV[5]	HDCUV[4]	HDCUV[3]	HDCUV[2]	HDCUV[1]	HDCUV[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	HDCUV	0	R/W	Base address for Huffman DC UV tables, should be even (32-bit aligned).

5.16.30 HUFFTAB_AC0

Huffman AC0 Table Base Address Register

HUFFTAB_AC0		DSP	IO	011D	offset:	0x1D									default:	0x0000
		SEQ		F11D												
		ARM		0009:E23A												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	HAC0[10]	HAC0[9]	HAC0[8]	HAC0[7]	HAC0[6]	HAC0[5]	HAC0[4]	HAC0[3]	HAC0[2]	HAC0[1]	HAC0[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	HAC0	0	R/W	Base address for Huffman AC0 tables, should be even (32-bit aligned).

5.16.31 HUFFTAB_AC1

Huffman AC1 Table Base Address Register

HUFFTAB_AC1		DSP	IO	011E	offset:	0x1E									default:	0x0000
		SEQ		F11E												
		ARM		0009:E23C												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	HAC1[10]	HAC1[9]	HAC1[8]	HAC1[7]	HAC1[6]	HAC1[5]	HAC1[4]	HAC1[3]	HAC1[2]	HAC1[1]	HAC1[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	HAC1	0	R/W	Base address for Huffman AC1 tables, should be even (32-bit aligned).

5.16.32 OFLEV_MAXOTAB

MPEG Max 0 Level Table Base Address Register

OFLEV_MAXOTAB		DSP	IO	011F	offset:	0x1F									default:	0x0000
		SEQ		F11F												
		ARM		0009:E23E												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	MAX0[10]	MAX0[9]	MAX0[8]	MAX0[7]	MAX0[6]	MAX0[5]	MAX0[4]	MAX0[3]	MAX0[2]	MAX0[1]	MAX0[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	MAX0	0	R/W	Base address for MPEG max 0 level tables, should be even (32-bit aligned).

5.16.33 OFLEV_MAX1TAB

MPEG Max 1 Level Table Base Address Register

OFLEV_MAX1TAB		DSP	IO	0120	offset:	0x20									default:	0x0000
		SEQ		F120												
		ARM		0009:E240												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	MAX1[10]	MAX1[9]	MAX1[8]	MAX1[7]	MAX1[6]	MAX1[5]	MAX1[4]	MAX1[3]	MAX1[2]	MAX1[1]	MAX1[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	MAX1	0	R/W	Base address for MPEG max 1 level tables, should be even (32-bit aligned).

5.16.34 CTLTAB_DCY

DC Y Control Lookup Table Base Address Register

CTLTAB_DCY		DSP	IO	0121	offset:	0x21									default:	0x0000
		SEQ		F121												
		ARM		0009:E242												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	CDCY[10]	CDCY[9]	CDCY[8]	CDCY[7]	CDCY[6]	CDCY[5]	CDCY[4]	CDCY[3]	CDCY[2]	CDCY[1]	CDCY[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	CDCY	0	R/W	This register contains the address pointer to the starting word of the Control Table DC Y lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32 bit word boundary.

5.16.35 CTLTAB_DCUV

DC UV Control Lookup Table Base Address Register

CTLTAB_DCUV		DSP	IO	0122	offset:	0x22									default:	0x0000
		SEQ		F122												
		ARM		0009:E244												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	CDCUV[10]	CDCUV[9]	CDCUV[8]	CDCUV[7]	CDCUV[6]	CDCUV[5]	CDCUV[4]	CDCUV[3]	CDCUV[2]	CDCUV[1]	CDCUV[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	CDCUV	0	R/W	This register contains the address pointer to the starting word of the Control Table DC UV lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32 bit word boundary.

5.16.36 CTLTAB_AC0

AC0 Control Lookup Table Base Address Register

CTLTAB_AC0		DSP	IO	0123	offset:	0x23									default:	0x0000
		SEQ		F123												
		ARM		0009:E246												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	CAC0[10]	CAC0[9]	CAC0[8]	CAC0[7]	CAC0[6]	CAC0[5]	CAC0[4]	0	CAC0[2]	CAC0[1]	CAC0[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	CAC0	0	R/W	This register contains the address pointer to the starting word of the Control Table DC AC0 lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32 bit word boundary.

5.16.37 CTLTAB_AC1

AC1 Control Lookup Table Base Address Register

CTLTAB_AC1		DSP	IO	0124	offset:	0x24									default:	0x0000
		SEQ		F124												
		ARM		0009:E248												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	CAC1[10]	CAC1[9]	CAC1[8]	CAC1[7]	CAC1[6]	CAC1[5]	CAC1[4]	CAC1[3]	CAC1[2]	CAC1[1]	CAC1[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	CAC1	0	R/W	This register contains the address pointer to the starting word of the Control Table DC AC1 lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32 bit word boundary.

5.16.38 OFFSET_DCY

DC Y Symbol Lookup Table Address Offset Register

OFFSET_DCY		DSP	IO	0125	offset:	0x25									default:	0x0000
		SEQ		F125												
		ARM		0009:E24A												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	ODCY[10]	ODCY[9]	ODCY[8]	ODCY[7]	ODCY[6]	ODCY[5]	ODCY[4]	ODCY[3]	ODCY[2]	ODCY[1]	ODCY[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	ODCY	0	R/W	This register contains the offset value used to address the Symbol Table DC Y lookup table. The purpose of this offset is to avoid negative numbers being used in the Control table.

5.16.39 OFFSET_DCUV

DC UV Symbol Lookup Table Address Offset Register

OFFSET_DCUV		DSP	IO	0126	offset:	0x26									default:	0x0000
		SEQ		F126												
		ARM		0009:E24C												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	ODCUV[10]	ODCUV[9]	ODCUV[8]	ODCUV[7]	ODCUV[6]	ODCUV[5]	ODCUV[4]	ODCUV[3]	ODCUV[2]	ODCUV[1]	ODCUV[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	ODCUV	0	R/W	This register contains the offset value used to address the Symbol Table DC UV lookup table. The purpose of this offset is to avoid negative numbers being used in the Control table.

5.16.40 OFFSET_AC0

AC0 Symbol Lookup Table Address Offset Register

OFFSET_AC0		DSP	IO	0127	offset:	0x27									default:	0x0000
		SEQ		F127												
		ARM		0009:E24E												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	OAC0[10]	OAC0[9]	OAC0[8]	OAC0[7]	OAC0[6]	OAC0[5]	OAC0[4]	OAC0[3]	OAC0[2]	OAC0[1]	OAC0[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	OAC0	0	R/W	This register contains the offset value used to address the Symbol Table DC AC0 lookup table. The purpose of this offset is to avoid negative numbers being used in the Control table.

5.16.41 OFFSET_AC1

AC1 Symbol Lookup Table Address Offset Register

OFFSET_AC1		DSP	IO	0128	offset:	0x28									default:	0x0000
		SEQ		F128												
		ARM		0009:E250												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	OAC1[10]	OAC1[9]	OAC1[8]	OAC1[7]	OAC1[6]	OAC1[5]	OAC1[4]	OAC1[3]	OAC1[2]	OAC1[1]	OAC1[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	OAC1	0	R/W	This register contains the offset value used to address the Symbol Table DC AC1 lookup table. The purpose of this offset is to avoid negative numbers being used in the Control table.

5.16.42 SYMTAB_DCY

DC Y Symbol Lookup Table Base Address Register

SYMTAB_DCY		DSP	IO	0129	offset:	0x29									default:	0x0000
		SEQ		F129												
		ARM		0009:E252												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	SDCY[10]	SDCY[9]	SDCY[8]	SDCY[7]	SDCY[6]	SDCY[5]	SDCY[4]	SDCY[3]	SDCY[2]	SDCY[1]	SDCY[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	SDCY	0	R/W	This register contains the address pointer to the starting word of the Symbol Table DC Y lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32 bit word boundary.

5.16.43 SYMTAB_DCUV

DC UV Symbol Lookup Table Base Address Register

SYMTAB_DCUV		DSP	IO	012A	offset:	0x2A									default:	0x0000
		SEQ		F12A												
		ARM		0009:E254												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	SDCUV[10]	SDCUV[9]	SDCUV[8]	SDCUV[7]	SDCUV[6]	SDCUV[5]	SDCUV[4]	SDCUV[3]	SDCUV[2]	SDCUV[1]	SDCUV[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	SDCUV	0	R/W	This register contains the address pointer to the starting word of the Symbol Table DC UV lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32 bit word boundary.

5.16.44 SYMTAB_AC0

AC0 Symbol Lookup Table Base Address Register

SYMTAB_AC0		DSP	IO	012B	offset:	0x2B									default:	0x0000
		SEQ		F12B												
		ARM		0009:E256												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	SAC0[10]	SAC0[9]	SAC0[8]	SAC0[7]	SAC0[6]	SAC0[5]	SAC0[4]	SAC0[3]	SAC0[2]	SAC0[1]	SAC0[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	SAC0	0	R/W	This register contains the address pointer to the starting word of the Symbol Table DC AC0 lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32 bit word boundary.

5.16.45 SYMTAB_AC1

AC1 Symbol Lookup Table Base Address Register

SYMTAB_AC1		DSP	IO	012C	offset:	0x2C									default:	0x0000
		SEQ		F12C												
		ARM		0009:E258												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	SAC1[10]	SAC1[9]	SAC1[8]	SAC1[7]	SAC1[6]	SAC1[5]	SAC1[4]	SAC1[3]	SAC1[2]	SAC1[1]	SAC1[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	SAC1	0	R/W	This register contains the address pointer to the starting word of the Symbol Table DC AC1 lookup table used by the UVLD. This address is a pointer into the VLCD coefficient memory. This table shall start on a 32 bit word boundary.

5.16.46 CTL

VLD Control Register

CTL		DSP	IO	012D	offset:	0x2D									default:	0x0000
		SEQ		F12D												
		ARM		0009:E25A												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	UVLD	DCYLEAD	DCUVLEAD	AC0LEAD	AC1LEAD	DCYSYMLE	DCUVSYMLE	AC0SYMLE	AC1SYMLE
R/W	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:9	RSV			Reserved
8	UVLD	0	R/W	Select UVLD algorithm » 0: Select VLD algorithm « » 1: Select RVLC algorithm «
7	DCYLEAD	0	R/W	This bit signifies that the Uvld is expecting a 1 leading Huffman Table for the DC Y Tables
6	DCUVLEAD	0	R/W	This bit signifies that the Uvld is expecting a 1 leading Huffman Table for the DC UV Tables
5	AC0LEAD	0	R/W	This bit signifies that the Uvld is expecting a 1 leading Huffman Table for the AC 0 Tables
4	AC1LEAD	0	R/W	This bit signifies that the Uvld is expecting a 1 leading Huffman Table for the AC 1 Tables
3	DCYSYMLEN	0	R/W	This bit signifies the bit length of the decoded symbol in the DC Y Symbol Table. Setting this bit to 1 signifies a 11 bit symbol, while 0 signifies a 12 bit table.
2	DCUVSYMLEN	0	R/W	This bit signifies the bit length of the decoded symbol in the DC UV Symbol Table. Setting this bit to 1 signifies a 11 bit symbol
1	AC0SYMLEN	0	R/W	This bit signifies the bit length of the decoded symbol in the AC 0 Symbol Table. Setting this bit to 1 signifies a 11 bit symbol
0	AC1SYMLEN	0	R/W	This bit signifies the bit length of the decoded symbol in the AC 1 Symbol Table. Setting this bit to 1 signifies a 11 bit symbol

5.16.47 VLD_NRBIT_DC

DC Number of Bits_all Register

VLD_NRBIT_DC		DSP	IO	012E	offset:	0x2E									default:	0x0000
		SEQ		F12E												
		ARM		0009:E25C												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	Y[4]	Y[3]	Y[2]	Y[1]	Y[0]	RSV	RSV	RSV	UV[4]	UV[3]	UV[2]	UV[1]	UV[0]
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	0	0	0	0	0	-	-	-	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:13	RSV			Reserved
12:8	Y	0	R/W	This field determines the number of bits to test for the DC Y term as input to the Uvld
7:5	RSV			Reserved
4:0	UV	0	R/W	This field determines the number of bits to test for the DC UV term as input to the Uvld

5.16.48 VLD_NRBIT_AC

AC Number of Bits Register

VLD_NRBIT_AC		DSP	IO	012F	offset:	0x2F									default:	0x0000
		SEQ		F12F												
		ARM		0009:E25E												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	AC0[4]	AC0[3]	AC0[2]	AC0[1]	AC0[0]	RSV	RSV	RSV	AC1[4]	AC1[3]	AC1[2]	AC1[1]	AC1[0]
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	0	0	0	0	0	-	-	-	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:13	RSV			Reserved
12:8	AC0	0	R/W	This field determines the number of bits to test for the AC0 term as input to the Uvld
7:5	RSV			Reserved
4:0	AC1	0	R/W	This field determines the number of bits to test for the AC1 term as input to the Uvld

5.16.49 BITS_BPTR

Bits Pointer Register

BITS_BPTR		DSP	IO	0130	offset:	0x30									default:	0x0000
		SEQ		F130												
		ARM		0009:E260												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	BPTR[3]	BPTR[2]	BPTR[1]	BPTR[0]
R/W	-	-	-	-	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:4	RSV			Reserved
3:0	BPTR	0	R/W	Number of valid bits in the 16-bit word pointed by VLCDOUT_ADDR(VLC) or VLCDIN_ADDR(VLD). Value = 0..15 example: BPTR = 5 b15 b0 BITS_WORD = VVVVV * * * * * * * * * * *

5.16.50 BITS_WORD

Bits Word Register

BITS_WORD		DSP	IO	0131	offset:	0x31									default:	0x0000
		SEQ		F131												
		ARM		0009:E262												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	WORD[15]	WORD[14]	WORD[13]	WORD[12]	WORD[11]	WORD[10]	WORD[9]	WORD[8]	WORD[7]	WORD[6]	WORD[5]	WORD[4]	WORD[3]	WORD[2]	WORD[1]	WORD[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	WORD	0	R/W	Last Bitstream at VLCDOUT_ADDR(VLC) or VLCDIN_ADDR(VLD).

5.16.51 BYTE_ALIGN

Byte Align Register

BYTE_ALIGN		DSP	IO	0132	offset:	0x32									default:	0x0000
		SEQ		F132												
		ARM		0009:E264												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	ENABLE	BIT
R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0

Bit	Name	Reset Value	R/W	Function
15:2	RSV			Reserved
1	ENABLE	0	R/W	VLC: When set to '1' , it enable byte align in current processing macro block VLD: When set to '1' , it means current processing bistream is byte aligned.
0	BIT	0	R/W	VLC: When byte align is selected, last byte is filled with this bit. Byte align is selected using bit 1 of this register. VLD: No meaning

5.16.52 HEAD_ADDR

Header Address Register

HEAD_ADDR		DSP	IO	0133	offset:	0x33									default:	0x0000
		SEQ		F133												
		ARM		0009:E266												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	HDA DDR[10]	HDA DDR[9]	HDA DDR[8]	HDA DDR[7]	HDA DDR[6]	HDA DDR[5]	HDA DDR[4]	HDA DDR[3]	HDA DDR[2]	HDA DDR[1]	HDA DDR[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	HDADDR	0	R/W	Base address for header data to be inserted, should be even (32-bit aligned).

5.16.53 HEAD_NUM

Number of Header Data Register

HEAD_NUM		DSP	IO	0134	offset:	0x34									default:	0x0000
		SEQ		F134												
		ARM		0009:E268												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	HDNUM [9]	HDNUM [8]	HDNUM [7]	HDNUM [6]	HDNUM [5]	HDNUM [4]	HDNUM [3]	HDNUM [2]	HDNUM [1]	HDNUM [0]
R/W	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:10	RSV			Reserved
9:0	HDNUM	0	R/W	Number of header data to be inserted(1-1023).

5.16.54 QIQ_CONFIG0

QIQ Configuration Register #0

QIQ_CONFIG0		DSP	IO	0135	offset:	0x35									default:	0x0000
		SEQ		F135												
		ARM		0009:E26A												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	SCAN[1]	SCAN[0]	RSV	QMATSEL[2]	QMATSEL[1]	QMATSEL[0]	RSV	IQMATSEL[2]	IQMATSEL[1]	IQMATSEL[0]	RSV	DCPREDSEL[2]	DCPREDSEL[1]	DCPREDSEL[0]
R/W	-	-	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Default	-	-	0	0	-	0	0	0	-	0	0	0	-	0	0	0

Bit	Name	Reset Value	R/W	Function
15:14	RSV			Reserved
13:12	SCAN	0	R/W	Scan Mode (VLC/VLD) » 00: Linear scan « » 01: Zig-Zag scan « » 10: Alternate-vertical scan « » 11: Alternate-horizontal scan «
11	RSV			Reserved
10:8	QMATSEL	0	R/W	Quant matrix selector (1 of 8 banks)
7	RSV			Reserved
6:4	IQMATSEL	0	R/W	Inverse Quantization matrix selector (1 of 8 banks)
3	RSV			Reserved
2:0	DCPREDSEL	0	R/W	DC predictor selector (1 of 6)

5.16.55 QIQ_CONFIG1

QIQ Configuration Register #1

QIQ_CONFIG1		DSP	IO	0136	offset:	0x36									default:	0x0000
		SEQ		F136												
		ARM		0009:E26C												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RSV	RSV	SCAN[1]	SCAN[0]	RSV	QMATSEL[2]	QMATSEL[1]	QMATSEL[0]	RSV	QMATSEL[2]	QMATSEL[1]	QMATSEL[0]	RSV	DCPREDSEL[2]	DCPREDSEL[1]	DCPREDSEL[0]
R/W	-	-	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Default	-	-	0	0	-	0	0	0	-	0	0	0	-	0	0	0

Bit	Name	Reset Value	R/W	Function
15:14	RSV			Reserved
13:12	SCAN	0	R/W	Scan Mode (VLC/VLD) +E804» 10: Alternate-vertical scan « » 11: Alternate-horizontal scan «
11	RSV			Reserved
10:8	QMATSEL	0	R/W	Quant matrix selector (1 of 8 banks)
7	RSV			Reserved
6:4	IQMATSEL	0	R/W	Inverse Quantization matrix selector (1 of 8 banks)
3	RSV			Reserved
2:0	DCPREDSEL	0	R/W	DC predictor selector (1 of 6)

5.16.56 QIQ_CONFIG2

QIQ Configuration Register #2

QIQ_CONFIG2		DSP	IO	0137	offset:	0x37									default:	0x0000
		SEQ		F137												
		ARM		0009:E26E												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	SCAN[1]	SCAN[0]	RSV	QMATSEL[2]	QMATSEL[1]	QMATSEL[0]	RSV	QMATSEL[2]	QMATSEL[1]	QMATSEL[0]	RSV	CPREDSEL[	CPREDSEL[	CPREDSEL[
R/W	-	-	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Default	-	-	0	0	-	0	0	0	-	0	0	0	-	0	0	0

Bit	Name	Reset Value	R/W	Function
15:14	RSV			Reserved
13:12	SCAN	0	R/W	Scan Mode (VLC/VLD) » 00: Linear scan « » 01: Zig-Zag scan « » 10: Alternate-vertical scan « » 11: Alternate-horizontal scan «
11	RSV			Reserved
10:8	QMATSEL	0	R/W	Quant matrix selector (1 of 8 banks)
7	RSV			Reserved
6:4	IQMATSEL	0	R/W	Inverse Quantization matrix selector (1 of 8 banks)
3	RSV			Reserved
2:0	DCPREDSEL	0	R/W	DC predictor selector (1 of 6)

5.16.57 QIQ_CONFIG3

QIQ Configuration Register #3

QIQ_CONFIG3		DSP	IO	0138	offset:	0x38									default:	0x0000
		SEQ		F138												
		ARM		0009:E270												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	SCAN[1]	SCAN[0]	RSV	QMATSEL[2]	QMATSEL[1]	QMATSEL[0]	RSV	IQMATSEL[2]	IQMATSEL[1]	IQMATSEL[0]	RSV	DCPREDSEL[2]	DCPREDSEL[1]	DCPREDSEL[0]
R/W	-	-	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Default	-	-	0	0	-	0	0	0	-	0	0	0	-	0	0	0

Bit	Name	Reset Value	R/W	Function
15:14	RSV			Reserved
13:12	SCAN	0	R/W	Scan Mode (VLC/VLD) » 00: Linear scan « » 01: Zig-Zag scan « » 10: Alternate-vertical scan « » 11: Alternate-horizontal scan «
11	RSV			Reserved
10:8	QMATSEL	0	R/W	Quant matrix selector (1 of 8 banks)
7	RSV			Reserved
6:4	IQMATSEL	0	R/W	Inverse Quantization matrix selector (1 of 8 banks)
3	RSV			Reserved
2:0	DCPREDSEL	0	R/W	DC predictor selector (1 of 6)

5.16.58 QIQ_CONFIG4

QIQ Configuration Register #4

QIQ_CONFIG4		DSP	IO	0139	offset:	0x39									default:	0x0000
		SEQ		F139												
		ARM		0009:E272												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	SCAN[1]	SCAN[0]	RSV	QMATSEL[2]	QMATSEL[1]	QMATSEL[0]	RSV	IQMATSEL[2]	IQMATSEL[1]	IQMATSEL[0]	RSV	DCPREDSEL[2]	DCPREDSEL[1]	DCPREDSEL[0]
R/W	-	-	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Default	-	-	0	0	-	0	0	0	-	0	0	0	-	0	0	0

Bit	Name	Reset Value	R/W	Function
15:14	RSV			Reserved
13:12	SCAN	0	R/W	Scan Mode (VLC/VLD) » 00: Linear scan « » 01: Zig-Zag scan « » 10: Alternate-vertical scan « » 11: Alternate-horizontal scan «
11	RSV			Reserved
10:8	QMATSEL	0	R/W	Quant matrix selector (1 of 8 banks)
7	RSV			Reserved
6:4	IQMATSEL	0	R/W	Inverse Quantization matrix selector (1 of 8 banks)
3	RSV			Reserved
2:0	DCPREDSEL	0	R/W	DC predictor selector (1 of 6)

5.16.59 QIQ_CONFIG5

QIQ Configuration Register #5

QIQ_CONFIG5		DSP	IO	013A	offset:	0x3A									default:	0x0000
		SEQ		F13A												
		ARM		0009:E274												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	SCAN[1]	SCAN[0]	RSV	QMATSEL[2]	QMATSEL[1]	QMATSEL[0]	RSV	IQMATSEL[2]	IQMATSEL[1]	IQMATSEL[0]	RSV	DCPREDSEL[2]	DCPREDSEL[1]	DCPREDSEL[0]
R/W	-	-	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Default	-	-	0	0	-	0	0	0	-	0	0	0	-	0	0	0

Bit	Name	Reset Value	R/W	Function
15:14	RSV			Reserved
13:12	SCAN	0	R/W	Scan Mode (VLC/VLD) » 00: Linear scan « » 01: Zig-Zag scan « » 10: Alternate-vertical scan « » 11: Alternate-horizontal scan «
11	RSV			Reserved
10:8	QMATSEL	0	R/W	Quant matrix selector (1 of 8 banks)
7	RSV			Reserved
6:4	IQMATSEL	0	R/W	Inverse Quantization matrix selector (1 of 8 banks)
3	RSV			Reserved
2:0	DCPREDSEL	0	R/W	DC predictor selector (1 of 6)

5.16.60 VLD_ERRCTL

VLD Error Control Register

VLD_ERRCTL		DSP	IO	013B	offset:	0x3B									default:	0x0004
		SEQ		F13B												
		ARM		0009:E276												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	EOB	RSV	UVLD
R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	R/W	-	R/W
Default	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	0

Bit	Name	Reset Value	R/W	Function
15:3	RSV			Reserved
2	EOB	1	R/W	EOB Error Control » 0:off « » 1:on « When an error occurs, stop huffman decoding Error occurs when EOB cannot be found during VLD
1	RSV			
0	UVLD	0	R/W	UVLD Error Control » 0:off « » 1:on « When an error occurs, stop huffman decoding. Error occurs when at least one out of table entry found in macro block

5.16.61 VLD_ERRSTAT

VLD Error Status Register

VLD_ERRSTAT		DSP	IO	013C	offset:	0x3C									default:	0x0000
		SEQ		F13C												
		ARM		0009:E278												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	FIFO	RINGBUFF	EOB	VLD	UVLD
R/W	-	-	-	-	-	-	-	-	-	-	-	R	R	R	R	R
Default	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:5	RSV			Reserved
4	FIFO	0	R	SDRAM FIFO over flow status This bit shows that SDRAM FIFO reached at max address in VLC mode.
3	RINGBUFF	0	R	Ringbuffer over flow status This bit shows that ring pointer reached at max address.
2	EOB	0	R	EOB Error » 0: No error « » 1: can‘t find the EOB in VLD «
1	VLD	0	R	VLD Error » 0: No error « » 1: found at least one non-FF00 code in macro block «
0	UVLD	0	R	UVLD Error » 0: No error « » 1: found at least 1 out of table entry in macro block «

5.16.62 RING_START

Ring Buffer Start Address Register

RING_START		DSP	IO	013D	offset:	0x3D									default:	0x0000
		SEQ		F13D												
		ARM		0009:E27A												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RSV	RSV	RSV	RSTART[12]	RSTART[11]	RSTART[10]	RSTART[9]	RSTART[8]	RSTART[7]	RSTART[6]	RSTART[5]	RSTART[4]	RSTART[3]	RSTART[2]	RSTART[1]	RSTART[0]
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:13	RSV			Reserved
12:0	RSTART	0	R/W	Ring buffer start address. VLC: When output pointer reaches end address, output pointer starts at start address automatically. VLD: When input pointer reaches end address, input pointer starts at start pointer automatically.

5.16.63 RING_END

Ring Buffer End Address Register

RING_END		DSP	IO	013E	offset:	0x3E									default:	0x0000
		SEQ		F13E												
		ARM		0009:E27C												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	REND[12]	REND[11]	REND[10]	REND[9]	REND[8]	REND[7]	REND[6]	REND[5]	REND[4]	REND[3]	REND[2]	REND[1]	REND[0]
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:13	RSV			Reserved
12:0	REND	0	R/W	Ring buffer end address. VLC: When output pointer reaches end address, output pointer starts at start address automatically. VLD: When input pointer reaches end address, input pointer starts at start pointer automatically.

5.16.64 VLD_PREFIX_DC

VLD DC prefix

VLD_PREFIX_DC		DSP	IO	013F	offset:	0x3F									default:	0x0000
		SEQ		F13F												
		ARM		0009:E27E												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	BITS_DC_Y	BITS_DC_Y	BITS_DC_Y	BITS_DC_Y	BITS_DC_Y	RSV	RSV	RSV	ITS_DC_UV	ITS_DC_UV	ITS_DC_UV	ITS_DC_UV	ITS_DC_UV
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	0	0	0	0	0	-	-	-	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:13	RSV			Reserved
12:8	NBITS_DC_Y	0	R/W	the number of prefix bits of control table for DC_Y term as input to UVLD
7:5	RSV			Reserved
4:0	NBITS_DC_UV	0	R/W	the number of prefix bits of control table for DC_UV term as input to UVLD

5.16.65 VLD_PREFIX_AC

VLD AC prefix

VLD_PREFIX_AC		DSP	IO	0140	offset:	0x40									default:	0x0000
		SEQ		F140												
		ARM		0009:E280												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	BITS_AC0[4	BITS_AC0[3	BITS_AC0[2	BITS_AC0[1	BITS_AC0[0	RSV	RSV	RSV	BITS_AC1[4	BITS_AC1[3	BITS_AC1[2	BITS_AC1[1	BITS_AC1[0
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	0	0	0	0	0	-	-	-	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:13	RSV			Reserved
12:8	NBITS_AC0	0	R/W	the number of prefix bits of control table for AC0 term as input to UVLD
7:5	RSV			Reserved
4:0	NBITS_AC1	0	R/W	the number of prefix bits of control table for AC1 term as input to UVLD

5.16.66 CTRL

VLCD control

CTRL		DSP	IO	0141	offset:	0x41									default:	0x0000
		SEQ		F141												
		ARM		0009:E282												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	ENDIAN	CLKON
R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0

Bit	Name	Reset Value	R/W	Function
15:2	RSV			Reserved
1	ENDIAN	0	R/W	Endian Control for VLC/VLD » 0: BIG Endian byte address 1->0->3->2 « » 1: LITTLE Endian byte address 0->1->2->3 «
0	CLKON	0	R/W	Dynamic Clock Controller » 0: ON « » 1: OFF «

5.16.67 SDRAM_ADDRH

Higher word of SDRAM address

SDRAM_ADDRH		DSP	IO	0142	offset:	0x42									default:	0x0000
		SEQ		F142												
		ARM		0009:E284												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	ADDRH[6]	ADDRH[5]	ADDRH[4]	ADDRH[3]	ADDRH[2]	ADDRH[1]	ADDRH[0]
R/W	-	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:7	RSV			Reserved
6:0	ADDRH	0	R/W	bits[22:16] of SDRAM address Note: SDRAM_ADDRH and SDRAM_ADDRL registers are updated upon completion of every VLC task (that was configured for direct DMA to SDRAM and actually writes to SDRAM)

5.16.68 SDRAM_ADDRL

Lower word of SDRAM address

SDRAM_ADDRL		DSP	IO	0143	offset:	0x43									default:	0x0000
		SEQ		F143												
		ARM		0009:E286												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	ADDRL[15]	ADDRL[14]	ADDRL[13]	ADDRL[12]	ADDRL[11]	ADDRL[10]	ADDRL[9]	ADDRL[8]	ADDRL[7]	ADDRL[6]	ADDRL[5]	ADDRL[4]	ADDRL[3]	ADDRL[2]	ADDRL[1]	ADDRL[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	ADDRL	0	R/W	bits[15:0] of SDRAM address Note: SDRAM_ADDRH and SDRAM_ADDRL registers are updated upon completion of every VLC task (that was configured for direct DMA to SDRAM and actually writes to SDRAM)

5.17 VLC – Huffman Table Configuration

5.17.1 VLC – Huffman Table Configuration

The table below provides more information on Huffman table configuration for VLC (encoding).

Offset Addr	*Register*	*Bit(s)*	*Description*
0x02	MODE	10	Flag to indicate the picture coding type. picture_coding_type = 1 => D-picture in MPEG-1.

		6	Flag to indicate the coding table selection, in the range of [0, 1]. The following rules are applied to determine value of this flag:
			IF (H263 || MPEG1)
			intra_vlc_format = 0;
			IF (MPEG4 && DC tables are used for DC coding) intra_vlc_format = 1;
			IF (MPEG4 && AC tables are used for DC coding)
			intra_vlc_format = 0;

0x36	Code Table DC Y Pointer		Starting address of the DC coding table for luminance. JPEG: each entry is in 32-bit, the first 16 MSBs are used to store the code length, the 16 LSBs are used to store the code word. MPEG: each entry is 16-bit, the first 5 MSBs are used to store the code length, the 11 LSBs are used to store the code word.
0x38	Code Table DC UV Pointer		Starting address of the DC coding table for chrominance. JPEG: each entry is in 32-bit, the first 16 MSBs are used to store the code length, the 16 LSBs are used to store the code word. MPEG: each entry is 16-bit, the first 5 MSBs are used to store the code length, the 11 LSBs are used to store the code word.
0x3A	Code Table AC0 Pointer		Starting address of the AC coding table 0 JPEG: each entry is in 32-bit, the first 16 MSBs are used to store the code length, the 16 LSBs are used to store the code word. MPEG: each entry is 16-bit, the first 5 MSBs are used to store the code length, the 11 LSBs are used to store the code word.
0x3C	Code Table AC1 Pointer		Starting address of the AC coding table 1 JPEG: each entry is in 32-bit, the first 16 MSBs are used to store the code length, the 16 LSBs are used to store the code word. MPEG: each entry is 16-bit, the first 5 MSBs are used to store the code length, the 11 LSBs are used to store the code word.

0x3E	Offset Level Max 0 Pointer		Pointer to array of 64 (for MPEG-1) or 128 (for H.263/MPEG-4) elements, in the range of [0x000 - 0xFFFF].
			Each entry has 16 bits, the 8 MSBs are used to store the (offset, levelmax), which is used for addressing the AC coding table and deciding coding mode chosen for a coding symbol
			
			Let (run, level) or (last, run, level) be a coding symbol in MPEG1 and H.263/MPEG-4, respectively. The following steps demonstrate how this field is utilized during the coding:
			IF (H263 || MPEG4) run +=64*last;
			offset = offset_levelmax0[run]>>8;
			levelmax = offset_levelmax0[run]&0xff;
			IF(level >levelmax) the symbol is coded in ESCAPE mode
			ELSE codtab_ac0[offset+level] is used for encoding the symbol in normal mode

0x40	Offset Level Max 1 Pointer		Pointer to array of 64 (for MPEG-1) or 128 for (H.263/MPEG-4) elements, in the range of [0x000 - 0xFFFF].
			Each entry has 16 bits, the 8 MSBs are used to store the (offset, levelmax), which is used for addressing the AC coding table and deciding coding mode chosen for a coding symbol
			
			Let (run, level) or (last, run, level) be a coding symbol in MPEG1 and H.263/MPEG-4, respectively. The following steps demonstrate how this field is utilized during the coding:
			IF (H263 || MPEG4) run +=64*last;
			offset = offset_levelmax1[run]>>8;
			levelmax = offset_levelmax1[run]&0xff;
			IF(level >levelmax) the symbol is coded in ESCAPE mode
			ELSE codtab_ac1[offset+level] is used for encoding the symbol in normal mode

5.17.2 VLC – Huffman Table Configuration Example – MPEG-1

The table below provides more information on Huffman table configuration for VLC (encoding) of MPEG-1

Addr	*Register*	*Bit(s)*	*Description*
0x02	MODE	10	0: I, P, B picture coding
		6	0: H.263 or MPEG-1 or MPEG-4 and AC tables used for DC coding
0x36	Code Table DC Y Pointer		1804 1000 1001 1805 1806 200e 281e 303e 387e 40fe 49fe 49ff
0x38	Code Table DC UV Pointer		1000 1001 1002 1806 200e 281e 303e 387e 40fe 49fe 53fe 53ff
0x3A	Code Table AC0 Pointer		0000 1003 2004 2805 3806 4026 4021 500a 601d 6018 6013 6010 681a 6819 6818 6817 701f 701e 701d 701c 701b 701a 7019 7018 7017 7016 7015 7014 7013 7012 7011 7010 7818 7817 7816 7815 7814 7813 7812 7811 7810 1803 3006 4025 500c 601b 6816 6815 781f 781e 781d 781c 781b 781a 7819 8013 8012 8011 8010 2005 3804 500b 6014 6814 2807 4024 601c 6813 2806 500f 6012 3007 5009 6812 3005 601e 8014 3004 6015 3807 6011 3805 6811 4027 6810 4023 801a 4022 8019 4020 8018 500e 8017 500d 8016 5008 8015 601f 601a 6019 6017 6016 681f 681e 681d 681c 681b 801f 801e 801d 801c 801b
0x3C	Code Table AC1 Pointer		0000 1003 2004 2805 3806 4026 4021 500a 601d 6018 6013 6010 681a 6819 6818 6817 701f 701e 701d 701c 701b 701a 7019 7018 7017 7016 7015 7014 7013 7012 7011 7010 7818 7817 7816 7815 7814 7813 7812 7811 7810 1803 3006 4025 500c 601b 6816 6815 781f 781e 781d 781c 781b 781a 7819 8013 8012 8011 8010 2005 3804 500b 6014 6814 2807 4024 601c 6813 2806 500f 6012 3007 5009 6812 3005 601e 8014 3004 6015 3807 6011 3805 6811 4027 6810 4023 801a 4022 8019 4020 8018 500e 8017 500d 8016 5008 8015 601f 601a 6019 6017 6016 681f 681e 681d 681c 681b 801f 801e 801d 801c 801b
0x3E	Offset Level Max 0 Pointer		0028 2812 3a05 3f04 4303 4603 4903 4c02 4e02 5002 5202 5402 5602 5802 5a02 5c02 5e02 6001 6101 6201 6301 6401 6501 6601 6701 6801 6901 6a01 6b01 6c01 6d01 6e01 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x40	Offset Level Max 1 Pointer		0028 2812 3a05 3f04 4303 4603 4903 4c02 4e02 5002 5202 5402 5602 5802 5a02 5c02 5e02 6001 6101 6201 6301 6401 6501 6601 6701 6801 6901 6a01 6b01 6c01 6d01 6e01 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

5.18 VLC – Header Insertion

5.18.1 VLC – Header Insertion Configuration

Addr	*Register*	*Bit(s)*	*Description*
0x66	HEAD_ADDR	10:0	Specifies the start address of bitstream to be appended. This macro block header should be stored in Huffman table (0xB800h-0xBFFFh), and HEAD_ADDR represents a relative address (0x0000h-0x07FF) within this table. Each entry is in 32-bit, the 16 MSBs are used to store the code length and the 16 LSBs are used to store the code word.
0x68	HEAD_NUM	9:0	Specifies the total number of entries. When a non-zero value is written to this register, the stored code word is inserted in the bitstream before starting the DCT block Huffman encoding.

5.18.2 VLC – Header Insertion Example

Addr	*Register*	*Bit(s)*	*Description*
0x66	HEAD_ADDR	10:0	0x700 → header starts at 0xB800+0x0700 = 0xBF00
0x68	HEAD_NUM	5:0	3

Addr	*Code Length*	*Code Word*
0xBF00	0x0003	0x0006 → 110
0xBF02	0x0004	0x0009 → 1001
0xBF04	0x0002	0x0002 → 10

The output bitstream is:

Previous bitstream	*Macroblock header*	*Current DCT bitstream*
---	110,1001,10	***

5.19 VLD – Huffman Table Configuration

5.19.1 VLD – Huffman Table Configuration

The table below provides more information on Huffman table configuration for VLD (decoding).

Addr	*Register*	*Bit(s)*	*Description*
0x02	MODE	10	Flag to indicate the picture coding type. picture_coding_type = 4 => D-picture in MPEG-1.

		6	Flag to indicate the coding table selection, in the range of [0, 1]. The following rules are applied to determine value of this flag:
			IF (H263 || MPEG1)
			intra_vlc_format = 0;
			IF (MPEG4 && DC tables are used for DC coding) intra_vlc_format = 1;
			IF (MPEG4 && AC tables are used for DC coding)
			intra_vlc_format = 0;

0x42	DC Y Control Table Pointer		Starting address of the DC decoding table for luminance. MPEG: each entry is 16-bit, the first 5 MSBs are used to store the "shift", the 11 LSBs are used to store the "offset"
0x44	DC UV Control Table Pointer		Starting address of the DC decoding table for chrominance. MPEG: each entry is 16-bit, the first 5 MSBs are used to store the "shift", the 11 LSBs are used to store the "offset"
0x46	AC0 Control Table Pointer		Starting address of the AC decoding table 0. MPEG: each entry is 16-bit, the first 5 MSBs are used to store the "shift", the 11 LSBs are used to store the "offset"
0x48	AC1 Control Table Pointer		Starting address of the AC decoding table 1. MPEG: each entry is 16-bit, the first 5 MSBs are used to store the "shift", the 11 LSBs are used to store the "offset"
0x4A	Control Table offset DC Y		16-bit global offset for the DC decoding control table of luminance
0x4C	Control Table offset DC UV		16-bit global offset for the DC decoding control table of chrominance
0x4E	Control Table offset AC 0		16-bit global offset for the AC decoding control table 0
0x50	Control Table offset AC 1		16-bit global offset for the AC decoding control table 1
0x52	Symbol table DC Y pointer		Starting address of the DC decoding table for luminance, each entry is in 16-bit, the first 4/5 MSBs contain the code length, 12/11LSBs contain the code symbol.
0x54	Symbol table DC UV pointer		Starting address of the DC decoding table for chrominance, each entry is 16-bits, the first 4/5 MSBs contain the code length, 12/11 LSBs contain the code symbol.
0x56	Symbol table AC 0 pointer		Starting address of the AC decoding table 0, each entry is 16-bits, the first 4/5 MSBs contain the code length, the 12/11 LSBs contain the code symbol.
0x58	Symbol table AC 1 pointer		Starting address of the AC decoding table 1, each entry is in 16-bit, the first 4/5 MSBs contain the code length, the 12/11 LSBs contain the code symbol.
0x5A	VLD_CTL	7	Flag to indicate the type of DC Y decoding table, 1: one leading table, 0: zero leading table
		6	Flag to indicate the type of DC UV decoding table, 1: one leading table, 0: zero leading table
		5	Flag to indicate the type of AC 0 decoding table, 1: one leading table, 0: zero leading table
		4	Flag to indicate the type of AC 1 decoding table, 1: one leading table, 0: zero leading table
		3	Number of LSBs used for storing the code symbol in each 16-bit entry of the luminance DC decoding table "Symbol Table DC Y". 0:12bit, 1:11bit
		2	Number of LSBs used for storing the code symbol in each 16-bit entry of the luminance DC decoding table "Symbol Table DC UV". 0:12bit, 1:11bit
		1	Number of LSBs used for storing the code symbol in each 16-bit entry of the luminance DC decoding table "Symbol Table AC 0". 0:12bit, 1:11bit
		0	Number of LSBs used for storing the code symbol in each 16-bit entry of the luminance DC decoding table "Symbol Table AC 1". 0:12bit, 1:11bit
0x5C	VLD_NRBIT_DC	12:8	Max code length of the luminance DC decoding table, in the range of [1-16].
		4:0	Max code length of the chrominance DC decoding table, in the range of [1-16].
0x5E	VLD_NRBIT_AC	12:8	Max code length of the luminance AC decoding table 0, in the range of [1-16].
		4:0	Max code length of the chrominance AC decoding table 1, in the range of [1-16].

5.19.2 VLD – Huffman Table Example – MPEG-1

The table below provides more information on Huffman table configuration for VLD (decoding).

Addr	*Register*	*Bit(s)*	*Description*
0x02	MODE	10	0: I, P, B picture coding
		6	0: H.263 or MPEG-1 or MPEG-4 and AC tables used for DC coding
0x42	DC Y Control Table Pointer		0000 0000 0801 1002 1803 2004 2805 3006 3006 3808
0x44	DC UV Control Table Pointer		0000 0000 0801 1002 1803 2004 2805 3006 3807 4008 4008
0x46	AC0 Control Table Pointer		0000 0000 0000 0000 0000 0000 0810 1020 1830 2040 3058 5067 4865 5069 4051 608d 7093
0x48	AC1 Control Table Pointer		0000 0000 0000 0000 0000 0000 0810 1020 1830 2040 3058 5067 4865 5069 4051 608d 4015
0x4A	Control Table offset DC Y		0
0x4C	Control Table offset DC UV		0
0x4E	Control Table offset AC 0		0
0x50	Control Table offset AC 1		0
0x52	Symbol table DC Y pointer		480b 480a 4009 3808 3007 2806 2005 1804 1803 1800 1002 1001
0x54	Symbol table DC UV pointer		500b 500a 4809 4008 3807 3006 2805 2004 1803 1002 1001 1000
0x56	Symbol table AC 0 pointer		0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 8052 8051 8050 804f 8183 8402 83c2 8382 8342 8302 82c2 87c1 8781 8741 8701 86c1 7828 7827 7826 7825 7824 7823 7822 7821 7820 784e 784d 784c 784b 784a 7849 7848 701f 701e 701d 701c 701b 701a 7019 7018 7017 7016 7015 7014 7013 7012 7011 7010 6a82 6a42 6943 68c4 6885 6847 6846 680f 680e 680d 680c 6e81 6e41 6e01 6dc1 6d81 600b 6202 6103 600a 6084 61c2 6541 6501 6009 64c1 6481 6045 60c3 6008 6182 6441 5401 5142 5007 5083 5044 53c1 5381 5102 3058 3882 3a41 3804 3a01 31c1 3181 3042 3141 4341 4006 4301 42c1 40c2 4043 4005 4281 2803 2803 2803 2803 2803 2803 2803 2803 2901 2901 2901 2901 2901 2901 2901 2901 28c1 28c1 28c1 28c1 28c1 28c1 28c1 28c1 2002 2081 1841 1841 17d0 1001

Addr	*Register*	*Bit(s)*	*Description*
0x58	Symbol table AC 1 pointer		0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 8052 8051 8050 804f 8183 8402 83c2 8382 8342 8302 82c2 87c1 8781 8741 8701 86c1 7828 7827 7826 7825 7824 7823 7822 7821 7820 784e 784d 784c 784b 784a 7849 7848 701f 701e 701d 701c 701b 701a 7019 7018 7017 7016 7015 7014 7013 7012 7011 7010 6a82 6a42 6943 68c4 6885 6847 6846 0000 0000 0000 0000 6e81 6e41 6e01 6dc1 6d81 0000 6202 6103 0000 0000 61c2 6541 6501 0000 64c1 6481 0000 60c3 0000 6182 6441 4942 4942 4b81 4b81 5084 5401 4bc1 4bc1 3058 39c1 3a01 3981 3882 3007 3006 3101 3141 4045 42c1 400b 400a 4341 4301 40c2 4044 2881 2881 2881 2881 2881 2881 2881 2881 2842 2842 2842 2842 2842 2842 2842 2842 28c1 28c1 28c1 28c1 28c1 28c1 28c1 28c1 1841 1841 27d0 2003 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 1802 2804 2804 2804 2804 2804 2804 2804 2804 2805 2805 2805 2805 2805 2805 2805 2805 3a41 3a41 3843 3843 3a81 3a81 3808 3808 3809 3809 400c 400d 4083 4102 400e 400f
0x5A	VLD_CTL	7	Flag to indicate the type of DC Y decoding table, **1**: one leading table
		6	Flag to indicate the type of DC UV decoding table, **1**: one leading table
		5	Flag to indicate the type of AC 0 decoding table, **0**: zero leading table
		4	Flag to indicate the type of AC 1 decoding table, **0**: zero leading table
		3	Number of LSBs used for storing the code symbol in each 16-bit entry of the luminance DC decoding table “Symbol Table DC Y”. **1**:11bit
		2	Number of LSBs used for storing the code symbol in each 16-bit entry of the luminance DC decoding table “Symbol Table DC UV”. **1**:11bit
		1	Number of LSBs used for storing the code symbol in each 16-bit entry of the luminance DC decoding table “Symbol Table AC 0”. **1**:11bit
		0	Number of LSBs used for storing the code symbol in each 16-bit entry of the luminance DC decoding table “Symbol Table AC 1”. **1**:11bit
0x5C	VLD_NRBIT_DC	12:8	0x09
		4:0	0x0A
0x5E	VLD_NRBIT_AC	12:8	0x10
		4:0	0x10

5.20 QIQ – Scanning paths and quantization matrices used

5.20.1 Zigzag scanning path, used in JPEG,H.263, MPEG-1 and MPEG4

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

5.20.2 Vertical scanning path, used in MPEG4

0	4	6	20	22	36	38	52
1	5	7	21	23	37	39	53
2	8	19	24	34	40	50	54
3	9	18	25	35	41	51	55
10	17	26	30	42	46	56	60
11	16	27	31	43	47	57	61
12	15	28	32	44	48	58	62
13	14	29	33	45	49	59	63

5.20.3 Horizontal scanning path, used in MPEG4

0	1	2	3	10	11	12	13
4	5	8	9	17	16	15	14
6	7	19	18	26	27	28	29
20	21	24	25	30	31	32	33
22	23	34	35	42	43	44	45
36	37	40	41	46	47	48	49
38	39	50	51	56	57	58	59
52	53	54	55	60	61	62	63

5.20.4 Default intra quantiser matrix for MPEG1 and MPEG4

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

5.20.5 Default inter quantiser matrix for MPEG1

16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

5.20.6 Default inter quantiser matrix for MPEG4

16	17	18	19	20	21	22	23
17	18	19	20	21	22	23	24
18	19	20	21	22	23	24	25
19	20	21	22	23	24	26	27
20	21	22	23	25	26	27	28
21	22	23	24	26	27	28	30
22	23	24	26	27	28	30	31
23	24	25	27	28	30	31	33

5.21 VLC Tables used

5.21.1 VLC table for MPEG1 dct_dc_size_luminance, 12 entries

```
0x1804, 0x1000, 0x1001, 0x1805, 0x1806, 0x200e, 0x281e, 0x303e,
0x387e, 0x40fe, 0x49fe, 0x49ff
```

5.21.2 VLC table for MPEG1 dct_dc_size_chrominance, 12 entries

```
0x1000, 0x1001, 0x1002, 0x1806, 0x200e, 0x281e, 0x303e, 0x387e,
0x40fe, 0x49fe, 0x53fe, 0x53ff
```

5.21.3 VLC table for MPEG1 DCT coefficients table zero, 112 entries

```
0x0000, /* dummy code */
0x1003, 0x2004, 0x2805, 0x3806, 0x4026, 0x4021, 0x500a, 0x601d,
0x6018, 0x6013, 0x6010, 0x681a, 0x6819, 0x6818, 0x6817, 0x701f,
0x701e, 0x701d, 0x701c, 0x701b, 0x701a, 0x7019, 0x7018, 0x7017,
0x7016, 0x7015, 0x7014, 0x7013, 0x7012, 0x7011, 0x7010, 0x7818,
0x7817, 0x7816, 0x7815, 0x7814, 0x7813, 0x7812, 0x7811, 0x7810,
0x1803, 0x3006, 0x4025, 0x500c, 0x601b, 0x6816, 0x6815, 0x781f,
0x781e, 0x781d, 0x781c, 0x781b, 0x781a, 0x7819, 0x8013, 0x8012,
0x8011, 0x8010, 0x2005, 0x3804, 0x500b, 0x6014, 0x6814, 0x2807,
0x4024, 0x601c, 0x6813, 0x2806, 0x500f, 0x6012, 0x3007, 0x5009,
0x6812, 0x3005, 0x601e, 0x8014, 0x3004, 0x6015, 0x3807, 0x6011,
0x3805, 0x6811, 0x4027, 0x6810, 0x4023, 0x801a, 0x4022, 0x8019,
0x4020, 0x8018, 0x500e, 0x8017, 0x500d, 0x8016, 0x5008, 0x8015,
0x601f, 0x601a, 0x6019, 0x6017, 0x6016, 0x681f, 0x681e, 0x681d,
0x681c, 0x681b, 0x801f, 0x801e, 0x801d, 0x801c, 0x801b
```

5.21.4 VLC table for MPEG2 DCT coefficients table one, 112 entries

```
0x0000, /* dummy code */
0x1002, 0x1806, 0x2007, 0x281c, 0x281d, 0x3005, 0x3004, 0x387b,
0x387c, 0x4023, 0x4022, 0x40fa, 0x40fb, 0x40fe, 0x40ff, 0x701f,
0x701e, 0x701d, 0x701c, 0x701b, 0x701a, 0x7019, 0x7018, 0x7017,
0x7016, 0x7015, 0x7014, 0x7013, 0x7012, 0x7011, 0x7010, 0x7818,
0x7817, 0x7816, 0x7815, 0x7814, 0x7813, 0x7812, 0x7811, 0x7810,
0x1802, 0x2806, 0x3879, 0x4027, 0x4020, 0x6816, 0x6815, 0x781f,
0x781e, 0x781d, 0x781c, 0x781b, 0x781a, 0x7819, 0x8013, 0x8012,
0x8011, 0x8010, 0x2805, 0x3807, 0x40fc, 0x500c, 0x6814, 0x2807,
0x4026, 0x601c, 0x6813, 0x3006, 0x40fd, 0x6012, 0x3007, 0x4804,
0x6812, 0x3806, 0x601e, 0x8014, 0x3804, 0x6015, 0x3805, 0x6011,
0x3878, 0x6811, 0x387a, 0x6810, 0x4021, 0x801a, 0x4025, 0x8019,
0x4024, 0x8018, 0x4805, 0x8017, 0x4807, 0x8016, 0x500d, 0x8015,
0x601f, 0x601a, 0x6019, 0x6017, 0x6016, 0x681f, 0x681e, 0x681d,
0x681c, 0x681b, 0x801f, 0x801e, 0x801d, 0x801c, 0x801b
```

5.21.5 offset_levelmax table for MPEG1/2 DCT coefficients table zero and one, 64 entries

0x0028, 0x2812, 0x3a05, 0x3f04, 0x4303, 0x4603, 0x4903, 0x4c02,
0x4e02, 0x5002, 0x5202, 0x5402, 0x5602, 0x5802, 0x5a02, 0x5c02,
0x5e02, 0x6001, 0x6101, 0x6201, 0x6301, 0x6401, 0x6501, 0x6601,
0x6701, 0x6801, 0x6901, 0x6a01, 0x6b01, 0x6c01, 0x6d01, 0x6e01,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000

5.22 VLD Tables used

5.22.1 VLD decoding table for MPEG1/2 – dct_dc_size_luminance (12 entries, nrbits_dc_y=9, oneleading_dc_y=1, size_offset_global_dc_y = 0, symbol is size)

```
0x480b, 0x480a, 0x4009, 0x3808, 0x3007, 0x2806, 0x2005, 0x1804,
0x1803, 0x1800, 0x1002, 0x1001
```

5.22.2 VLD decoding control table for MPEG1/2 dct_dc_size_luminance (10 entries, nrbits_dc_y=9, oneleading_dc_y=1, size_offset_global_dc_y = 0)

```
0x0000, 0x0000, 0x0801, 0x1002, 0x1803, 0x2004, 0x2805, 0x3006,
0x3006, 0x3808
```

5.22.3 VLD decoding table for MPEG1/2 dct_dc_size_chrominance (12 entries, nrbits_dc_uv=10, oneleading_dc_uv=1, size_offset_global_dc_uv = 0, symbol is size)

```
0x500b, 0x500a, 0x4809, 0x4008, 0x3807, 0x3006, 0x2805, 0x2004,
0x1803, 0x1002, 0x1001, 0x1000
```

5.22.4 VLD decoding control table for MPEG1/2 dct_dc_size_chrominance (11 entries, nrbits_dc_uv=10, oneleading_dc_uv=1, size_offset_global_dc_uv = 0)

```
0x0000, 0x0000, 0x0801, 0x1002, 0x1803, 0x2004, 0x2805, 0x3006,
0x3807, 0x4008, 0x4008
```

5.22.5 VLD decoding table for MPEG1/2 DCT coefficients table zero (151 entries, nrbits_ac0=16, oneleading_ac0=0, offset_global_ac0 = 0, symbol is 64*run + level, symbol 0x58 denotes ESCAPE, entry value 0x07d0 means EOB)

```
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x8052, 0x8051, 0x8050, 0x804f, 0x8183, 0x8402, 0x83c2, 0x8382,
0x8342, 0x8302, 0x82c2, 0x87c1, 0x8781, 0x8741, 0x8701, 0x86c1,
0x7828, 0x7827, 0x7826, 0x7825, 0x7824, 0x7823, 0x7822, 0x7821,
0x7820, 0x784e, 0x784d, 0x784c, 0x784b, 0x784a, 0x7849, 0x7848,
0x701f, 0x701e, 0x701d, 0x701c, 0x701b, 0x701a, 0x7019, 0x7018,
0x7017, 0x7016, 0x7015, 0x7014, 0x7013, 0x7012, 0x7011, 0x7010,
0x6a82, 0x6a42, 0x6943, 0x68c4, 0x6885, 0x6847, 0x6846, 0x680f,
0x680e, 0x680d, 0x680c, 0x6e81, 0x6e41, 0x6e01, 0x6dc1, 0x6d81,
0x600b, 0x6202, 0x6103, 0x600a, 0x6084, 0x61c2, 0x6541, 0x6501,
0x6009, 0x64c1, 0x6481, 0x6045, 0x60c3, 0x6008, 0x6182, 0x6441,
0x5401, 0x5142, 0x5007, 0x5083, 0x5044, 0x53c1, 0x5381, 0x5102,
0x3058, 0x3882, 0x3a41, 0x3804, 0x3a01, 0x31c1, 0x3181, 0x3042,
0x3141, 0x4341, 0x4006, 0x4301, 0x42c1, 0x40c2, 0x4043, 0x4005,
0x4281, 0x2803, 0x2803, 0x2803, 0x2803, 0x2803, 0x2803, 0x2803,
0x2803, 0x2901, 0x2901, 0x2901, 0x2901, 0x2901, 0x2901, 0x2901,
0x2901, 0x28c1, 0x28c1, 0x28c1, 0x28c1, 0x28c1, 0x28c1, 0x28c1,
0x28c1, 0x2002, 0x2081, 0x1841, 0x1841, 0x17d0, 0x1001
```

6 Sequencer

6.1 Introduction

Sequencer is a small 16-bit microprocessor targeted for simple control, address calculation and loop control functions in a coprocessor subsystem. It is used in DM320 to coordinate among iMX, VLCD, DCT coprocessors, COP DMA controller, C54x DSP and ARM9. This document describes the architectural model in brief, more of programming model and the assembly language tools for Sequencer.

6.2 Architectural details

The Sequencer in DM320 is by itself an auxiliary processor. In general the sequencer will be used to off-load the sequential operations to ease the data processing steps. The Sequencer is simple in architecture, allowing only simple jumps and conditional based on ALU operations. Several data manipulation operators will be supported. A set of status bits reflects the change in the state of internal registers. The Sequencer also supports interrupt handling.

Figure 11 shows the block diagram of the DM320 Sequencer.

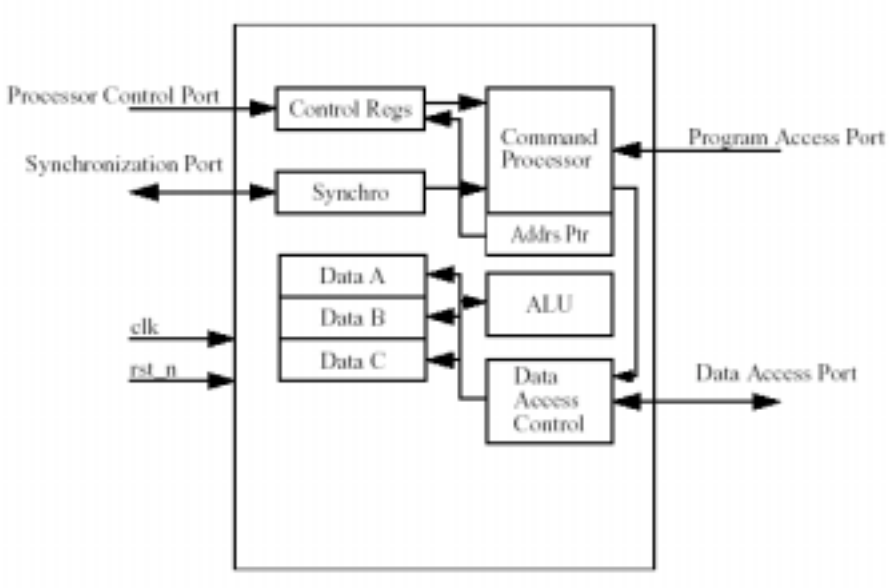


Figure 11: Sequencer Block Diagram

As shown in Figure 11 the Sequencer is communicating to the rest of the DM320 system through a set of ports. They are:

- Processor Control Port (PCP)

  The system processor (ARM or DSP) controls the operations of Sequencer through this port. This port allow operations such as:

  - Boot Address load
  - Start and Stop control
  - Program Abort
  - Internal Register read for debug

  Thus this port provides access for ARM or DSP to control sequencer flow and the ability to start and stop the sequencer.

- Program Access Port (PAP)

  Sequencer is associated with a Program RAM (PGMRAM) having two ports, one for access to the memory by the system processor and the other for access by Sequencer only. The Sequencer reads its program from PGMRAM through PAP.

  The Program Address Pointer provides the access addresses to the PAP. The Sequencer internal command processor directs the PAP to read data from the PGMRAM and the PAP acknowledges back when the operation is completed.

  This is a read only port.

- Data Access Port (DAP)

  The sequencer accesses other memory blocks and peripheral control registers through this port. Thus DAP provides operational control of other peripherals and access to computational operands for the Sequencer operations.

  This port is capable of both Read and Write. Also it supports access wait states.

- Synchronization and Interrupt Ports

  Sequencer handles a single interrupt with associated interrupt vector input as well as a single synchronization input.

- System Port

  This port consists of the clock and reset signals. The sequencer clock can be controlled by ARM or by DSP. Refer chapters on Clock Controller of ARM subsystem and Coprocessor subsystem.

  The sequencer reset is driven from the system reset.

6.3 Programming Model

6.3.1 PCP registers

As mentioned before Sequencer acts as an auxiliary processor in DM320 system, and thus has a processor control port for the system processor to control its execution. Sequencer Register Map (SEQ) section of this document gives the details of the various PCP port registers.

6.3.2 The normal execution flow

Before execution the Sequencer program needs to be present in the PGMRAM. One port of the PGMRAM is connected to PAP of Sequencer while other port is connected to the system processor network to load the Sequencer program into the PGMRAM. ARM, DSP or COP DMAC can load the PGMRAM with Sequencer Program.

The normal execution flow of the Sequencer is

Idle -> run -> busy -> stop -> idle

The Sequencer starts execution when the RUN bit of **CTRL** register is written '1', and while executing, the RUN bit will be '1' to indicate BUSY status. Upon executing STOP instruction, sequencer turns into idle state and the read back value of RUN bit becomes 0. The address from which Sequencer starts execution is specified in the BOOT register.

Right after run is issued, the contents of BOOT register is transferred into internal register and thus the BOOT register can be modified without affecting the on-going execution.

If necessary, Sequencer can be aborted from normal execution, by writing 1 to the ABORT bit of **CTRL** register. Sequencer will abort even when it's held by the SYNC instruction. After the abort the RUN bit becomes '0'. To restart it, the Abort control bit needs to be explicitly written 0, and the RUN bit be written 1.

Note: The ABORT bit is a write only bit and will read back '0' always.

Note: The BOOT register is not mirrored and thus this should be changed if it is certain that the BOOT operation has occurred. The scenario where this might cause a problem - if sequencer is started, but the PGMRAM is busy servicing another access, the Sequencer will be held off. If the BOOT address is changed before the read command is acknowledged, this will change where the Sequencer boots.

6.3.3 Sequencer Internal Registers

The Sequencer has the following internal registers

General purpose register A, 16-bit

General purpose register B, 16-bit

General purpose register C, 16-bit

General purpose register P, 16-bit.

A, B, C, and P are accessible by all ALU instructions as operands. P can also be used for a pointer to a memory operand, by specifying as *P or *P++ in the assembly.

Program Counter PC, 16-bit

Status register ST, consisting of:

- Z: Zero flag, true when last ALU or Load outcome is zero
- N: Negative flag, true when the last ALU or Load outcome is negative (bit 15 being 1)
- C: Carry flag, true when the last ALU operation generates carry. Load operation always clears this flag.
- V: Overflow, true when the last ALU operation overflows. Load operation always clears this flag.

The ALU or Load operation refers to the main data operation indicated in the instruction, not any address increment arithmetic. Each execution of ALU/Load instruction always updates all 4 flags. AND instruction, for example, never overflows or generates carries, and thus always clears C and V flags.

In unsigned addition, carry flag indicates overflow. For example, 0x8000 + 0x9000 activates overflow flag.

In unsigned subtraction, carry flag indicates zero or positive result. For example, 0x9200 – 0x1100 is carried out as 0x9200 + 2's compl(0x1100) = 0x9200 + (0xEEFF + 1) = 0x18100, result = lower 16-bit = 0x8100, and carry = bit 16 = 1.

In signed addition and subtraction, overflow flag indicates overflow, which is defined as two same-signed operands of addition resulting in opposite-signed sum. Subtraction is converted adding the 2's complement of second operand before applying the overflow test. Two opposite-sign addition operands will never overflow. For example, 0x6000 + 0x6000 activates overflow flag.

In signed addition and subtraction, sign flag indicates whether the result is negative, provided that there's no overflow (when there's overflow the correct result cannot fit 16-bit and thus the result in the destination register is wrong). For example, 0x6000 – 0x7000 activates sign flag.

Return register RTN

RTN is automatically loaded with address of the next instruction upon any branch instruction resulting in a branch taken. RTN's content is transferred to PC upon executing an RTN (return) instruction. RTN also can be accessed by STRTN Instruction, which transfers RTN's content to a memory location.

Supporting registers for interrupt handling consist of

- Interrupt return register (PC_I)
- Interrupt shadow register for RTN (RTN_I)
- Interrupt shadow register for ST (ST_I)

Upon interrupt, PC is saved in PC_I, RTN is saved in RTN_I, and ST is saved in ST_I. Upon executing an RTI instruction, PC, RTN, and ST are restored.

Note: These supporting registers are not accessible using any instructions. These are working registers for Sequencer's operation.

6.3.4 Instruction Set and Encoding

Instruction set and encoding summary is as shown in Table 19, Table 20, Table 21 and Table 22 below.

15...14	13……10	9...8	7	6…. 4	3	2…..0
class	opcode	dst	ex1	src2	ex2	src1

16b
Optional imm16 or addr16

Table 19: Normal Format

15...14	13……10	9...8	7……0
class	opcode	dst	imm8

Table 20: Immediate 8-bit format

Flow control (class=00)	Data movement (class=01)	Arithmetic/Logic (class=10)	Branch (class=11)
0000: NOP 0001: SYNC 1111: STOP	0001: ST 0010: LD 1001: STINC * 1010: LDINC * 1011: STRTN	0000: ADD 0001: SUB 0010: AND 0011: OR 0100: XOR 0101: LSH 0110: ASH 1000: NORM 1001: NOT	0000: B (imm8 offset) 0001: BV (imm8 offset) 0010: BZ (imm8 offset) 0011: BNZ (imm8 offset) 0100: BC (imm8 offset) 0101: BNC (imm8 offset) 0110: BN (imm8 offset) 0111: BPZ (imm8 offset) 1000: BRI (reg absolute) 1001: RTN 1010: RTI

Table 21: Class/Opcode

src1/src2 for ALU	**src1 for LD, LDINC**	**src1 for ST, STINC**	**src1 for BRI**	**dst for ALU, LD, LDINC**	**dst for ST, STINC, STRTN**
000: A 001: B 010: C 011: P 100: addr16 (src1 only) 110: imm16 (src1 only) 111: *P (src1 only)	100: addr16 110: imm16 111: *P or *P++	000: A 001: B 010: C 011: P	000: A 001: B 010: C 011: P 101: imm16	00: A 01: B 10: C 11: P	00: addr16 01: *P or *P++

Table 22: Src/Dst fields

Note: The Sequencer assembler does not recognize LDINC and STINC as opcodes. They should be coded as LD/ST with *P++ as the memory operand. For example,

```
LD    *P++, A
ST    B, *P++
```

6.4 Assembly mnemonic

The following notations are used:

A/B/C/P: Sequencer internal registers

*P: memory operand pointed by P

*P++: memory operand pointed by P, and with the execution P+1 → P

PC: program counter

RTN: return address register

addr16: memory operand pointed by the 16-bit address coded as the 2nd instruction word

imm16: 16-bit immediate value coded as the 2nd instruction word

imm8: 8-bit immediate value coded as part of the 1st instruction word in immediate 8-bit format.

The instruction cycle count assumes that data memory read/write is arbitrated with instruction access and thus causes one wait state. Wait states for accessing non-local-memory location depends on the bus interface and that location (hardware module registers typically), and for simplicity is assumed to be 1 wait state as well.

6.4.1 ADD

Syntax	**ADD** src1, src2, dst
Operands	src1: A, B, C, P, addr16, imm16, *P src2: A, B, C, P dst: A, B, C, P
Execution	src1 + src2 → dst
Affect Status Bits	Yes
Description	This instruction calculates the sum of two operands and writes the result in the destination.
Words	2 words when addr16 or imm16 is used, otherwise 1 word

Cycles	2 cycles when addr16 or imm16 is used, otherwise 1 cycle

6.4.2 AND

Syntax	**AND** src1 ,src2, dst
Operands	src1: A, B, C, P, addr16, imm16, *P src2: A, B, C, P dst: A, B, C, P
Execution	src1 bit-wise-and src2 → dst
Affect Status Bits	Yes
Description	This instruction calculates the bit-wise AND of two operands and writes the result in the destination.
Words	2 words when addr16 or imm16 is used, otherwise 1 word
Cycles	2 cycles when addr16 or imm16 is used, otherwise 1 cycle

6.4.3 ASH

Syntax	**ASH** src1, src2, dst
Operands	src1: A, B, C, P, addr16, imm16, *P src2: A, B, C, P dst: A, B, C, P

Execution	Arithmetic shift:
	if (src2 >= 0)
	src1 << src2 → dst
	else
	src1 >> (-src2) → dst

Affect Status Bits	Yes
Description	This instruction shifts operand1 by operand2. It can be a right or left shift, depending on the sign of operand2. Sign is preserved in case of right shift.
Words	2 words when addr16 or imm16 is used, otherwise 1 word

Cycles	2 cycles when addr16 or imm16 is used, otherwise 1 cycle

6.4.4 B

Syntax	**B** addr
Operands	addr: an address within 8-bit hop, [-128, 127] of the next instruction's address. The imm8 offset is coded
Execution	PC + 1 → RTN addr → PC
Affect Status Bits	No
Description	This instruction causes execution to branch to the address in the argument. Note the reach of this branch instruction is limited by an 8-bit offset.
Words	1 word
Cycles	2 cycles

6.4.5 BC

Syntax	**BC** addr
Operands	addr: an address within 8-bit hop, [-128, 127] of the next instruction's address The imm8 offset is coded
Execution	PC + 1 → RTN addr → PC if (carry_flag == 1)
Affect Status Bits	No
Description	This instruction causes execution to branch to the address in the argument if the carry flag is set. Note the reach of this branch instruction is limited by an 8-bit offset.
Words	1 word
Cycles	2 cycles if branch taken, otherwise 1 cycle

6.4.6 BN

Syntax	**BN** addr

Operands	addr: an address within 8-bit hop, [-128, 127] of the next instruction's address The imm8 offset is coded
Execution	PC + 1 → RTN addr → PC if (negative_flag == 1)
Affect Status Bits	No
Description	This instruction causes execution to branch to the address in the argument if the negative flag is set. Note the reach of this branch instruction is limited by an 8-bit offset.
Words	1 word
Cycles	2 cycles if branch taken, otherwise 1 cycle

6.4.7 BNC

Syntax	**BNC** addr
Operands	addr: an address within 8-bit hop, [-128, 127] of the next instruction's address The imm8 offset is coded
Execution	PC + 1 → RTN addr → PC if (carry_flag == 0)
Affect Status Bits	No
Description	This instruction causes execution to branch to the address in the argument if the carry flag is clear. Note the reach of this branch instruction is limited by an 8-bit offset.
Words	1 word
Cycles	2 cycles if branch taken, otherwise 1 cycle

6.4.8 BNZ

Syntax	**BNZ** addr
Operands	addr: an address within 8-bit hop, [-128, 127] of the next instruction's address The imm8 offset is coded
Execution	PC + 1 → RTN addr → PC if (zero_flag == 0)

Affect Status Bits	No
Description	This instruction causes execution to branch to the address in the argument if the zero flag is clear. Note the reach of this branch instruction is limited by an 8-bit offset.
Words	1 word
Cycles	2 cycles if branch taken, otherwise 1 cycle

6.4.9 BRI

Syntax	**BRI** reg
Operands	reg: A, B, C, or P
Execution	PC + 1 → RTN reg → PC
Affect Status Bits	No
Description	This instruction causes execution to branch to the value in the register. Note that the 16-bit branch destination itself is in the register compared with an 8-bit relative address for the other branch instructions.
Words	1 word
Cycles	2 cycles

6.4.10 BV

Syntax	BV addr
Operands	addr: an address within 8-bit hop, [-128, 127] of the next instruction's address The imm8 offset is coded
Execution	PC + 1 → RTN addr → PC if (overflow_flag == 1)
Affect Status Bits	No
Description	This instruction causes execution to branch to the address in the argument if the overflow flag is set. Note the reach of this branch instruction is limited by an 8-bit offset.
Words	1 word
Cycles	2 cycles if branch taken, otherwise 1 cycle

6.4.11 BZ

Syntax	BZ addr
Operands	addr: an address within 8-bit hop, [-128, 127] of the next instruction's address The imm8 offset is coded
Execution	PC + 1 → RTN addr → PC if (zero_flag == 1)
Affect Status Bits	No
Description	This instruction causes execution to branch to the address in the argument if the zero flag is set. Note the reach of this branch instruction is limited by an 8-bit offset.
Words	1 word
Cycles	2 cycles if branch taken, otherwise 1 cycle

6.4.12 LD

Syntax	LD *P, dst
	LD *P++, dst
	LD addr16, dst
	LD imm16, dst

Operands	dst: A, B, C or P addr16: 16-bit address coded in the second instruction word imm16: 16-bit immediate value coded in the second instruction word
Execution	*P → dst *P → dst, P+1 → P *addr16 → dst imm16 → dst
Affect Status Bits	Yes
Description	This instruction loads a value and stores it in the destination register. The value can be in the memory location pointed by P, the memory location pointed by the second instruction word, or itself coded the second instruction word.
Words	2 words for addr16, imm16

	1 word for *P, *P++
Cycles	3 cycles for addr16, imm16
	cycles for *P, *P++

6.4.13 LSH

Syntax	**LSH** src1 ,src2, dst
Operands	src1: A, B, C, P, addr16, imm16, *P
	src2: A, B, C, P
	dst: A, B, C, P
Execution	Logical shift:
	if (src2 >= 0)
	src1 << src2 → dst
	else
	((long) (0xFFFF & src1)) >> (-src2) → dst
Affect Status Bits	Yes
Description	This instruction performs logical right/left shift of operand1 by operand2 bits. For logical right shift, zero is shifted into the MSB (as opposed to sign-extended).
Words	2 words when addr16 or imm16 is used, otherwise 1 word
Cycles	2 cycles when addr16 or imm16 is used, otherwise 1 cycle

6.4.14 NOP

Syntax	**NOP**
Operands	none
Execution	none
Affect Status Bits	No
Description	This instruction does nothing
Words	1 word
Cycles	1 cycle

6.4.15 NORM

Syntax	**NORM** src, dst
Operands	src: A, B, C, P, addr16, imm16, *P

	dst: A, B, C, P
Execution	normalize(src) → dst
Affect Status Bits	Yes
Description	This instruction finds the number of bits that the source can be left-shifted without going overflow. Thus, if the source is positive, it reports the leading 0 bits – 1. If the source is negative, it reports the leading 1 bits – 1. If the source is zero, 16 is returned.
Words	2 words when addr16 or imm16 is used, otherwise 1 word
Cycles	2 cycles when addr16 or imm16 is used, otherwise 1 cycle

6.4.16 NOT

Syntax	**NOT** src, dst
Operands	src: A, B, C, P, addr16, imm16, *P
	dst: A, B, C, P
Execution	bit-wise-not(src) → dst
Affect Status Bits	Yes
Description	This instruction calculates the (one's) complement of the source and writes the result in the destination.
Words	2 words when addr16 or imm16 is used, otherwise 1 word
Cycles	2 cycles when addr16 or imm16 is used, otherwise 1 cycle

6.4.17 OR

Syntax	**OR** src1 ,src2, dst
Operands	src1: A, B, C, P, addr16, imm16, *P
	src2: A, B, C, P
	dst: A, B, C, P
Execution	src1 bit-wise-or src2 → dst
Affect Status Bits	Yes

Description	This instruction calculates the OR of two operands and writes the result in the destination.
Words	2 words when addr16 or imm16 is used, otherwise 1 word
Cycles	2 cycles when addr16 or imm16 is used, otherwise 1 cycle

6.4.18 RTI

Syntax	**RTI**
Operands	none
Execution	PC_I → PC ST_I → ST RTN_I → RTN
Affect Status Bits	No
Description	This instruction returns to the normal processing thread by restoring PC, ST, and RTN from the interrupt-saved copies.
Words	1 word
Cycles	2 cycles

6.4.19 RTN

Syntax	**RTN**
Operands	none
Execution	RTN register → PC
Affect Status Bits	No
Description	This instruction returns to the caller from subroutine by loading PC with the contents of the RTN register, which is loaded automatically on every branch instruction
Words	1 word
Cycles	2 cycles

6.4.20 ST

Syntax	**ST** src, *P **ST** src, *P++ **ST** src, addr16
Operands	src: A, B, C or P

	addr16: 16-bit address coded in the second instruction word
Execution	src → *P
	src → *P, P+1 → P
	src → *addr16
Affect Status Bits	No
Description	This instruction stores a register value into a destination location. The location can be pointed by P (with optional post increment on P), or coded in the second instruction word.
Words	2 words for addr16
	1 word for *P, *P++
Cycles	3 cycles for addr16
	2 cycles for *P, *P+

6.4.21 STOP

Syntax	**STOP**
Operands	none
Execution	Stop execution
Affect Status Bits	No
Description	This instruction stops Sequencer execution
Words	1 word
Cycles	1 cycle

6.4.22 STRTN

Syntax	**STRTN** *P
	STRTN addr16
Operands	addr16: 16-bit address coded in the second instruction word
Execution	RTN → *P
	RTN → *addr16
Affect Status Bits	No
Description	This instruction stores the RTN register value into a destination location. The location can be the pointed by P, or coded in the second instruction word.

Words	2 words for addr16
	1 word for *P
Cycles	3 cycles for addr1
	6 cycles for *P

6.4.23 SUB

Syntax	**SUB** src1 ,src2, dst
Operands	src1: A, B, C, P, addr16, imm16, *P
	src2: A, B, C, P
	dst: A, B, C, P
Execution	src1 – src2 → dst
Affect Status Bits	Yes
Description	This instruction calculates the difference of two operands and writes the result in the destination.
Words	2 words when addr16 or imm16 is used, otherwise 1 word
Cycles	2 cycles when addr16 or imm16 is used, otherwise 1 cycle

6.4.24 SYNC

Syntax	SYNC
Operands	none
Execution	Pause until SYNC==1
Affect Status Bits	No
Description	This instruction pauses Sequencer execution until the SYNC signal input , typically from an interrupt controller external to the Sequencer, becomes 1.
Words	1 word
Cycles	N cycles (until synchronization achieved)

6.4.25 XOR

Syntax	**XOR** src1 ,src2, dst
Operands	src1: A, B, C, P, addr16, imm16, *P
	src2: A, B, C, P

	dst: A, B, C, P
Execution	src1 bit-wise-exclusive-or src2 → dst
Affect Status Bits	Yes
Description	This instruction calculates the exclusive or result of two operands and writes the result in the destination.
Words	2 words when addr16 or imm16 is used, otherwise 1 word
Cycles	2 cycles when addr16 or imm16 is used, otherwise 1 cycle\

6.4.26 Cycle count

Addr16:

Cycle1 opcode fetch

Cycle2 addr fetch

Cycle3 data fetch (execute)

Imm16:

Cycle1 opcode fetch

Cycle2 data fetch (execute)

*P:

cycle1 opcode fetch (data fetch)

cycle2 data fetch (execute)

Any branch. (B,BN,BNZ,...) will take only 1 cycle. Operations that load data (ld,add,sub,.....) require multiple cycles as shown above. Therefore:

Addr16 = 3

Immed16 = 2

*P = 2

Any register to register operation takes only 1 cycle.

Eg. ADD A,B,C

Store operations are different, the Sequencer does not have to wait for the data to arrive to execute the command:

Addr16 = 2

*P = 1

This of course must be qualified by the absence of wait state insertion by an external source.

6.5 Sync input for the Sequencer

SYNC is a synchronous input, edge synchronized with Sequencer clock. It is similar to an interrupt in controlling the operational flow of the sequencer, it differs in that the sync state must be commanded by the execution of the sync opcode.

SYNC signal must be held until the sequencer recognizes the state of the SYNC signal. This state is not held within the sequencer. If the sequencer has not executed the “sync” instruction prior to the sync input going active, it will not be recognized as having occurred. Therefore if it cannot be guaranteed that the sequencer will be in the sync state before the SYNC input goes high, the SYNC input must persist until it is cleared by the sequencer.

SEQ_SYNC_STATE register of the COP sub system contains sync status bits from the coprocessors, DMA, sequencer, DSP, and ARM. The status bit is set when a strobe pulse is received from the corresponding module. It is cleared by writing a 1 to the bit position. The status bits read back to ARM, DSP, and sequencer are pre-mask, that is, not affected by the setting of the SYNC_MSK registers. SEQ_SYNC_MASK register controls which of the sync status bits get ANDed / ORed to produce the 1-bit sync_seq signal sent to the sequencer.

There are two very specific differences between an interrupt and a sync that must be considered.

1. There is no dynamic reaction to the sync input signal being active.

2. Execution of the sync opcode sensitizes the sequencer to the SYNC input signal and it also causes program execution to pause until SYNC is received.

The possibility of missing any SYNC because of Sequencer being in RUN condition (Figure 12) can be avoided by persisting the SYNC until Sequencer clears it by writing a ‘1’ to the corresponding bit in the SEQ_SYNC_STATE register after entering the SYNC state as in Figure 13.

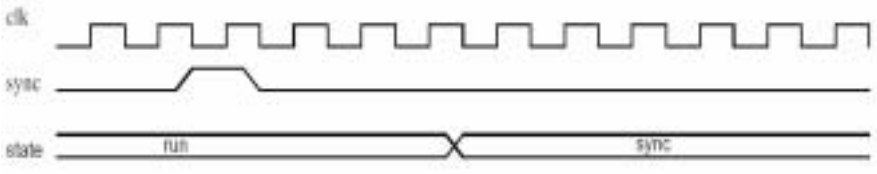


Figure 12: Sequencer in “run” state, sync does not persist and is missed

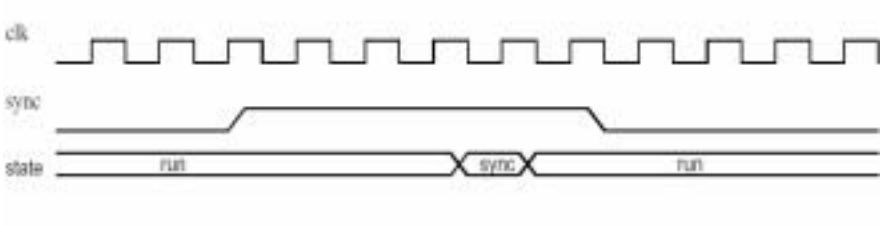


Figure 13: Sync with sequencer in "run" state, sync persists until cleared

6.6 Interrupt/Strobe

The interrupt to Sequencer is an active high signal which is high for one clock cycle and is synchronous to the Sequencer clock. A 16-bit interrupt vector provides the absolute branch location for the current interrupt. The Sequencer will register the interrupt input on the next clock cycle, therefore interrupt must be deasserted on the next clock cycle. On getting the interrupt signal the Sequencer will branch to the location provided as interrupt vector. Before branching Sequencer saves the return address of the instruction that would have been executed in this clock cycle and the current status register. SEQ_INT_VECTOR register of the Coprocessor subsystem contains the interrupt vector address.

The CP_INTC register of the coprocessor sub system facilitates the strobe generation by various coprocessors, ARM, DSP and Sequencer. Sequencer sends its strobe pulse by setting SEQSTRB bit of CP_INTC register. Similarly ARM and DSP can also generate strobe signals by setting corresponding bit in CP_INTC. As the Coprocessor Bus arbitrates and serializes accesses from ARM, DSP and sequencer, there is no coherence problem if ARM, DSP and Sequencer each write 1 to the

bit position it is assigned to. The CP_INTC register is write-only. Writing 1 sends a pulse**;** there is no need to write 0 to end the pulse.

6.7 Break-Point Function

Sequencer breakpoint feature is implemented by the BRKPT_TRG and BRKPT_CLR registers. These provides a way for inserting break points in the sequencer code, and have DSP to come in and examine coprocessors, image buffers, and so on upon these break points. Sequencer/DSP interaction for break points is shown in the Figure 14.

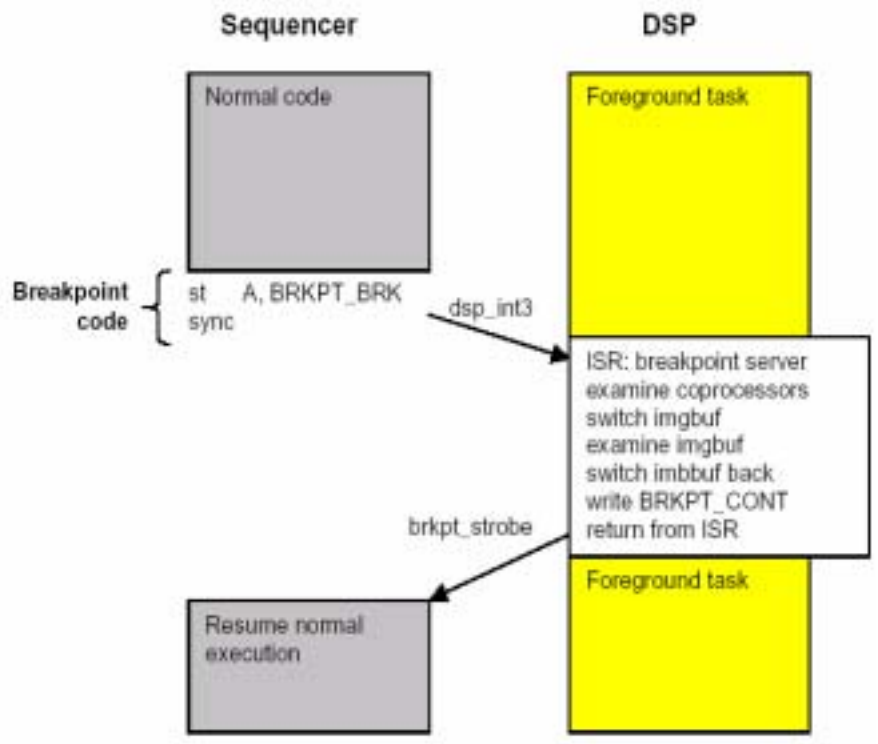


Figure 14: Sequencer/DSP interaction during break point

6.8 Assembly Language Tool Flow

Figure 15 shows the sequencer assembly language tool flow.

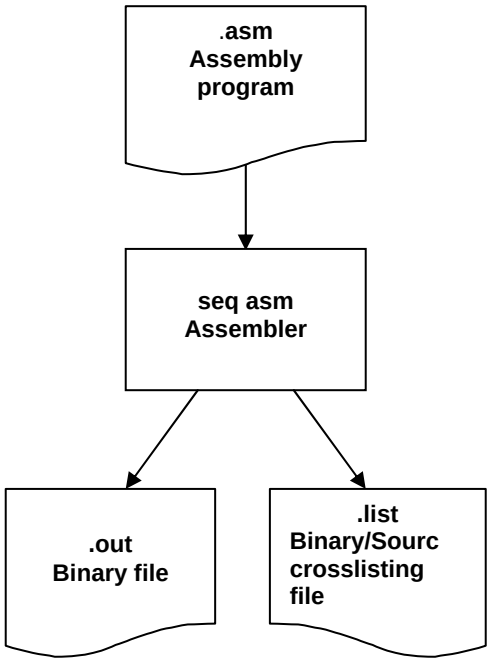


Figure 15: Sequencer Tool Flow

6.9 Assembler

A perl based assembler program, seqasm, is provided for development of Sequencer firmware. This is a two-pass assembler.

6.9.1 Assembler Directives

6.9.1.1 New Section (.sect)

Usage [Label[:]] .sect constant_argument

This directive defines the starting address of code that follows. The constant argument evaluates to this address. The label, if present, is assigned the address.

Example

```
Pgm_section:   .sect   0x100
Data_section:  .sect   0x400
```

6.9.1.2 Set Value(.Set)

Usage Label[:] .set constant_argument

This directive assigns a constant to the label. The constant argument evaluates to this constant.

Example

```
Control_reg0:   .set    0xA020

Intr_mask1:     .set    0xD
```

6.9.1.3 Short Data (.short)

Usage [Label[:]] .short constant_argument, constant_argument, …

This directive allocates one or more short (C terminology, short = 16-bit) data words. The label, if present, is assigned the starting address, or address of the first data word.

Example

```
Variable1:      .short  0

Array2          .short  2, 5, 8, 0x800

Array3          .short  0               ; want 256 element array, un-initialized

                .sect   Array3 + 256

Variable4       .short  -6              ; assembler assigns Variable4 = Array3 + 256
```

6.9.2 Invoking the Assembler

System requirement: Perl interpreter(*perl*)

Assembler is invoked by the following command:

perl seqasm program1.asm

The assembler reports any errors found, if none it generates two files:

program1.out : converted binary code, in hexadecimal format

program1.lst : assembly-binary cross listing text file

The following listing is an example cross listing file, which displays address, code, cumulative execution time (reset at every branch or label), and source assembly program.

```
                    ; test subroutine call/return

                    ; data section

                              .sect   0x100
0100 0000          0 arg1     .short  0
0101 0000          0 arg2     .short  0
0102 0000          0 result   .short  0
0103 0000          0 temp     .short  0

                    value1    .set    0x0011

                    ; program section

                              .sect   0

0000 4806 1234     3 main:    LD      #0x1234, A
0002 4906 0023     6          LD      #0x23, B
0004 4400 0100     8          ST      A, arg1
0006 4401 0101    10          ST      B, arg2
0008 e006 000f    13          BRI     add_func

000a 4804 0102    16          LD      result, A
000c 4400 0103    18          ST      A, temp

000e 3c00         19       STOP

                    add_func:
000f 4804 0100     3          LD      arg1, A
0011 8004 0101     7          ADD     arg2, A, A
0013 4400 0102     9          ST      A, result
0015 e400         11          RTN
```

Figure 16: Listing 1 Sample cross listing generated by the assembler tool

6.9.3 Source Statement Format

Each valid source line can be Empty, or white-space-only line Comment line, indicated by “*” at the first column Statement line, conforming to the following format:

[Label[:]] [operator] [argument] [,argument]* [;comments]

Note that The assembler is case insensitive. Thus, RowData, ROWDATA, and rowdata are the same to the assembler. Label must begin with the very first column; no preceding white space is allowed. Blank and tab are valid white spaces. There is at least one white space before the operator. There is at least one white space between the operator and the argument list (if present). There can be zero, one, or multiple arguments, separated by “,”.

6.9.4 Label

A label begins with a letter, and the remaining characters can be letter, number, or "_". The optional colon, ":", is part of label formatting, and not part of the name. A label definition associates the name with a value. Name in the label must NOT be one of the following reserved words:

A, B, C, P, *P, *P++ (and the lowercase versions)

Though permitted by the assembler, using assembler directives and instruction mnemonics as labels are confusing so should be avoided. A label can be defined only once in the source program.

As mentioned earlier **seqasm** is a two-pass assembler. On the first pass, all labels are defined and assigned values. On the second pass, binary code is generated. Thus, some dependency rules must be followed:

A label can be used before it is assigned value in the source program, but only for code generation

A label whose value is depended upon by a second label must be assigned before the second label is, in the source program

The following listing shows valid and invalid use of variables and labels.

```
jump_addr1  .short label1           ; forward reference, OK since jump_addr1
                                    ; doesn't depend on label1.  In pass 2 at
                                    ; this point, label1 will have been defined
constant1:  .set  constant2 + 2     ; invalid forward reference

label1:     LD    operand1, A       ; OK
            ADD   operand2, A, B    ; OK
            ST    B, result         ; OK
            RTN

operand1:   .short  3
operand2:   .short  5
result:     .short  7

constant2:  .set  0x4567
constant3:  .set  constant2 + 0x20  ; OK, constant2 is defined at this point
                                    ; in pass 1
```

Figure 17: Listing 2 Variable and label example

Note that labels are used either to represent some constant value, like #define in C, or to point to memory data object, like the use of variables in C. The assembler does not distinguish these two usages, though the label's value is, in the first case, a value, or in the second case, an address. The programmer must be aware of such distinction.

6.9.5 Operator

The operator is either an assembler directive or an instruction mnemonic. They are listed in Sections 6.9.1 and 6.4 respectively.

6.9.6 Argument

An argument provides a value to a label, to an assembler, or to an instruction. Presence or absence of prefix "#" is needed in some instructions to distinguish between immediate and address.

An argument can be a numeric constant, a variable, a reserved processor reference (register, *P, *P++), or an expression involving numeric constants and variables.

The following are examples of valid arguments:

5 (numeric constant)

loop_count (variable)

B (processor register)

*P (memory operand referenced by register P)

(loop_count + 0x20) (expression)

6.9.7 Numeric Constant

A numeric constant can be decimal or hexadecimal (with prefix 0x), and optionally can carry a minus sign. All numeric constants are evaluated to 16-bit. Only integers are allowed.

The following are examples of valid numeric constants:

2

0x3a00

190

–0x56AA

6.9.8 Expression

This assembler supports add/subtract expressions with one level of parenthesis. An expression must be evaluated in the first pass if depended upon by a label. Otherwise it can be evaluated in the second pass.

The following are examples of valid expressions:

#(450 + 399)

(data_array + 0x80)

(brach_target + offset3 – 4)

6.10 Sequencer Register Map (SEQ)

Arm, DSP and sequencer can access the control registers and control the sequencer operations.

DSP	SEQ	ARM	Register	Description
0200	F200	0009:E400	CTRL	Sequencer control register
0201	F201	0009:E402	BOOT	Sequencer boot address register
0202	F202	0009:E404	RSV	reserved
0203	F203	0009:E406	AREG	A register of Sequencer (debug)
0204	F204	0009:E408	BREG	B register of Sequencer (debug)
0205	F205	0009:E40A	CREG	C register of Sequencer (debug)
0206	F206	0009:E40C	PREG	P register of Sequencer (debug)
0207	F207	0009:E40E	PCREG	PC register of Sequencer (debug)
0208	F208	0009:E410	STATUS	Status register of Sequencer (debug)
0209	F209	0009:E412	CONFIG	Configuration of Sequencer

6.11 Sequencer Registers

The details of the various control registers of Sequencer are given below.

6.11.1 CTRL

Sequencer control register

CTRL		DSP	IO	0200	offset:	0x00									default:	0x0000
		SEQ		F200												
		ARM		0009:E400												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	ABORT	RUN
R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0

Bit	Name	Reset Value	R/W	Function
15:2	RSV			Reserved
1	ABORT	0	R/W	Sequencer abort control: 0: running 1: abort Note: Abort is equivalent to stop command in assembly
0	RUN	0	R/W	Sequencer run control: 0: idle 1: start

6.11.2 BOOT

Sequencer program start address registers

BOOT		DSP	IO	0201	offset:	0x01									default:	0x0000
		SEQ		F201												
		ARM		0009:E402			Sequencer boot address register									
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	ADDR[10]	ADDR[9]	ADDR[8]	ADDR[7]	ADDR[6]	ADDR[5]	ADDR[4]	ADDR[3]	ADDR[2]	ADDR[1]	ADDR[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	ADDR	0	R/W	Sequencer Boot address, Sequencer will start executing by fetching the first instruction from this address

6.11.3 Reserved – Reserved Register

RSV		DSP	IO	0202	offset:	0x02									default:	0x0000
		SEQ		F202												
		ARM		0009:E404												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Default	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit	Name	Reset Value	R/W	Function
15:0	RSV			Reserved

6.11.4 AREG

Sequencer A Register contents

AREG		DSP	IO	0203	offset:	0x03									default:	0x0000
		SEQ		F203												
		ARM		0009:E406												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	AREG[15]	AREG[14]	AREG[13]	AREG[12]	AREG[11]	AREG[10]	AREG[9]	AREG[8]	AREG[7]	AREG[6]	AREG[5]	AREG[4]	AREG[3]	AREG[2]	AREG[1]	AREG[0]
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	AREG	0	R	Sequencer A register contents Note: Should only be read if sequencer is halted using Sync or Stop command

6.11.5 BREG

Sequencer B Register contents

BREG		DSP	IO	0204	offset:	0x04									default:	0x0000
		SEQ		F204												
		ARM		0009:E408												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	BREG[15]	BREG[14]	BREG[13]	BREG[12]	BREG[11]	BREG[10]	BREG[9]	BREG[8]	BREG[7]	BREG[6]	BREG[5]	BREG[4]	BREG[3]	BREG[2]	BREG[1]	BREG[0]
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	BREG	0	R	Sequencer B register contents Note: Should only be read if sequencer is halted using Sync or Stop command

6.11.6 CREG

Sequencer C Register contents

CREG		DSP	IO	0205	offset:	0x05									default:	0x0000
		SEQ		F205												
		ARM		0009:E40A												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	CREG[15]	CREG[14]	CREG[13]	CREG[12]	CREG[11]	CREG[10]	CREG[9]	CREG[8]	CREG[7]	CREG[6]	CREG[5]	CREG[4]	CREG[3]	CREG[2]	CREG[1]	CREG[0]
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	CREG	0	R	Sequencer C register contents Note: Should only be read if sequencer is halted using Sync or Stop command

6.11.7 PREG

Sequencer P Register contents

PREG		DSP	IO	0206	offset:	0x06									default:	0x0000
		SEQ		F206												
		ARM		0009:E40C												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	PREG[15]	PREG[14]	PREG[13]	PREG[12]	PREG[11]	PREG[10]	PREG[9]	PREG[8]	PREG[7]	PREG[6]	PREG[5]	PREG[4]	PREG[3]	PREG[2]	PREG[1]	PREG[0]
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	PREG	0	R	Sequencer P register contents Note: Should only be read if sequencer is halted using Sync or Stop command

6.11.8 PCREG

Sequencer PC Register contents

PCREG		DSP	IO	0207	offset:	0x07									default:	0x0000
		SEQ		F207												
		ARM		0009:E40E												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	PCREG[15]	PCREG[14]	PCREG[13]	PCREG[12]	PCREG[11]	PCREG[10]	PCREG[9]	PCREG[8]	PCREG[7]	PCREG[6]	PCREG[5]	PCREG[4]	PCREG[3]	PCREG[2]	PCREG[1]	PCREG[0]
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:0	PCREG	0	R	Sequencer PC register contents Note: Should only be read if sequencer is halted using Sync or Stop command

6.11.9 STATUS

Sequencer status register

STATUS		DSP	IO	0208	offset:	0x08									default:	0x0000
		SEQ		F208												
		ARM		0009:E410												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	V	N	Z	C
R/W	-	-	-	-	-	-	-	-	-	-	-	-	R	R	R	R
Default	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:4	RSV			Reserved
3	V	0	R	0: Arithmetic instruction casued no overflow 1: Arithmetic instruction casued an overflow
2	N	0	R	0: Data value last loaded into a register had MSB '0' 1: Data value last loaded into a register had MSB '1'
1	Z	0	R	0: Data value last loaded into a register was non-zero 1: Data value last loaded into a register was zero
0	C	0	R	0: Arithmetic instruction casued no carry 1: Arithmetic instruction casued a carry

6.11.10 CONFIG

Sequencer configuration register

CONFIG		DSP	IO	0209	offset:	0x09									default:	0x0000
		SEQ		F209												
		ARM		0009:E412												

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ALU[7]	ALU[6]	ALU[5]	ALU[4]	ALU[3]	ALU[2]	ALU[1]	ALU[0]	VER[7]	VER[6]	VER[5]	VER[4]	VER[3]	VER[2]	VER[1]	VER[0]
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:8	ALU	0	R	Type of ALU Note: Each ALU design is given a type number to uniquely identify the contents of the ALU.
7:0	VER	0	R	Version number of Sequencer

7 DCT/IDCT

7.1 Introduction

In JPEG and MPEG-1/2/4 compression standards, DCT (discrete cosine transform) is specified as the method in the compression flow to transform blocks of image in the spatial domain to the 2-D frequency domain. DCT concentrates the signal energy to the lower frequency components to facilitate compression. IDCT (inverse discrete cosine transform) is numerically the inverse of DCT, and is used in the decompression flow to recover image from frequency domain data.

In these image/video coding standards, fixed size, 8x8 2-D DCT/IDCT are used. Consequently, the computation required is a standard and fixed function, it is advantageous to implement DCT/IDCT in hardware, as opposed to on programmable microprocessor or DSP. Hardware implementation offers better cost/performance/power tradeoffs than software implementation.

The DCT module in DM320 performs 2-D 8x8 DCT and IDCT. The DCT module and other coprocessors together provide high performance and power-efficient imaging and video coding and decoding capabilities in DM320.

The DCT performance in DM320 is 144 coprocessor clock cycles for a 8x8 block of data, which means 2.25 coprocessor clock cycles per DCT coefficient for forward or inverse 2-D transforms.

7.2 Features of DM320 DCT

The DM320 DCT module possess the following features:

- Complies with the 8x8 2-D DCT and IDCT standard specified in JPEG, MPEG-1, MPEG-2, MPEG-4, and H.263 image/video coding standards.
- DCT assumes 16-bit per data item in input, output, and intermediate result (in between row and column transforms).
- The addressing control is optimized for most frequently used cases: 4:2:2, and 4:2:0 YUV format. It also supports a sequential 8x8 blocks mode to deal with any other formats, but data has to be placed/dealt with the spatial domain in sequential 8x8 block order.
- In 4:2:2, the spatial domain data is organized as 1 Luma block of 16x8 and 2 Chroma blocks of 8x8, and the frequency domain, 4 blocks of 8x8.
- In 4:2:0, the spatial domain data is organized as 1 Luma block of 16x16 and 2 Chroma blocks of 8x8, and the frequency domain, 6 blocks of 8x8.
- In the sequential 8x8 block mode, both spatial and frequency domain data are organized as N blocks of 8x8 blocks, with the restriction that N >= 2.

- The frequency domain can optionally be formatted the same as the spatial domain (16x16, 16x8 or 8x8). This is in case the Q or IQ step after DCT or before IDCT is hard-coded to deal with the spatial domain format.
- Output is normally saturated to the range [-2048, 2047] for DCT and to the range [-256, 255] for IDCT, and can optionally be disabled in the register interface.

Note: The data formats specified are with respect to spatial domain and frequency domain, not tied to input and output. Thus, a DCT task followed by an IDCT task can keep the same configuration, to produce final IDCT output in the same format as the input to DCT.

7.3 Luma/chroma formats

Luma/Chroma formats are shown in the following diagrams.

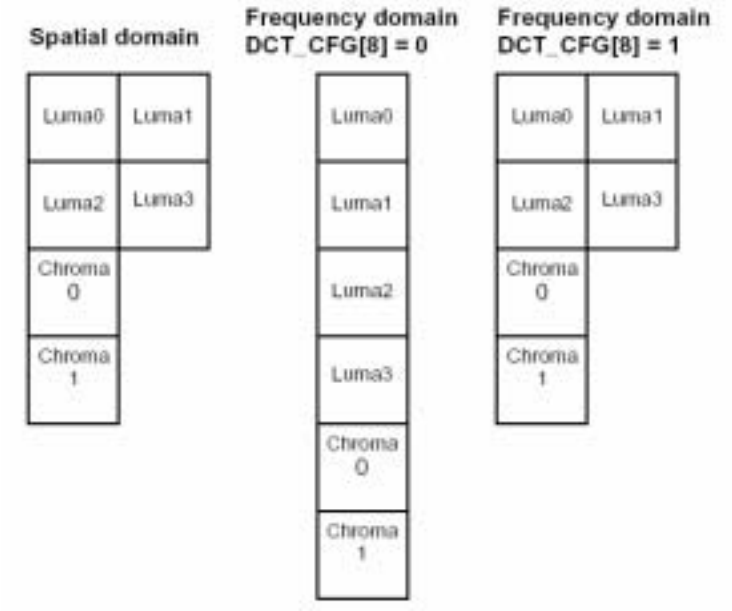


Figure 18: 4:2:0 Format

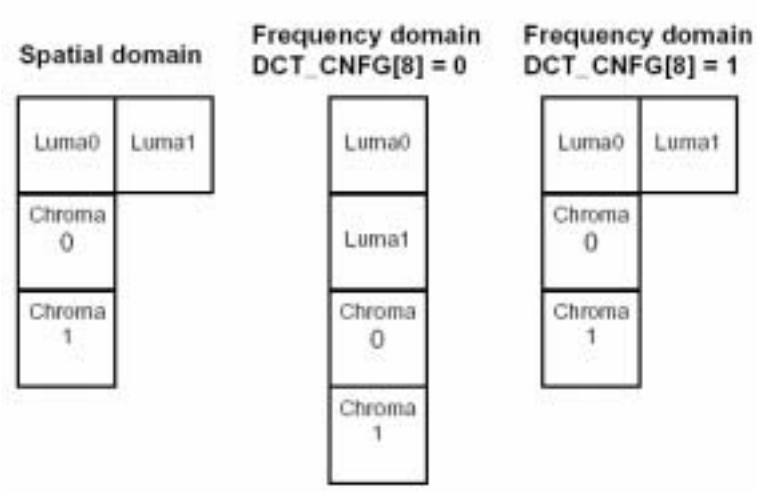


Figure 19: 4:2:2 Format

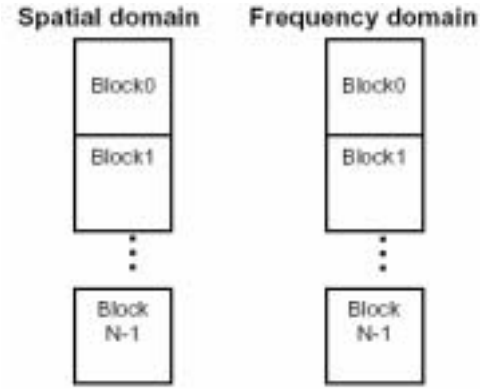


Figure 20: Sequential Format

7.4 Interrupt

DCT can generate sync signal to DSP, ARM or Sequencer upon completion of the task. INT bit of the CFG register enables or disables the interrupt generation. Refer the section of Coprocessor Interrupt Controller in Coprocessor subsystem chapter.

7.5 Precision

The DM320 DCT precision coefficients are given in Table 23 below.

Characteristic	IDCT	IEEE Requirement for IDCT	DCT
Peak error any coefficient	1.0	<=1.0	1.0
Max mean square error (any location of 8x8)	0.0135	<= 0.06	0.0655
Mean square error over all coefficients	0.009884	<= 0.02	0.017994
Max mean error (any location of 8x8)	0.0025	<= 0.015	0.0655
Mean error over all coefficients	0.000181	<= 0.0015	0.005697

Table 23: DM320 DCT precision coefficients

7.6 DCT Register map

The DCT registers are mapped in ARM, DSP and Sequencer address spaces.

DSP	SEQ	ARM	Register	Description
0180	F180	0009:E300	CTRL	DCT control register
0181	F181	0009:E302	CFG	DCT configuration register
0182	F182	0009:E304	IPTR	Input pointer (byte address)
0183	F183	0009:E306	IMPTR	Intermediate pointer (byte address)
0184	F184	0009:E308	OPTR	Output pointer (byte address)
0185	F185	0009:E30A	OPTR2	Secondary Output pointer (byte address)

7.7 DCT Registers

7.7.1 CTRL

DCT Control Register

CTRL		DSP	IO	0180	offset:	0x00									default:	0x0000
		SEQ		F180												
		ARM		0009:E300												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV	START_DI	START_I	START_D
R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0

Bit	Name	Reset Value	R/W	Function
15:3	RSV			Reserved
2	START_DI	0	R/W	Start DCT-IDCT / status: » 0: idle « » 1: Start / Busy «
1	START_I	0	R/W	Start IDCT / status: » 0: idle « » 1: Start / Busy «
0	START_D	0	R/W	Start DCT / status: » 0: idle « » 1: Start / Busy «

7.7.2 CFG

DCT Configuration Register

CFG		DSP	IO	0181	offset:	0x01									default:	0x0000
		SEQ		F181												
		ARM		0009:E302												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	RSV	RSV	FREQ	NUM[3]	NUM[2]	NUM[1]	NUM[0]	FORMAT[1]	FORMAT[0]	SAT	INT
R/W	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:9	RSV			Reserved
8	FREQ	0	R/W	Frequency domain block organization » 0: Frequency domain organized as sequential 8x8 blocks « » 1: Frequency domain orgranzed spatially (16x16, 16x8 or 8x8) «
7:4	NUM	0	R/W	Number of sequential 8x8 blocks - 1 for the sequential 8x8 format Note: Num of blocks >= 2
3:2	FORMAT	0	R/W	Format of Data » 00: 4:2:0 format « » 01: 4:2:2 format « » 10: Sequential 8x8 format «
1	SAT	0	R/W	Saturation mode » 0: Enable saturation « » 1: Disable saturation «
0	INT	0	R/W	Interrupt mode » 0: Interrupt disabled « » 1: Interrupt enabled «

7.7.3 IPTR

DCT input pointer Register

IPTR		DSP	IO	0182	offset:	0x02									default:	0x0000
		SEQ		F182												
		ARM		0009:E304												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	IPTR[10]	IPTR[9]	IPTR[8]	IPTR[7]	IPTR[6]	IPTR[5]	IPTR[4]	IPTR[3]	IPTR[2]	IPTR[1]	IPTR[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	IPTR	0	R/W	Input pointer Note1: This is an 32-bit per word address and should be 32-bit aligned Note2: For DCT-then-IDCT mode, applies to input to DCT

7.7.4 IMPTR

DCT intermediate Pointer Register

IMPTR		DSP	IO	0183	offset:	0x03									default:	0x0000
		SEQ		F183												
		ARM		0009:E306												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	IMPTR[10]	IMPTR[9]	IMPTR[8]	IMPTR[7]	IMPTR[6]	IMPTR[5]	IMPTR[4]	IMPTR[3]	IMPTR[2]	IMPTR[1]	IMPTR[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	IMPTR	0	R/W	Intermediate pointer Note1: This is an 32-bit per word address and should be 32-bit aligned Note2: For DCT-then-IDCT mode, applies to DCT and IDCT

7.7.5 OPTR

DCT output pointer Register

OPTR		DSP	IO	0184	offset:	0x04									default:	0x0000
		SEQ		F184												
		ARM		0009:E308												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	OPTR[10]	OPTR[9]	OPTR[8]	OPTR[7]	OPTR[6]	OPTR[5]	OPTR[4]	OPTR[3]	OPTR[2]	OPTR[1]	OPTR[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	OPTR	0	R/W	Output pointer Note1: This is an 32-bit per word address and should be 32-bit aligned Note2: For DCT-then-IDCT mode, applies to DCT output, which is also an input to IDCT

7.7.6 OPTR2

DCT output secondary pointer

OPTR2		DSP	IO	0185	offset:	0x05									default:	0x0200
		SEQ		F185												
		ARM		0009:E30A												
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	RSV	RSV	RSV	RSV	RSV	OPTR2[10]	OPTR2[9]	OPTR2[8]	OPTR2[7]	OPTR2[6]	OPTR2[5]	OPTR2[4]	OPTR2[3]	OPTR2[2]	OPTR2[1]	OPTR2[0]
R/W	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	-	-	-	-	-	0	1	0	0	0	0	0	0	0	0	0

Bit	Name	Reset Value	R/W	Function
15:11	RSV			Reserved
10:0	OPTR2	512	R/W	Secondary Output pointer Note1: This is an 32-bit per word address and should be 32-bit aligned Note2: For DCT-then-IDCT mode, applies to IDCT output

8 Appendix B

8.1 Introduction

The Sequencer can be used to coordinate tasks between iMX, VLCD, DCT, COP DMA, to reduce the scheduling load on ARM and DSP. The sequencer program can be tailored for different results like best execution time or minimum code size etc. By utilizing the pipelining possibilities to the best, minimum execution time can be achieved. But this approach may not be the best in terms of code size and may need more frequent intervention of the system processors. On the other hand Sequencer code can be tailored to achieve minimum code size or minimum intervention from system processors, at the cost of execution time. The trade off can be determined based on the application.

The following sequencer example illustrates a typical scenario in digital still image processing. Pipeline planning and construction of concurrent schedule is based on estimated or measured computation time. DM320 still image capture sequence is given in Table 24.

Note: Figures shown below are only a rough estimate.

Description	**Performance estimate in 180 MHz clock cycles**
Copy 32x24 data block from SDRAM to image buffer	2.5 cyc/pxl (DMA)
Fault pixel (50 per 2M)	0.07 cyc/pxl (Sequencer)
Clamp / white balance	0.38 cyc/pxl (iMX)
Gamma LUT	3 cyc/pix (iMX)
History buffer management	0.09 cyc/pxl (iMX)
CFA (7x7 FIR on G, 7-tap separable on R/B)	8.79 cyc/pxl (iMX)
Color space transformation	2.11 cyc/pxl (iMX)
Edge enhancement / coring / false Color suppression	3 cyc/pxl (iMX)
Color subsampling	0 cyc/pxl (iMX)
JPEG: DCT	3.0 cyc/pxl (DCT)
JPEG: Quantization	3.0 cyc/pxl (VLCD)
JPEG: Huffman	2.63 cyc/pxl (VLCD)
Copy bitstream to SDRAM	0.21 cyc/pxl (DMA)
Total iMX + 10%	19.11 cyc/pixel -> 9.42 Mpix/sec
Total other tasks + 10%	12.55 cyc/pixel

Table 24: DM320 Still Image Capture Sequence

8.2 Programming / Performance Example

To simplify the schedule, we shall use 16x16-processing blocks, and skip the faulty pixel correction step. Using 16x16 block size, versus 32x16 will incur higher percentage overhead, so the above cycles per pixel numbers shall increase slightly. Since this is just an example, we ignore this effect and just scale these cycles per pixel numbers by 256 to get the processing time for each 16x16 pixel block. What we get is the following approximate computation times for each block. Each coprocessor and DMA happen to gets a consecutive group of tasks, and in DM320, requires just one "go" command interaction from the sequencer per data block.

Task	**Clock cycles (in 100MHz sequencer clock)**
DMA input	400
iMX	2800
DCT	500
VLCD (with DMA transferring the bit stream back to SDRAM)	900

Table 25: DM320 Still Image Capture Computation Time Summary

8.2.1 Concurrent Schedule

An efficient concurrent schedule is sketched out as follows. Note the use of image buffers A and B to support data passing and concurrency.

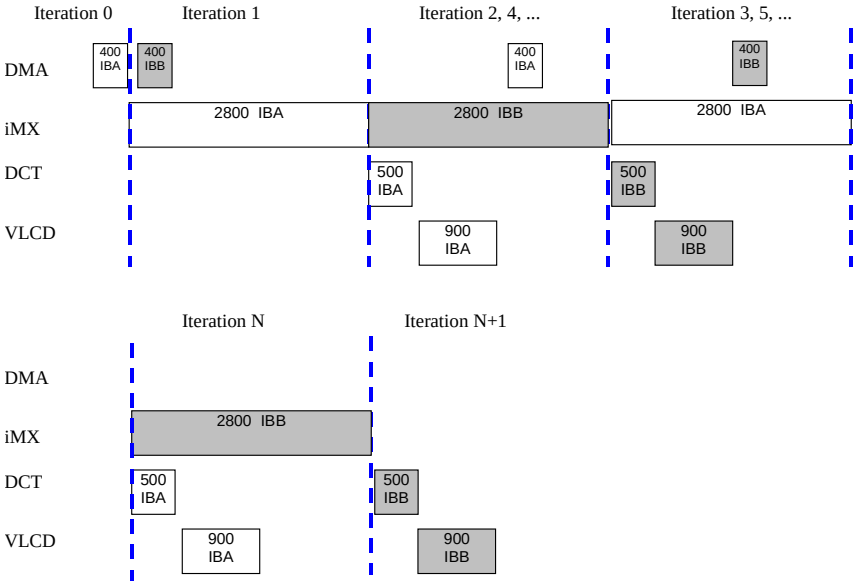


Figure 21: Concurrent Schedule forDM320 Still Image Capture

We can see there are 6 kinds of iterations.

- Iteration 0 has just the DMA task
- Iteration 1 has DMA and iMX
- Iteration 2, 4, and so on goes into steady state, with iMX on Image Buffer B and the others on Image Buffer A. Iteration 3, 5, and so on are steady state as well, with opposite buffer assignment compared to the even iterations. If N, number of blocks, is even, we'll start iteration N ramp down with iMX on Buffer B and DCT, VLCD on Buffer A.
- Iteration N+1 is the last iterartion, with DCT and VLCD on Buffer B.

We shall use the most straightfoward approach of coding all these 6 iterations. DSP only needs to provide the SDRAM DMA starting address, and initiate sequencer with the correct starting address out of these 6 iterartions. We can see that we can ***pair up*** every 2 iterations, and cut number of DSP interactions by half. The SDRAM DMA starting address is linearly incrementing, so DSP can easily precompute 2 addresses per interaction.

8.2.2 Sequencer Routines

The 3 sequencer routines will be called seq1, seq2, and seq3:

- Seq1 contains iterations 0 and 1, the loop prolog
- Seq2 contains the even and odd iterations of steady-state loop
- Seq3 contains iterations N and N+1, the loop epilog

The following diagram shows DSP/sequencer interactions in DM320 still image capture flow, for N=6.

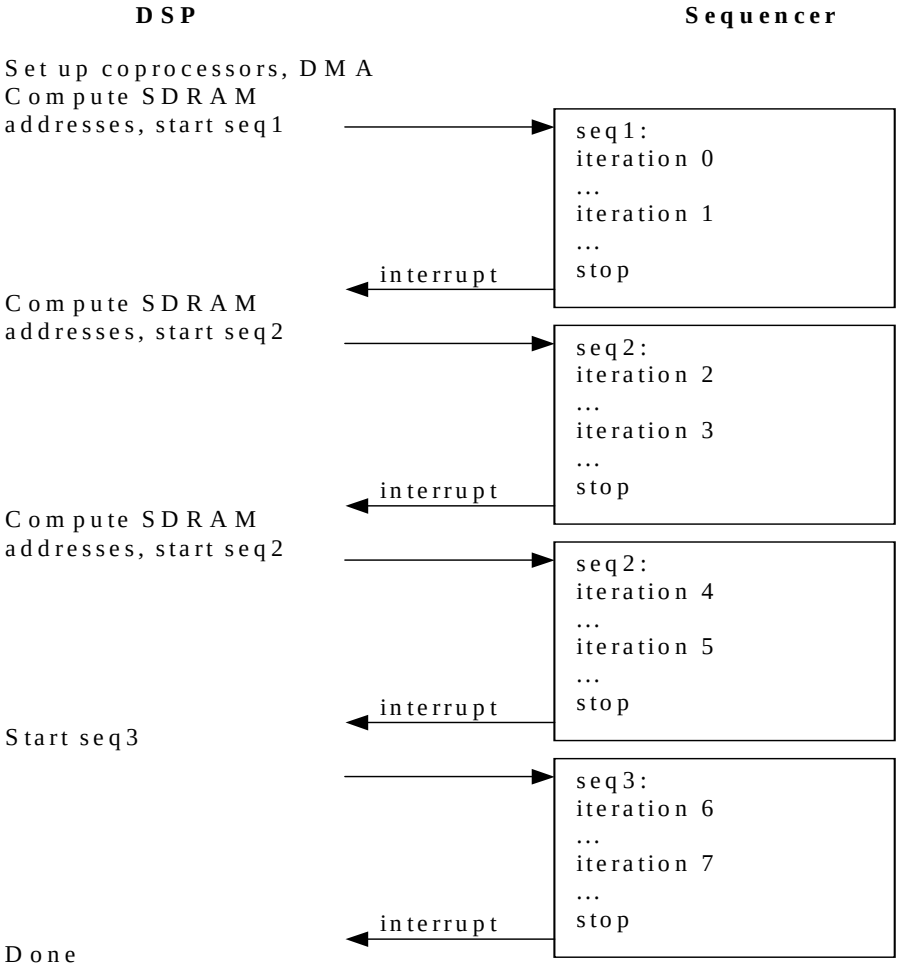


Figure 22: DSP/Sequencer Interactions during Still Image Capture

Each of the 3 segments of sequencer code can be planned out by looking at the corresponding segment in the concurrent schedule, and starting coprocessor, waiting for completion, switching buffers, and stopping at the appropriate sequence. The actions are annotated on the concurrent schedule diagram, then the sequencer code is shown.

8.2.2.1 Seq1 Routine: Iterations 0 and 1

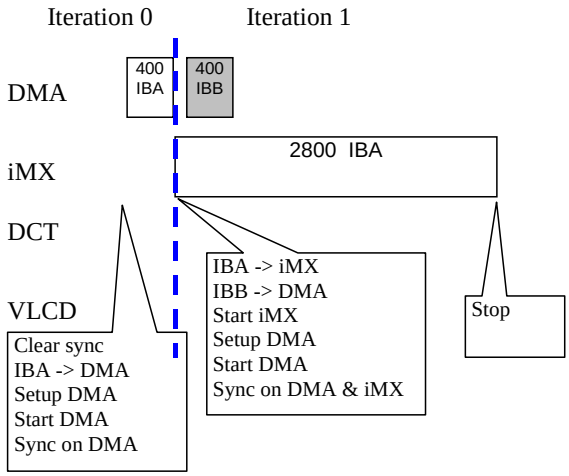


Figure 23: Concurrent tasks in Seq1 routine

```
ipipe_seq_start1:                       ; iterations 0 and 1

                ; ================== iteration 0 ======================

                ld      #SYNCMSK_ALL, A     ; clear sync
                st      A, IBF_SEQ_SYNC

                ld      #ibufsw_even3, A    ; IBA->DMA
                st      A, IBF_BUF_MUX0

                ld      sdr_ptr_1L, A       ; set up SDRAM addr in DMA
                st      A, IBF_SDEM_ADDRL
                ld      sdr_ptr_1H, A
                st      A, IBF_SDEM_ADDRH
                ld      #DMA_TIME, A        ; start DMA
                st      A, IBF_DMA_CTRL

                ld      #SYNCMSK_DMA, A     ; set up sync mask for DMA
                st      A, IBF_SEQ_SYNCMSK
                sync                        ; wait for DMA to complete

                ; ================== iteration 1 ======================

                ld      #SYNCMSK_ALL, A     ; clear sync
                st      A, IBF_SEQ_SYNC

                ld      #ibufsw_odd3, A     ; IBB->DMA, IBA->iMX
                st      A, IBF_BUF_MUX0

                ld      #IMX_TIME, A        ; start iMX
                st      A, IMX_START_REG

                ld      sdr_ptr_2L, A       ; set up SDRAM addr in DMA
                st      A, IBF_SDEM_ADDRL
                ld      sdr_ptr_2H, A
                st      A, IBF_SDEM_ADDRH
                ld      #DMA_TIME, A        ; start DMA
                st      A, IBF_DMA_CTRL

                ld      #(SYNCMSK_DMA + SYNCMSK_IMX), A  ; mask = DMA | iMX
                st      A, IBF_SEQ_SYNCMSK
                sync                        ; wait for both DMA and iMX

                  stop
```

Figure 24: Seq1 Sequencing Code

8.2.2.2 Seq2 Routine: Two Iterations in Steady State

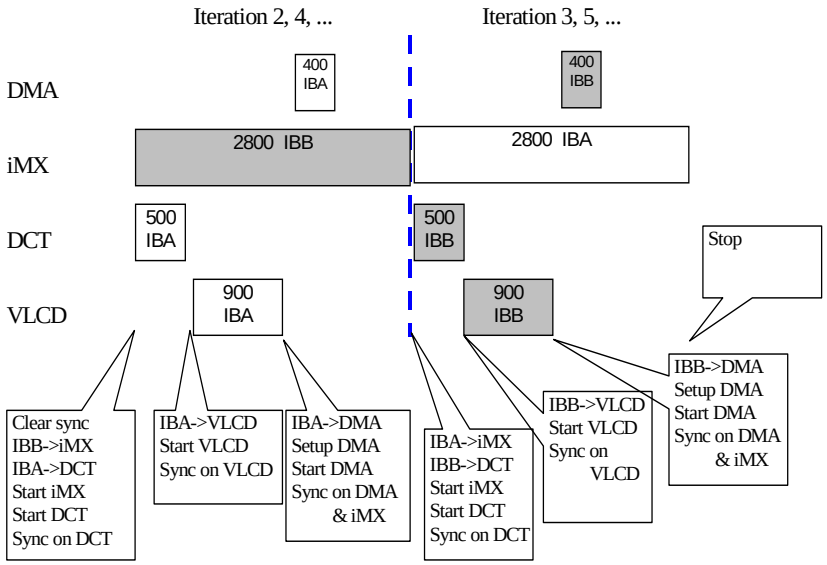


Figure 25: Concurrent tasks in Seq2 routine

```
ipipe_seq_start2:                          ; iterations 2i and 2i+1
                ; ================== iteration 2i ======================

                ld      #SYNCMSK_ALL, A     ; clear sync
                st      A, IBF_SEQ_SYNC

                ld      #ibufsw_even1, A    ; IBA->DCT, IBB->iMX
                st      A, IBF_BUF_MUX0

                ld      #IMX_TIME, A        ; start iMX
                st      A, IMX_START_REG

                ld      #DCT_TIME, A        ; start DCT
                st      A, DCT_START_REG

                ld      #SYNCMSK_DCT, A     ; set up sync mask for DCT
                st      A, IBF_SEQ_SYNCMSK
                sync                        ; wait for DCT to complete

                ld      #ibufsw_even2, A    ; IBA->VLCD, IBB->iMX
                st      A, IBF_BUF_MUX0

                ld      #VLCD_TIME, A       ; start VLCD
                st      A, VLCD_START_REG

                ld      #SYNCMSK_VLCD, A    ; set up sync mask for VLCD
                st      A, IBF_SEQ_SYNCMSK
                sync                        ; wait for VLCD to complete

                ld      #ibufsw_even3, A    ; IBA->DMA, IBB->iMX
                st      A, IBF_BUF_MUX0

                ld      sdr_ptr_1L, A       ; set up SDRAM addr in DMA
                st      A, IBF_SDEM_ADDRL
                ld      sdr_ptr_1H, A
                st      A, IBF_SDEM_ADDRH
                ld      #DMA_TIME, A        ; start DMA
                st      A, IBF_DMA_CTRL

                ld      #(SYNCMSK_DMA + SYNCMSK_IMX), A  ; mask = DMA | iMX
                st      A, IBF_SEQ_SYNCMSK
                sync                        ; wait for both DMA and iMX
```

Figure 26: Seq2 Sequencing Code First Part

```
; ================== iteration 2i+1 ======================

ld      #SYNCMSK_ALL, A     ; clear sync
st      A, IBF_SEQ_SYNC

ld      #ibufsw_odd1, A     ; IBB->DCT, IBA->iMX
st      A, IBF_BUF_MUX0

ld      #IMX_TIME, A        ; start iMX
st      A, IMX_START_REG

ld      #DCT_TIME, A        ; start DCT
st      A, DCT_START_REG

ld      #SYNCMSK_DCT, A     ; set up sync mask for DCT
st      A, IBF_SEQ_SYNCMSK
sync                        ; wait for DCT to complete

ld      #ibufsw_odd2, A     ; IBB->VLCD, IBA->iMX
st      A, IBF_BUF_MUX0

ld      #VLCD_TIME, A       ; start VLCD
st      A, VLCD_START_REG

ld      #SYNCMSK_VLCD, A    ; set up sync mask for VLCD
st      A, IBF_SEQ_SYNCMSK
sync                        ; wait for VLCD to complete

ld      #ibufsw_odd3, A     ; IBB->DMA, IBA->iMX
st      A, IBF_BUF_MUX0

ld      sdr_ptr_2L, A       ; set up SDRAM addr in DMA
st      A, IBF_SDEM_ADDRL
ld      sdr_ptr_2H, A
st      A, IBF_SDEM_ADDRH
ld      #DMA_TIME, A        ; start DMA
st      A, IBF_DMA_CTRL

ld      #(SYNCMSK_DMA + SYNCMSK_IMX), A  ; mask = DMA | iMX
st      A, IBF_SEQ_SYNCMSK
sync                        ; wait for both DMA and iMX

stop
```

Figure 27: Seq2 Sequencing Code Second Part

8.2.2.3 Seq3 Routine: Ramp-Down Iterations, N and N+1

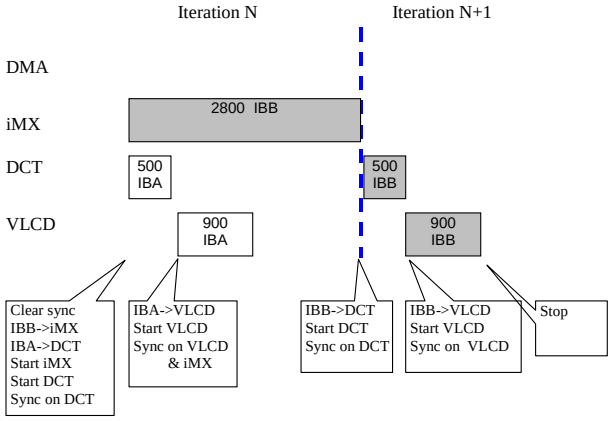


Figure 28: Sequencing Tasks on Concurrent Schedule, Iterations N and N+1

```
ipipe_seq_start3:                    ; iterations N and N+1
                ; ================= iteration N ======================
                ld      #SYNCMSK_ALL, A     ; clear sync
                st      A, IBF_SEQ_SYNC

                ld      #ibufsw_even1, A    ; IBA->DCT, IBB->iMX
                st      A, IBF_BUF_MUX0

                ld      #IMX_TIME, A        ; start iMX
                st      A, IMX_START_REG

                ld      #DCT_TIME, A        ; start DCT
                st      A, DCT_START_REG

                ld      #SYNCMSK_DCT, A     ; set up sync mask for DCT
                st      A, IBF_SEQ_SYNCMSK
                sync                        ; wait for DCT to complete

                ld      #ibufsw_even2, A    ; IBA->VLCD, IBB->iMX
                st      A, IBF_BUF_MUX0

                ld      #VLCD_TIME, A       ; start VLCD
                st      A, VLCD_START_REG

                ld      #(SYNCMSK_VLCD + SYNCMSK_IMX), A ; mask = VLCD | iMX
                st      A, IBF_SEQ_SYNCMSK
                sync                        ; wait for VLCD and iMX to complete

                ; ================= iteration N+1 ====================
                ld      #SYNCMSK_ALL, A     ; clear sync
                st      A, IBF_SEQ_SYNC

                ld      #ibufsw_odd1, A     ; IBB->DCT
                st      A, IBF_BUF_MUX0

                ld      #DCT_TIME, A        ; start DCT
                st      A, DCT_START_REG

                ld      #SYNCMSK_DCT, A     ; set up sync mask for DCT
                st      A, IBF_SEQ_SYNCMSK
                sync                        ; wait for DCT to complete

                ld      #ibufsw_odd2, A     ; IBB->VLCD
                st      A, IBF_BUF_MUX0

                ld      #VLCD_TIME, A       ; start VLCD
                st      A, VLCD_START_REG

                ld      #SYNCMSK_VLCD, A    ; set up sync mask for VLCD
                st      A, IBF_SEQ_SYNCMSK
                sync                        ; wait for VLCD to complete

                stop
```

Figure 29: Seq3 Sequencing Code